UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO

FACULTAD DE INGENIERIA

DIVISION DE ESTUDIOS DE POSGRADO

TEMA:

IMPLANTACION ELECTRONICA DE LA TRANSFORMADA DISCRETA COSENO

(D. C. T.)

Proyecto: "Nepohualtzintzin"

Director del proyecto: Dr. Francisco García Ugalde.

Realizador: Ing. Abel Clemente Reyes.

UNAM – Dirección General de Bibliotecas Tesis Digitales Restricciones de uso

I N D I C E

I. INTRODUCCION.

Hoy por hoy, el desarrollo alcanzado en las ciencias de la computación ha permitido revolucionar notablemente a la sociedad en la que vivimos, máxime considerando la gran cantidad de avances registrados en las diferentes ramas de las ciencias y la ingeniería. Resulta interesante notar que el común denominador de tales avances es, sin lugar a dudas, el procesamiento digital de los datos y la información que habitualmente se encuentran presentes en las señales que son consideradas como entradas a un determinado sistema. Esto último sienta las bases para afirmar que actualmente una de las áreas que ha presentado un mayor auge dentro de la ingeniería, es la que precisamente se relaciona con lo que se conoce como *Procesamiento Digital de Señales*, el cual usualmente se denomina como DSP (*Digital Signal Processing*).

Ahora bien, se puede entender que el procesamiento digital de señales se refiere al tratamiento algebraico efectuado ya sea sobre secuencias de números, o sobre vectores numéricos, los cuales generalmente representan señales que son externas al sistema digital en uso [GORD88]. Siendo esta clase de procesamiento el utilizado en gran variedad de aplicaciones prácticas tales como los sistemas destinados a las telecomunicaciones, el radar, el sonar, el tratamiento de voz, las imágenes biomédicas y la sismología entre otros casos.

Es frecuente considerar al procesamiento digital de señales como una herramienta de gran utilidad al momento de tratar una cierta señal, ya sea para filtrarla digitalmente o, en su caso, para efectuar sobre ella un análisis espectral; sin embargo, conviene aclarar que dicho análisis se puede asociar al filtrado de datos mediante una colección de filtros pasobanda, lo cual obliga a pensar que la operación más importante al realizar esta clase de procesamiento es, justamente, la convolución o correlación de datos numéricos.

Por otra parte, al considerar la dualidad existente entre los dominios del tiempo y la frecuencia, así como las propiedades inherentes de la convolución, resulta explicable la búsqueda de diferentes esquemas de transformación entre ambos dominios, a fin de lograr su aplicación al DSP tomando en cuenta la optimización de los recursos de cómputo al momento de efectuar la convolución entre señales, mediante un reducido número de adiciones y multiplicaciones.

En sus inicios, el procesamiento digital de señales utilizó una técnica ampliamente conocida para efectuar la transformación lineal, entre las secuencias de datos discretos presentes en los dominios del tiempo y la frecuencia. Dicha técnica es la *Transformada Discreta de Fourier* (DFT). Sin embargo, y debido a lo limitado que resulta esta transformada en ciertas aplicaciones, surge entonces la necesidad de desarrollar otros esquemas de transformaciones lineales que vinculen a ambos dominios, considerando las propiedades intrínsecas de las secuencias numéricas sobre las que operan; lográndose diferentes técnicas de transformación

que van desde la *Transformada Rápida de Fourier* (FFT), hasta los sofisticados algoritmos desarrollados alrededor de las *Transformadas Discreta Coseno* (DCT) y de *Hotelling*, los cuales, por sus propiedades, son de gran utilidad en el procesamiento de imágenes digitales.

Conviene mencionar que en las secciones II, II.1, II.2, II.2.1 y II.2.2 serán tratados diferentes algoritmos destinados a soportar esquemas de transformación que están orientados a facilitar el cálculo de la transformada discreta coseno, con el propósito de establecer (mediante lo tratado en las secciones III, IV y V) cuál es la técnica que mejor permite la implantación física y lógica de la DCT bidimensional, a fin de lograr la mayor rapidez posible en el procesamiento digital de las señales provenientes de una cierta imagen digital. Una vez que se ha logrado establecer lo anterior, las secciones VI, VI.1, y VI.2 permiten esbozar las características que deben ser tomadas en cuenta al momento de diseñar una arquitectura basada en microprocesadores y que sea adecuada para implantar físicamente un sistema veloz para el cálculo de la transformada discreta coseno bidimensional, tanto para el caso directo como para el caso inverso; esto último se logra llevar al cabo durante el tratamiento de los temas que se plantean en las secciones VII, VIII y IX. En la sección X se muestran algunos comentarios y conclusiones que son producto del presente estudio. Asímismo, se anexan cinco apéndices, donde se incluyen algunos datos técnicos específicos que fueron requeridos durante la labor de diseño desarrollada en la sección VIII. De igual forma está presente una sección que incluye las principales fichas bibliográficas de los documentos escritos que sirvieron como base para la realización del presente estudio.

Por último, cabe resaltar que la arquitectura planteada y diseñada en las secciones VII y VIII, sigue un enfoque original ya que surgió a partir de una filosofía de procesamiento que usualmente se adopta tanto en supercomputadoras de alto grado de paralelismo, como en el diseño de circuitos integrados de alta eficiencia numérica. Dicha filosofía no es otra que el paralelismo llevado a la práctica mediante técnicas sistólicas; por lo tanto, al momento de utilizar dispositivos VLSI bajo esta técnica, surge de inmediato una nueva filosofía para el diseño de sistemas basados en microprocesadores.

II. TECNICAS DE TRANSFORMACION.

El Algebra Lineal, mediante la teoría de las transformaciones lineales, ha tenido un papel preponderante en lo que se refiere al procesamiento de imágenes, siendo actualmente un tópico de interés en los campos que le competen tanto en los casos de investigaciones teóricas como en los casos de trabajos de aplicación práctica. En los casos relacionados con el tratamiento de imágenes, resulta de particular interés el análisis de las transformadas discretas bidimensionales, las cuales son utilizadas para lograr ya sea el mejoramiento de imágenes, o bien la restauración, la codificación y la descripción de una cierta imagen digital [GONZ87].

En esta sección serán analizados diferentes esquemas discretos de transformación, todos ellos utilizados dentro del procesamiento de imágenes, con el propósito de establecer un marco teórico que permita contar con elementos de evaluación para determinar cuál de todos ellos resulta ser el idóneo para lograr una implantación física en un sistema electrónico digital. Algunos conceptos serán introducidos por medio de la transformada de Fourier, dando paso posteriormente al análisis de las transformadas de Walsh, de Hadamard y la Discreta Coseno vinculando ésta última con la transformada de Hotelling. Por otra parte, los algoritmos que aquí se traten, se encuentran orientados al cálculo de las transformadas discretas bidimensionales útiles en el procesamiento de imágenes digitales; siendo analizados tanto los métodos de cálculo como algunas de las propiedades intrínsecas de dichos métodos.

II.1 LA TRANSFORMADA DE FOURIER.

Una de las herramientas más comunes para efectuar el análisis espectral de la información contenida en una cierta señal es la transformada de Fourier, la cual mapea a una señal definida en el dominio del tiempo, a una señal definida en el dominio de la frecuencia; por otra parte, dicha transformada se caracteriza por ser, entre otras cosas, lineal y biyectiva lo que garantiza la existencia de la transformación inversa. En la mayoría de las aplicaciones donde interviene esta transformada, se considera únicamente el caso unidimensional; esto es, la regla de transformación actúa sobre funciones que dependen únicamente de una variable independiente, lo cual conduce a la siguiente expresión matemática, misma que es ampliamente conocida en el medio de las telecomunicaciones:

$$F(\omega) = \int_{-\infty}^{\infty} f(t) e^{-j\omega t}\, dt \qquad (1)$$

Sin embargo, en el caso del tratamiento de señales asociadas a imágenes, se tiene que éstas se encuentran definidas en un dominio que involucra a más de una variable independiente. Si se considera que lo usual es contar con un dominio espacial que depende de las coordenadas (x,y), las cuales requieren ser transformadas a un dominio frecuencial que depende de las coordenadas (u,v), entonces la expresión (1) se convierte en:

$$\mathcal{F}\{f(x,y)\}=F(u,v) = \int_{-\infty}^{\infty}\int f(x,y)e^{-j2\pi(ux+vy)}dx\,dy = R(u,v) + jI(u,v) \qquad (2)$$

donde $j = \sqrt{-1}$, y R(u,v), I(u,v) corresponden, respectivamente, a la parte real y a la parte imaginaria de F(u,v)

Por otra parte, la expresión:

$$| F(u,v) | = [R^2(u,v) + I^2(u,v)]^{1/2} \qquad (3)$$

recibe el nombre de espectro de Fourier de la función f(x,y), mientras que $| F(u,v) |^2$ se conoce como el espectro de energía, de acuerdo al teorema de Parseval.

El ángulo de fase de este tipo de transformada se obtiene mediante la expresión:

$$\Phi(u,v) = \tan^{-1}\frac{I(u,v)}{R(u,v)} \qquad (4)$$

Para poder efectuar el cálculo de la transformación inversa se cuenta con la expresión siguiente:

$$\mathcal{F}^{-1}\{F(u,v)\} = f(x,y) = \int_{-\infty}^{\infty}\int F(u,v)e^{j2\pi(ux+vy)}\,du\,dv \qquad (5)$$

En este caso, las funciones f(x,y) y F(u,v) reciben el nombre de par de transformación de Fourier y, como se puede apreciar, este par permite efectuar la transformación de funciones bidimensionales entre los dominios espacial y frecuencial.

Ahora bien, existe una técnica conocida como la *Transformada Discreta de Fourier* (DFT), que permite calcular en forma aproximada la transformada de Fourier, considerando que se aplica dicha regla de transformación a señales de naturaleza discreta, como podrían serlo las imágenes digitales. Cuando se está efectuando el tratamiento de esta clase de señales, las integrales empleadas en las ecuaciones (2) y (5) pueden ser cambiadas por sumatorias [GONZ87], quedando entoces las expresiones siguientes:

$$F(u,v) = \frac{1}{N} \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} f(x,y)\, e^{-j\,2\pi(ux+vy)/N} \qquad (6)$$

para $u,v = 0,1,2,\ldots,N-1$

$$f(x,y) = \frac{1}{N} \sum_{u=0}^{N-1} \sum_{v=0}^{N-1} F(u,v)\, e^{j2\pi(ux+vy)/N} \qquad (7)$$

para $x,y = 0,1,2,\ldots,N-1$.

Las expresiones que se enuncian en las ecuaciones (6) y (7), se conocen como el par de transformación de la transformada discreta de Fourier. Por otra parte, es aparente que, dada una cierta imagen digital, se puede calcular su transformada con sólo aplicar esas dos ecuaciones, las cuales dan la idea que únicamente deben ser efectuadas unas cuantas adiciones y multiplicaciones. Sin embargo, al momento de analizar con detalle el algoritmo de transformación expresado mediante las ecuaciones (6) y (7), se advierte que si bien no cuenta con una gran complejidad matemática, si se involucra un número bastante grande de adiciones y multiplicaciones. Es práctica común en el tratamiento de señales bidimensionales, explotar las propiedades intrínsecas de las transformadas a emplear, con el propósito de efectuar mediante transformadas unidimensionales cálculos más rápidos por el uso de algoritmos más simples.

A fin de transformar cada línea de una imagen digital mediante una transformación unidimensional, y aplicando el algoritmo de la transformada discreta de Fourier, es necesario modificar las expresiones (6) y (7), quedando las ecuaciones siguientes:

$$F(u,Y) = \frac{1}{N} \sum_{x=0}^{N-1} f(x,Y)\, e^{-j2\pi ux/N} \qquad (8)$$

para u = 0, 1, 2, ..., N-1; siendo Y una constante que representa el número de línea que se esté transformando en la imagen dada. De forma semejante, para el caso de la transformación inversa se cuenta con la expresión siguiente:

$$f(x,V) = \sum_{x=0}^{N-1} F(u,V)\, e^{j2\pi ux/N} \qquad (9)$$

para x = 0, 1, 2, ..., N-1; siendo V una constante que representa el número de la línea a la que se esté aplicando la transformación inversa.

En este caso, si se considera que una línea está compuesta por N pixeles, entonces al desarrollar el algoritmo de la DFT, se involucra la ejecución de, aproximadamente, N^2 operaciones; lo que constituye un gran consumo de recursos, principalmente de tiempo, del sistema que se esté utilizando. Con base en lo anterior, se han desarrollado diferentes algoritmos que tratan de reducir al máximo posible el número de operaciones a realizar, así como el tiempo de procesamiento empleado, aprovechando algunas de las propiedades que se presentan en ciertos agrupamientos de datos; dentro de esos algoritmos, el más difundido resulta ser el que se conoce como la *Transformada Rápida de Fourier* (FFT), mismo que puede reducir la cantidad de operaciones en la DFT convencional de aproximadamente N^2, a un total también aproximado de $N\log_2 N$. Es decir, si la imagen en tratamiento constara a manera de ejemplo de 256 pixeles, entonces la transformada discreta de Fourier convencional requeriría aproximadamente de 65,536 operaciones mientras que la misma imagen procesada bajo el algoritmo de la FFT requeriría tan sólo de 2,048 operaciones.

Como se ha mencionado con anterioridad, es factible obtener la transformada bidimensional de Fourier (2-D FFT), empleando repetidamente la transformada unidimensional de Fourier (1-D FFT), siendo ésta una técnica ampliamente conocida en el campo propio del DSP. Para poder lograr lo anterior, la argucia matemática que se emplea es transformar unidimensionalmente cada línea de la imagen a transformar, para después transponer la matriz que la representa y así, volver a transformar unidimensionalmente cada línea de la imagen en estudio, considerando que ésta fue transpuesta. A fin de ejemplificar lo anterior, considérese una cierta imagen radiológica almacenada en una matriz de orden 1024x1024; bajo la restricción de que cada pixel está compuesto, cuando más, por 16 bits. Para efectuar la transformada 2-D FFT, es necesario reservar en la memoria del sistema, una línea buffer (línea destinada para un almacenamiento auxiliar) con capacidad de 1,024 caracteres de 64 bits cada uno de ellos. En este caso, una línea con 1024 pixeles de la imagen se transfiere al buffer, según se muestra en la figura 2.1. Como se aprecia, en dicha figura se ilustra la secuencia de pasos a seguir para efectuar la transformada 2-D FFT, partiendo del método descrito mediante la ecuación (8) y la argucia matemática antes mencionada.

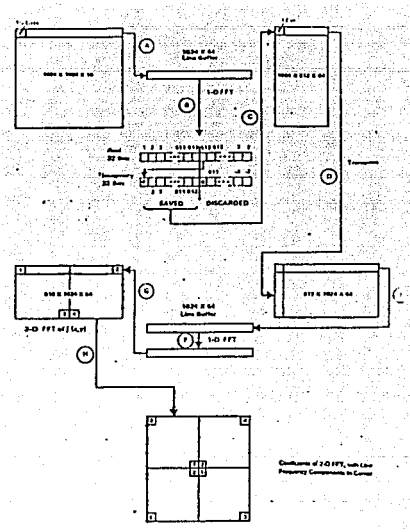


Figura 2.1 Procedimiento paso a paso para efectuar la 2D-FFT usando 1D-FFT (KUNG88).

Conviene notar que al aplicar la 1-D FFT a la línea en tratamiento, se obtiene una nueva línea que consiste de 1,024 números complejos de 64 bits cada uno de ellos, de los cuales los 32 primeros representan a la parte real, mientras que los 32 restantes corresponden a la parte imaginaria; sin embargo, únicamente se requiere almacenar la mitad del buffer en una memoria auxiliar para imágenes, lo que implica retener solamente a 512 elementos de 64 bits para un procesamiento posterior. Por otra parte, también es necesario tomar en cuenta que la función f(x,y), la cual representa a una determinada imagen, es real; así como las propiedades de simetría de la misma transformada de Fourier.

Adicionalmente a lo anterior, y como se puede apreciar en la figura 2.1, una vez que se ha efectuado la transformación de cada línea de la imagen en tratamiento, se encuentra en la memoria un banco auxiliar de 1024x512 dígitos con la información correspondiente a la transformada 1-D FFT. En ese momento se efectúa la transposición de dicha matriz numérica, a fin de aplicarle a la transpuesta el mismo algoritmo de transformación; sin embargo, en este último paso se obtendrá como resultado final la transformada 2-D FFT de la imagen original.

Cabe resaltar una vez más, que el procedimiento aquí descrito considera la aplicación del método de la FFT unidimensional, el cual ha sido ampliamente estudiado en diferentes textos relacionados con el DSP.

Por otra parte, si bien al aplicar el algoritmo de la FFT se reduce de manera notable el número de operaciones para efectuar la DFT de una cierta imagen digital, áún se mantiene con la FFT un nivel de complejidad similar al de la DFT, redundando esto en un

desaprovechamiento de los recursos de cómputo en el sistema que se esté usando.

Un aspecto importante que rara vez se toma en cuenta al aplicar la FFT, se encuentra en el hecho de que, en la mayoría de los casos, se actúa sobre datos cuya naturaleza es real; mientras que las ecuaciones (8) y (9) se encuentran definidas considerando el caso más general donde se transforman datos complejos. Sin que lo anterior constituya una restricción, un algoritmo basado en la FFT, con la mitad de la dimensión necesaria para calcular la DFT de una secuencia real, requiere un número substancialmente menor de operaciones con respecto a un método empleado para calcular una FFT simple, aplicada simultáneamente a dos secuencias reales. Este último método tiene como desventajas requerir efectuar dos transformadas del tipo DFT, al mismo instante, además de que el ordenamiento de salida utiliza mayor cantidad de adiciones. Por otra parte, el hecho de que tanto la entrada como la salida sean secuencias de números reales, se emplea explícitamente en un algoritmo de convolución real, donde las transformadas DFT (Directa e inversa), se calculan mediante una sola FFT compleja [VETT84].

Al considerar lo anterior, y cuando se trata de aplicar una transformación a datos reales surge, entre otras muchas, la transformada discreta coseno, comúnmente denotada como DCT, la cual presenta ciertas características que han hecho su uso muy popular en el tratamiento de imágenes. A partir de la introducción de la DCT, la búsqueda de algoritmos rápidos de transformación ha seguido dos tendencias principales, siendo una de ellas el cálculo de la DCT mediante una FFT de la misma dimensión, existiendo la limitante de tomar dos transformadas simultáneamente; mientras que la otra tendencia corresponde a una aproximación más directa, considerando algoritmos mejor involucrados con su aplicación. Conviene aclarar que la técnica formal se desempeña mejor que todas las técnicas posteriores, si se considera la aplicación de la FFT óptima, lo cual es un hecho que habitualmente permanece obscuro [VETT84].

En fechas recientes, la evaluación de los algoritmos destinados al procesamiento digital de señales, ha dejado un poco al margen la contabilización exclusiva de las multiplicaciones a efectuar, para tomar en cuenta, principalmente, el número total de operaciones a realizar, incluyendo las transferencias de datos. Esto se debe a que los cocientes que se definen como (tiempo de multiplicación)/(tiempo de adición) y (tiempo de multiplicación)/(tiempo de carga) son cercanos a la unidad, en la mayoría de las computadoras y procesadores destinados al DSP. Otro aspecto que actualmente está tomando gran importancia se refiere a la generación de software eficiente en tiempo, considerando que la eficiencia de un algoritmo se basa en una combinación no trivial, tanto del conteo de operaciones como de la complejidad de su estructura [VETT84].

Con base en lo expuesto hasta aquí, se puede afirmar que conviene establecer un marco teórico que permita entender las propiedades más generales de las transformadas lineales del tipo discreto, considerando que algunas de ellas fueron introducidas mediante el análisis de la transformada de Fourier; sin embargo, es necesario refinarlas antes de intentar una realización del algoritmo de la *Transformada Discreta Coseno*. En las próximas secciones serán expuestas algunas de esas propiedades, haciendo énfasis en las propiedades de simetría y separabilidad de una determinada transformada.

II.2 TRANSFORMADAS SEPARABLES.

En la sección anterior han sido introducidos ciertos conceptos relacionados con las transformadas discretas, mediante los diferentes algoritmos conocidos para obtener la transformada de Fourier; sin embargo, ese esquema de transformación puede no ser el más conveniente para resolver ciertos casos de aplicación. Por tal motivo, resulta importante conocer algunos otros métodos de transformación, así como sus principales propiedades, con el propósito de determinar cuál de ellos es el mejor, a fin de que dicho método pueda ser llevado al caso de una implantación detallada. Para lograr esto último, se hace necesario establecer un marco teórico más general, lo que se logra a partir de las consideraciones que a continuación serán descritas.

Como se desprende de la ecuación (9), la transformada discreta de Fourier puede ser expresada en términos de la siguiente relación general [GONZ87]:

$$T(u) = \sum_{x=0}^{N-1} f(x)g(x,u) \qquad (10)$$

donde T(u) corresponde a la transformada de f(x), mientras que g(x,u) se conoce como el kernel de la transformación directa; siendo u una variable que toma los valores de 0, 1,..., N-1. De manera semejante, la transformación inversa está dada por la expresión:

$$f(x) = \sum_{u=0}^{N-1} T(u)h(x,u) \qquad (11)$$

donde h(x,u) corresponde al kernel de la transformación inversa mientras que x es una variable que toma los valores de 0, 1, ..., N-1 [GONZ87].

Es importante mencionar que el kernel de una transformación, según la acepción que aquí se plantea, corresponde al conjunto de reglas de correspondencia que definen a la transformada, ya sea directa o inversa, quedando determinada su naturaleza por las propiedades con que cuente dicho kernel.

En el caso de transformaciones bidimensionales, las transformadas directa e inversa están dadas por las expresiones [GONZ87]:

$$T(u,v) = \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} f(x,y)g(x,y,u,v) \qquad (12)$$

$$f(x,y) = \sum_{u=0}^{N-1} \sum_{v=0}^{N-1} T(u,v)h(x,y,u,v) \qquad (13)$$

donde, como antes, $g(x,y,u,v)$ y $h(x,y,u,v)$ reciben los nombres de kernel de la transformación directa e inversa respectivamente.

Cuando se tiene el caso de transformadas bidimensionales, se dice que un kernel es separable siempre que:

$$g(x,y,u,v) = g_1(x,u)g_2(y,v) \qquad (14)$$

Adicionalmente a esto, el kernel es simétrico si g_1 es funcionalmente igual a g_2. En ese caso, la expresión (14) puede ser expresada en la forma [GONZ87]:

$$g(x,y,u,v) = g_1(x,u)g_1(y,v) \qquad (15)$$

Es interesante observar que estas dos últimas propiedades de los kernels, son importantes en cuanto a que facilitan notablemente el desarrollo de métodos para el cálculo rápido de una transformada permitiéndose, de esta forma, lograr ágiles esquemas de transformación útiles en situaciones de aplicación cuyo ritmo de variación es muy alto.

Por otra parte, es fácil deducir que la transformada bidimensional es un caso especial de la ecuación (12), donde el kernel está dado por la expresión siguiente [GONZ87]:

$$g(x,y,u,v) = \frac{1}{N} e^{-j2\pi(ux+vy)/N} \qquad (16)$$

el cual es separable y simétrico, puesto que se cumple:

$$g(x,y,u,v) = g_1(x,u)g_1(y,v) \qquad (17)$$

$$g(x,y,u,v) = \frac{1}{\sqrt{N}} e^{-j2\pi ux/N} \cdot \frac{1}{\sqrt{N}} e^{-j2\pi vy/N} \qquad (18)$$

Adicionalmente a esto último, se puede mostrar que el kernel de la transformada inversa bidimensional de Fourier es también separable y simétrico.

Resulta de gran importancia mencionar, para los fines que persigue este estudio, que cuando una transformada bidimensional posee un kernel separable, entonces ésta puede ser calculada en dos pasos, cada uno de ellos requiriendo de una transformada unidimensional. Inicialmente se obtiene una transformada unidimensional a lo largo de cada renglón de $f(x,y)$, lo que da origen a [GONZ87]:

$$T(x,v) = \sum_{y=0}^{N-1} f(x,y)g_2(y,v) \qquad (19)$$

donde $x,v = 0, 1, 2, \ldots, N-1$. Posteriormente, se obtiene la siguiente transformada unidimensional a lo largo de cada columna de $T(x,v)$; lo que conduce a la expresión siguiente:

$$T(u,v) = \sum_{x=0}^{N-1} T(x,v)g_1(x,u) \qquad (20)$$

para $u,v = 0, 1, 2, \ldots, N-1$. Como se puede notar, éste procedimiento coincide con aquel descrito en la sección II.1, destinada a la obtención de la transformada rápida de Fourier para el caso bidimensional. Cabe mencionar que se puede llegar a los mismos resultados finales si la transformada unidimensional se toma primero respecto a las columnas de $f(x,y)$, obteniéndose $T(y,u)$, para entoces transformar cada uno de los renglones de ésta última para llegar finalmente a la obtención de $T(u,v)$ [GONZ87].

Por otra parte, una propiedad importante en los kernels separables, es que estos también pueden ser expresados en forma matricial mediante la siguiente expresión:

$$T = AFA \qquad (21)$$

donde F es la matriz de orden NxN, asociada a la imagen en tratamiento; A es la matriz de orden NxN asociada a la transformación, siendo simétrica y cuyos elementos son $a_{ij} = g_i(i,j)$; T es la matriz de orden NxN que contiene los resultados obtenidos mediante la transformada en uso, para los valores de u,v en el rango de 0, 1, 2, ..., N-1 [GONZ87].

En lo que respecta a la obtención de la transformada inversa, es necesario premultiplicar y postmultiplicar a la ecuación (21) por la matriz B asociada a la inversa, lo cual conduce a la expresión:

$$BTB = BAFAB \qquad (22)$$

Si $B = A^{-1}$, entonces:

$$F = BTB \qquad (23)$$

lo cual indica que una cierta imagen digital puede ser completamente recuperada a partir de su transformada. Cabe mencionar que si B no es estrictamente igual a A^{-1}, entonces la expresión (22) permite lograr sólo una aproximación de F, dada por la relación:

$$\hat{F} = BAFAB \qquad (24)$$

En relación con esto último, es importante aclarar que si la transformada cuenta con otras propiedades adicionales a las de simetría, entonces las ecuaciones (21) y (23) pueden ser modificadas de manera que se facilite aún más su cálculo. Tal es la condición de las transformadas ortogonales donde $A^{-1} = A^{T}$, lo cual permite establecer ecuaciones de similaridad mediante (21) y (23); aprovechándose, en ese caso, la existencia de los valores y vectores característicos con que cuentan ciertas transformaciones, como lo acostumbran hacer los métodos relacionados y que se aproximan a la transformada de Hotelling. Al aplicar la propiedad de separabilidad en el kernel de una transformada, la reducción en el número de operaciones a realizar es equivalente al que se logra mediante el algoritmo de la FFT, siendo del orden de $N\log_2 N$ operaciones de multiplicación y suma por cada renglón o columna de una imagen matricial de orden NxN [GONZ87].

II.2.1 LAS TRANSFORMADAS DE WALSH Y HADAMARD.

Un esquema de transformación muy conocido en el campo del procesamiento digital de señales es, sin lugar a dudas, el que corresponde a la transformada de Walsh de una función f(x), la cual se denota como W(u) y que se obtiene, cuando $N = 2^n$, mediante el kernel:

$$g(x,u) = \frac{1}{N} \prod_{i=0}^{n-1} (-1)^{b_i(x)b_{n-1-i}(u)} \qquad (25)$$

el cual, al ser substituído en la ecuación (10), conduce a la expresión:

$$W(u) = \frac{1}{N} \sum_{x=0}^{N-1} f(x) \prod_{i=0}^{n-1} (-1)^{b_i(x)b_{n-1-i}(u)} \qquad (26)$$

donde $b_k(z)$ corresponde al k-ésimo bit en la representación binaria de z. Después de analizar la ecuación (25), se llega a la conclusión de que el arreglo formado por el kernel de la transformada de Walsh forma una matriz simétrica cuyos renglones y columnas son ortogonales. Esta propiedad, la cual es general e independiente del orden, permite que el kernel de la transformación inversa sea idéntico al de la directa, excepto por una constante multiplicativa igual a $1/N$; esto es:

$$h(x,u) = \prod_{i=0}^{n-1} (-1)^{b_i(x)b_{n-1-i}(u)} \qquad (27)$$

Por lo tanto, la transformada inversa de Walsh está dada por la expresión:

$$f(x) = \sum_{u=0}^{N-1} W(u) \prod_{i=0}^{n-1} (-1)^{b_i(x)b_{n-1-i}(u)}. \qquad (28)$$

Como se puede apreciar en esta última ecuación, y a diferencia de la transformada de Fourier la cual se basa en términos trigonométricos, la transformada de Walsh consiste en una expansión en series de funciones básicas, mismas que toman los valores de +1 ó -1 [GONZ87].

De lo anterior se desprende que las transformadas de Walsh, tanto la directa como la inversa, difieren únicamente por el término 1/N. Por lo tanto, cualquier algoritmo que se utilice para calcular la transformada directa, puede tambien ser empleado para obtener la inversa simplemente al multiplicar el resultado obtenido, por el factor N. Adicionalmente a esto, debe tomarse en cuenta que las transformadas de Walsh son reales por lo que, en comparación con la transformada de Fourier, requieren menor cantidad de memoria del sistema de cómputo en uso.

En lo que respecta al caso bidimensional para las transformadas de Walsh, ambos kernels son separables y simétricos; motivo por el cual tanto la transformada directa como la inversa, pueden ser obtenidas a partir de aplicar reiteradamente los esquemas de transformación unidimensionales.

Si bien, la transformada de Walsh tiene ciertas ventajas con respecto a la transformada de Fourier, existe otro esquema de transformación que es muy similar al de Walsh, conocido como la transformada de Hadamard, y cuyo kernel para el caso directo está dado por [GONZ87]:

$$g(x,u) = \frac{1}{N}(-1)^{\sum_{i=0}^{n-1} b_i(x)b_i(u)} \qquad (29)$$

donde la sumatoria del exponente se realiza en aritmética de módulo 2, y como en la ecuación (26), $b_k(z)$ corresponde al k-ésimo bit de la representación binaria de z. Al sustituir la ecuación (29) en la ecuación general (10), se llega a la siguiente expresión para la transformada de Hadamard unidimensional [GONZ87]:

$$H(u) = \frac{1}{N}\sum_{x=0}^{N-1} f(x)(-1)^{\sum_{i=0}^{n-1} b_i(x)b_i(u)} \qquad (30)$$

donde $N = 2^n$, mientras que u toma valores en el rango de 0,1,2,...,N-1.

Por otra parte, el kernel de la transformada de Hadamard, como en el caso de la transformada de Walsh, forma una matriz cuyos renglones y columnas son ortogonales. Esta condición nuevamente conduce a que el kernel de la transformada inversa, excepto por el término 1/N, es igual al kernel de la transformada directa; esto es [GONZ87]:

$$h(x,u) = (-1)^{\sum_{i=0}^{n-1} b_i(x) b_i(u)} \qquad (31)$$

que al sustituir este kernel en la ecuación (11), da lugar a la siguiente expresión para la transformada inversa de Hadamard [GONZ87]:

$$f(x) = \sum_{u=0}^{N-1} H(u) (-1)^{\sum_{i=0}^{n-1} b_i(x) b_i(u)} \qquad (32)$$

para $x = 0, 1, 2, \ldots, N-1$.

Para el caso bidimensional, los kernels de las transformadas directa e inversa cuentan con las propiedades de simetría y separabilidad, por lo cual la obtención de dichas transformadas se logra después de aplicar repetidamente los algoritmos que efectúan el cálculo de la transformada de Hadamard unidimensional, descritos mediante las ecuaciones (30) y (32).

Ahora bien, si se comparan las matrices generadas por los kernels respectivos de las transformadas de Walsh y de Hadamard, se llega a la conclusión de que son prácticamente los mismos, diferenciándose tan sólo en el orden en que se encuentran ciertas columnas y renglones. De hecho, cuando $N = 2^n$, ésta es la única diferencia que existe entre esas dos transformadas. Sin embargo, cuando N difiere de ser una potencia entera de 2, la discrepancia entre ambos esquemas es más relevante en cuanto a que mientras la transformada de Walsh puede ser formulada para cualquier potencia entera de N, la existencia de la transformada de Hadamard para valores de N distintos a los de una potencia entera de 2, se ha comprobado únicamente hasta el valor N = 200 [GONZ87].

Con fundamento en lo anterior, y si se considera que la mayoría de las aplicaciones de transformaciones en el procesamiento de imágenes se basan en el tratamiento de $N = 2^n$ muestras por renglón o columna, entonces el uso de las transformadas de Walsh y de Hadamard se entremezcla para dar lugar en la terminología cotidiana de ese campo del DSP, al término de la transformada de Walsh-Hadamard, el cual sirve para denotar a cualquiera de los dos esquemas de transformación aquí descritos. Cabe mencionar que esos dos esquemas son de gran utilidad en el procesamiento de imágenes, toda vez que son requeridos métodos de transformación que consideren igualmente importantes las transformadas directas e inversas, al margen de tomar en cuenta la rapidez intrínseca que acarrea consigo el efectuar operaciones entre matrices cuyos únicos elementos son los números +1 y -1.

Por otra parte, una ventaja adicional en el algoritmo de Hadamard, se presenta al momento de la generación recursiva de las matrices asociadas a dicha transformación, en cuanto a que las matrices de órdenes superiores se pueden obtener a partir de las matrices de órdenes inferiores. Esto es, si la matriz de Hadamard con el menor de todos los órdenes (N=2) está dada por:

$$H_2 = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \qquad (33)$$

entonces, si se considera que H_N representa a la matriz de orden N, la relación recursiva está dada por la expresión [GONZ87]:

$$H_{2N} = \begin{bmatrix} H_N & H_N \\ H_N & -H_N \end{bmatrix} \qquad (34)$$

donde H_{2N} corresponde a una matriz de Hadamard de orden 2N, asumiendo que $N = 2^n$.

II.2.2 LA TRANSFORMADA DISCRETA COSENO.

Como se ha mencionado, desde su aparición la *Transformada Discreta Coseno* (DCT), ha encontrado gran cantidad de aplicaciones en lo que se refiere al procesamiento de imágenes y señales digitales, debiéndose esto, principalmente, al hecho de que la DCT se comporta de manera muy similar a la Transformada de Karhunen-Loève (KLT), misma que también es conocida como la Transformada de Hotelling, para datos aleatorios y Markovianos de primer orden [HSIE87]. La transformada KLT se distingue por ser el algoritmo óptimo de transformación para imágenes; sin embargo, no existe un método computacional lo suficientemente rápido para calcularla puesto que no se trata de una transformada separable. Por otra parte, la DCT se desempeña mejor que otras transformadas ortogonales (como la DFT, la DWHT, la DST, etc.), al momento de considerar diferentes criterios importantes tales como la tasa de distorsión, la decorrelación de datos, la energía de compresión, etc. [DEFI89].

Con base en lo anterior, recientemente se han desarrollado diferentes métodos algoritmicos que permiten el cálculo rápido de la DCT a partir de las propiedades inherentes con que cuénta dicha transformada. Esos algoritmos pueden ser clasificados básicamente en cualquiera de las categorías siguientes, dependiendo de su método de aproximación [HSIE87]:

a) Cálculo indirecto.

b) Factorización directa.

c) Cálculo recursivo.

En el caso (a) se emplean las transformadas de Fourier y de Walsh-Hadamard a fin de obtener la DCT por métodos indirectos, sin embargo, frecuentemente se efectúan operaciones innecesarias en diversos pasos de cómputo. Al momento de utilizar factorizaciones con matrices esparcidas, los algoritmos que caen en el caso (b) ganan rapidez en comparación con otros algoritmos puesto que una matriz unitaria, teóricamente, siempre puede ser factorizada como el producto de matrices relativamente esparcidas. Los algoritmos que cuentan con una factorización directa han sido utilizados para las matrices asociadas a la DCT, requiriéndose en esos casos una pequeña cantidad de multiplicaciones o adiciones. Por otra parte, el enfoque recursivo que se intenta en el caso (c), pretende generar matrices asociadas a la DCT, de órdenes mayores a partir de matrices de órdenes inferiores.

Cabe mencionar que si bien todos los esfuerzos que se han desarrollado para la obtención de algoritmos rápidos de la DCT, tienden a reducir el tiempo de procesamiento en un contexto basado en sistemas con microprocesadores; como consecuencia, los criterios de optimización se han orientado a la reducción del número de multiplicaciones por efectuar. Sin embargo, al momento de considerar una implantación en VLSI de alguno de esos algoritmos para el cálculo rápido de la DCT, esta clase de reducciones deja de ser una imperiosa necesidad [DEFI88].

Ahora bien, si se considera la factibilidad de obtener la transformada bidimensional DCT, a partir de varias transformadas unidimensionales puesto que el kernel de la 2-D DCT es simétrico y separable [GONZ87], entonces el problema de calcular una transformada DCT bidimensional de NxN puntos, se reduce a calcular 2N transformadas unidimensionales de N puntos [DEFI89]. Por lo tanto, y a menos que se especifique otra cosa, nos abocaremos únicamente a tratar el caso de transformadas DCT unidimensionales (1-D DCT).

Desde el punto de vista del Algebra Lineal, conviene precisar que la Transformada Discreta Coseno es un operador lineal ortogonal, que puede representarse matricialmente a partir de una base de vectores, los cuales son funciones cosenoidales muestreadas de acuerdo a un cierto orden. Una matriz asociada a la DCT, referida a una base normalizada de N vectores, se obtiene mediante la expresión [MING89]:

$$c_{k,l} = \sqrt{2/N} \cos \frac{([2k-1)(l-1)]}{2N} \qquad (35)$$

para $k = 1, 2, \ldots, N$; $l = 2, 3, \ldots, N$ y $c_{k,l} = N^{-1/2}$ cuando $l = 1$.

A partir de esto último, se puede afirmar que la transformada bidimensional DCT, de tamaño NxN, está definida de acuerdo a la ecuación siguiente:

$$Z = C^T X C \qquad (36)$$

donde C^T es la transpuesta de la matriz C asociada a la DCT, mientras que X es la matriz de datos y Z es la matriz de imágenes de X bajo los efectos de la DCT.

Es interesante notar que una implantación directa de la ecuación (36) consume intensamente los recursos de cómputo de cualquier sistema, debiéndose esto a la gran cantidad de operaciones que requieren los productos matriciales. A manera de ejemplo, considérese que se asume una tasa de muestreo de 14.3 MHz y una N=16, entonces para estas condiciones se requiere efectuar aproximadamente medio billón de multiplicaciones y adiciones por segundo. A fin de reducir esa cantidad de operaciones, la realización del algoritmo de la DCT de orden NxN, de acuerdo a la definición dada en (36), se logra utilizando diferentes formas de esquemas o diagramas de mariposa (Estos esquemas llegan a recibir el nombre de mariposeo) los cuales consideran la menor cantidad posible de multiplicadores. Además, es posible afirmar que diferentes aproximaciones propuestas para la implantación de la DCT, han enfatizado el hecho de reducir el número requerido de multiplicadores; sin embargo, todavía siguen siendo necesarias grandes cantidades de estos elementos con el propósito de alcanzar una tasa adecuada de trabajo útil por unidad de tiempo. Más aún, utilizar una aproximación por mariposeo frecuentemente conduce a una arquitectura irregular con un ruteo complicado, lo que afecta directamente al diseño de un sistema orientado a soportar la DCT. Con fundamento en lo anterior, se puede afirmar que el uso de variadas etapas de multiplicación, acompañándose esto por los errores de redondeo y truncamiento en una aritmética de precisión finita, pueden causar que se degrade severamente la precisión interna y general del sistema [MING89].

En lo que respecta a los diferentes algoritmos que han sido desarrollados para el cálculo rápido de la DCT, se puede afirmar que el de Chen fue uno de los primeros en aparecer, y por sus características ha servido como referencia de comparación para medir la eficiencia de algoritmos más recientes. Por su parte, el algoritmo de Lee se considera entre los más veloces para efectuar la transformada; sin embargo, este algoritmo requiere de la inversión o de la división de coeficientes cosenoidales, lo que acarrea consigo ciertas inestabilidades numéricas debido a la presencia de errores de redondeo al utilizar registros de longitud finita [HSIE87]. El algoritmo de Vetterli calcula la DCT con base en métodos indirectos, en cuanto a que se obtiene la transformada coseno a partir de la FFT. Al momento de proponer los métodos de factorización Wang, Suehiro y Hatori demostraron haber obtenido un algoritmo rápido similar al de Wang [HSIE87]. El algoritmo de Hou, que parte del mismo modelo que el de Lee, no requiere de la inversión de factores cosenoidales lo que lo hace estable y rápido; además, se puede considerar como una generalización del algoritmo de Cooley-Tukey para el cálculo de la FFT, siendo recursivo y tomando en cuenta algunas de las características de las transformadas de Walsh y Hadamard [HSIE87].

Con el propósito de poder comparar objetivamente los algoritmos comentados en el párrafo anterior, las tablas I y II muestran datos referentes al total de multiplicaciones y adiciones que deben ser realizadas en cada uno de esos métodos de cálculo.

TABLA I

CANTIDAD DE MULTIPLICACIONES.

N	Chen	Wang	Lee	Vetterli	Suehiro	Hou
4	6	5	4		4	4
8	16	13	12	12	12	12
16	44	35	32	32	32	32
32	116	91	80	80	80	80
64	292	227	192	192	192	192

TABLA II

CANTIDAD DE ADICIONES

N	Chen	Wang	Lee	Vetterli	Suehiro	Hou
4	8	9	9		9	9
8	26	29	29	29	29	29
16	74	83	81	81	81	81
32	194	219	209	209	209	209
64	482	547	513	513	513	513

A partir de lo que hasta aquí se ha expuesto, y considerando las posibilidades de los dispositivos VLSI, cabe señalar que ha surgido una prometedora versión del algoritmo de la DCT, ideal para ciertas implantaciones físicas. Tal versión se denomina *Transformada Discreta Coseno Modificada Simétrica* (Modified Symetric DCT) y es conocida simplemente como MSDCT, siendo una variante que ofrece algunas ventajas sobre el algoritmo de la *DCT Simétrica* (SDCT). Conviene mencionar que ambas transformadas, tanto la SDCT como la MSDCT, presentan la característica común de contar cada una de ellas con las mismas matrices asociadas para las transformadas directa e inversa. Esto tiene como consecuencia importante el poder diseñar una misma arquitectura digital que pueda soportar, indistintamente, la obtención de las transformadas directa e inversa en cada algoritmo.

Por otra parte, de acuerdo al algoritmo MSDCT, dada una secuencia de N datos $\{x(n)\} = \{x(0), x(1), \ldots, x(N-1)\}$, entonces las transformadas directa e inversa de la DCT tienen la siguiente forma [DEFI89]:

$$X(k) = \sqrt{\frac{2}{N-1}} \sum_{n=0}^{N-1} c_n x(n) \cos\left(\frac{nk\pi}{N-1}\right) \quad ; \; k = 0, 1, 2, \ldots, N-1. \qquad (37)$$

$$x(n) = \sqrt{\frac{2}{N-1}} \sum_{n=0}^{N-1} c_n X(k) \cos\left(\frac{nk\pi}{N-1}\right) \quad ; \; n = 0, 1, 2, \ldots, N-1. \qquad (38)$$

$$\text{donde } c_i = \begin{cases} 1/2 & \text{si } i=0 \text{ ó } i=N-1 \\ 1 & \text{en caso contrario.} \end{cases}$$

A manera de ejemplo, considérese que se tiene cierta secuencia de datos a transformar, constituida por 16 puntos diferentes; lo que se traduce en fijar el valor de N como N = 16. A partir de lo anterior, y de acuerdo a las expresiones (37) y (38), se llega a que las matrices asociadas a las transformaciones DCT, directa e inversa, son idénticas entre sí e iguales a la matriz que se muestra en la figura 2.2.

Como se puede apreciar en la figura 2.2, la matriz asociada a la DCT, obtenida a partir del algoritmo modificado, cuenta con algunas propiedades de simetría, semejantes a aquellas que se presentaban en las transformadas de Walsh-Hadamard y como es fácil verificar, los renglones de la MSDCT son, alternadamente, simétricos o antisimétricos alrededor de la columna central [DEFI89].

Figura 2.2 Matriz asociada a la DCT bajo el esquema MSDCT.

Se debe recordar que para efectuar la transformada 2-D DCT sobre una matriz de datos, siendo ésta de orden 16x16, es necesario utilizar repetidamente la matriz que se muestra en la figura 2.2; puesto que la transformada bidimensional DCT se puede calcular a partir de varias transformadas unidimensionales, de acuerdo a la propiedad de la separabilidad del kernel de la DCT como ya se mencionó previamente.

Por otra parte, a fin de lograr un diseño eficiente y regular del sistema electrónico a emplear, conviene utilizar una arquitectura concurrente que incorpore los principios y beneficios de la aritmética distribuida (la cual será tratada con más detalle adelante); así como una estructura orientada al buen manejo de la memoria, lo que permite una implantación electrónica eficiente de la 2-D DCT, misma que debe contar con gran rapidez y gran precisión. Los principales méritos a buscar en este tipo de arquitectura son:

i) Un esquema electrónico más eficiente, debido al reemplazo de los multiplicadores por tablas de asociación de elementos (Look-up Tables).

ii) La expectativa de lograr resultados con una gran exactitud, empleando la misma precisión interna; debiéndose esto a que los resultados acumulados pasan por menor número de etapas de redondeo y truncamiento que otras arquitecturas.

iii) Una estructura regular, lo que permite un diseño modular.

iv) Un ahorro significativo en el área a utilizar, así como el uso de operaciones de gran rapidez, resultando esto como consecuencia de combinar las ventajas de estructuras del tipo serie y paralelo.

Lo anterior conduce a pensar que las características de esta arquitectura, permiten un diseño de gran rendimiento, máxime si se considera que este tipo de procesadores se basa en memorias, sumadores y registros de diferentes tipos; omitiéndose, por tanto, el uso de multiplicadores [MING89].

Después de analizar lo que se ha expuesto hasta aquí, se deduce que un elemento fundamental de cualquier arquitectura susceptible de ser implantada físicamente es, sin lugar a dudas, la aritmética distribuida; razón por la cual conviene precisar a continuación ciertas peculiaridades y conceptos relacionados con ese tipo de aritmética.

III. LA ARITMETICA DISTRIBUIDA.

La técnica denominada como *Aritmética Distribuida*, y que comunmente se conoce como DA, recibe ese nombre debido a que las operaciones aritméticas que aparecen, tales como adiciones y multiplicaciones, no se encuentran agrupadas de acuerdo a una cierta forma que sea familiar y cómoda; sino, por el contrario, están distribuidas de manera irreconocible en la mayoría de los casos [WHIT89]. Por otra parte, la operación que más frecuentemente se encuentra en este tipo de aritmética es la suma de productos; lo que constituye la base del cálculo de un cierto producto interno, siendo ésta una operación esencial de las transformadas discretas. Cabe agregar a lo anterior, que esa clase de cómputos es la que se realiza con mayor eficiencia en la técnica de DA.

De acuerdo a lo expuesto, conviene aclarar que esta clase de aritmética, desde el punto de vista computacional, es altamente eficiente y sus ventajas se explotan en el diseño de circuitos digitales, principalmente en los del tipo VLSI. Sin embargo, en el caso de utilizar circuitos integrados en forma discreta, es difícil lograr una adecuada implantación de la Aritmética Distribuída como tal.

Por otra parte, la técnica de la aritmética distribuida consiste básicamente en una operación de cómputo, en el formato serie, del producto interno entre dos vectores; en particular mediante esta técnica se busca efectuar el cálculo del producto escalar ordinario. Adicionalmente una de las ventajas de utilizar la DA, estriba en el hecho de que ésta es fácilmente mecanizable; sin embargo, una de las desventajas que frecuentemente se argumenta en su contra, es su aparente lentitud, debida a su naturaleza serie. Sin embargo, esa desventaja no es real, si se compara el número de componentes de un vector contra el número de bits de cada una de las componentes, por ejemplo, el tiempo que se requiere para introducir a un cierto sistema ocho datos, de ocho bits cada uno de ellos, mediante un formato en paralelo es exactamente el mismo tiempo que se requiere para introducir, simultáneamente mediante ocho líneas diferentes, todos los ocho datos en formato serie [WHIT89].

A fin de ilustrar la generación directa de un producto interno, mediante DA, considérese la suma de productos siguiente:

$$y = \bar{a}^T \cdot \bar{x} = \sum_{i=0}^{N-1} a_i x_i \qquad (38)$$

donde el vector $\bar{x}$ representa a los datos de entrada y el vector $\bar{a}^T$ contiene ciertos coeficientes predeterminados. Si se toma en cuenta que cada dato de entrada pude ser representado mediante su equivalente en complemento a dos, el cual, por conveniencia y no por necesidad, puede ser escalado de tal manera que $|x_i| < 1$, por lo tanto, cada x_i puede expresarse como:

$$x_i = \sum_{j=1}^{Wd-1} x_{ij}.2^{-j} - x_{io} \qquad (39)$$

En la expresión (39), el bit x_{io} corresponde al bit de signo, mientras los bits x_{ij} pueden tener, indistintamente, los valores de 0 ó 1 lógicos; se considera que el bit $x_{i,Wd-1}$ corresponde al bit menos significativo (LSB) del dato en tratamiento. Por otra parte, Wd representa la longitud de cada palabra o dato.

A partir de combinar las expresiones (38) y (39), se llega entonces a la siguiente igualdad:

$$y = \sum_{i=0}^{N-1} a_i \left[\sum_{j=1}^{Wd-1} x_{ij}.2^{-j} - x_{io} \right] \qquad (40)$$

Por otra parte, y con fundamento en las propiedades de la adición, el orden de las sumatorias de la expresión (40), puede ser intercambiado, a fin de realizar primero la sumatoria de los términos producto del j-ésimo bit de x_i por a_n, efectuando posteriormente sobre todos los bits restantes. Es necesario realizar una substracción final para el bit de signo, lo que da lugar a la expresión siguiente [DEFI89]:

$$y = \sum_{j=1}^{Wd-1} \left[\sum_{i=0}^{N-1} a_i x_{ij} \right] 2^{-j} - \sum_{i=0}^{N-1} a_i x_{io} \qquad (41)$$

Conviene aclarar que si bien las expresiones (40) y (41) básicamente llegan al mismo resultado numérico, la expresión (40) define la forma convencional de calcular el producto interno, mediante cómputos aritméticos agrupados; mientras que la expresión (41) es el paso crucial que permite definir los cómputos numéricos mediante el principio de la aritmética distribuida [WHIT89]. Una diferencia importante entre las expresiones (40) y (41) se encuentra en que en la primera de ellas las operaciones se efectúan por palabras, mientras que la última expresión se aplica por cada bit de la palabra en uso.

Con la finalidad de simplificar la expresión (41), se define F_j como una función del j-ésimo bit de cada x_n, de acuerdo a la expresión:

$$F_j = a_0x_{0j} + a_1x_{1j} + a_2x_{2j} + \ldots + a_{(N-1)}x_{(N-1)j} \qquad (42)$$

Por lo tanto, las sumatorias de la expresión (41) pueden ser escritas como:

$$y = \sum_{j=1}^{Wd-1} F_j \cdot 2^{-j} - F_0 \qquad (43)$$

Puesto que cada bit sólo puede tomar los valores de 0 y 1 lógicos, Fj puede tomar únicamente 2^N diferentes valores, los cuales pueden ser precalculados y almacenados en una tabla de asociación de datos (look-up table), misma que puede encontrarse grabada en memoria tipo ROM. En este caso, los j-ésimos bits de x0 a xN-1 se utilizan para direccionar a esa tabla de datos. Lo que se conoce como la forma de Horner puede ser utilizada para efectuar los cálculos de la expresión (43), quedando como:

$$y = (...((0 + F_{Wd-1})2^{-1} + F_{Wd-2})2^{-1} + ... + F_1)2^{-1} - F_0 \qquad (44)$$

La circuitería resultante es muy simple puesto que, para implantar el producto interno, se requiere únicamente de un sumador/substractor con un registro acumulador (ALU), además de una tabla en memoria ROM [DEFI89]. La estructura de dicha circuitería se muestra en la figura 3.1, misma que se conoce con el nombre de ***Procesador de Aritmética Distribuida***.

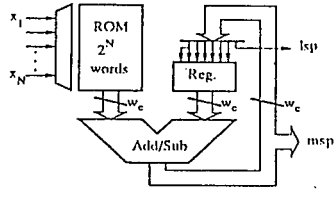


Figura 3.1 Procesador de aritmética distribuida.

Respecto a la unidad ALU, conviene aclarar que no es necesario diseñarla de manera que procese exactamente Wd bits al mismo tiempo; sino, de hecho, únicamente se deben de procesar Wc bits; donde Wc es un coeficiente que representa al ancho de la palabra de la memoria ROM en uso. La consideración anterior es importante puesto que al hacerla, se logran significativos ahorros en la electrónica, debido esto a que en la mayoría de las aplicaciones se tiene que Wc < Wd. En este caso, la división entre dos que es necesario efectuar en el algoritmo a usar, se realiza mediante un corrimiento de la entrada del registro acumulador. Es factible aplicar el procedimiento anterior en las entradas, mismas que van desde x1 hasta xN, donde se lleva al cabo un corrimiento iniciando por el bit menos significativo. Al efectuar lo anterior, se genera una dirección de la memoria ROM en la cual se encuentran

almacenados los coeficientes Fj. El factor Fj se suma al contenido del acumulador, a fin de que el resultado sea almacenando nuevamente en dicho registro después de efectuar un corrimiento, mismo que, por ejemplo, puede consistir en una división por dos. Este procedimiento se repite para todos los bits de las entradas, con excepción del bit de signo para el cual se debe de efectuar una substracción en lugar de una adición [DEFI89].

Por otra parte, en algunos casos al período de reloj en el cual todos los bits de signo llegan simultáneamente, se le llama ''Tiempo del Bit de Signo'' (Sign-bit Time). Adicionalmente a esta designación, generalmente conviene asumir que es nulo el intervalo de tiempo existente entre el instante en el cual se recibe una dirección en la ROM y el instante en el que esta memoria responde con un dato en sus salidas; respecto al retardo producido en la malla del acumulador conviene asumir que no es mayor a un ciclo de reloj, consumiéndose dicho tiempo en el sumador del ALU [WHIT89].

Ahora bien, con el propósito de aplicar la transformada discreta coseno a un conjunto de N datos, se requiere utilizar N procesadores de aritmética distribuida, considerando que cada uno de ellos debe calcular un producto interno con N términos. Lo anterior presenta la complicación de que para lograr la DCT de 16 puntos, la memoria ROM asociada a cada procesador necesita contener 2^{16} coeficientes [DEFI89], lo que resulta en un consumo excesivo de memoria; sin embargo, afortunadamente es posible reducir el tamaño de la memoria ROM al explotar las simetrías de las funciones base de la DCT, manifiestas en la matriz asociada a la transformación puesto que, como ya se mencionó, la matriz que se genera con las expresiones (38) y (39) del algoritmo MSDCT, son alternadamente simétricas o antisimétricas respecto a la columna central, lo cual conduce a la expresión siguiente:

$$X(k) = x(0)a_{k0} + x(1)a_{k1} + \ldots \pm x(N-2)a_{k1} \pm x(N-1)a_{k0} \qquad (45)$$

De la expresión (45) se puede observar que es posible factorizar términos comunes, a fin de presumar, o prerestar, los datos de entrada; al momento de efectuar estos cómputos previos, se está también realizando lo que constituye la primera etapa del mariposeo efectuado en la mayoría de los algoritmos denominados rápidos. Por otra parte, la importancia de efectuar este tipo de precálculos se refleja, de forma inmediata, en el hecho de poder realizar la DCT de N puntos, usando productos internos de N/2 términos; con base en esto, la expresión (45) se puede agrupar quedando como:

$$X(k) = [x(0) \pm x(N-1)]a_{k0} + [x(1) \pm x(N-2)]a_{k1} + \ldots \qquad (46)$$

Como se puede deducir, el cálculo de la DCT aún requiere de N procesadores de aritmética distribuida; sin embargo, ahora la memoria ROM de cada procesador contiene únicamente $2^{N/2}$ coeficientes, lo cual se traduce en el caso de una DCT de 16 puntos, a 256 coeficientes en lugar de los 65,536 del arreglo inicial del algoritmo MSDCT [DEFI89]. Conviene

aclarar que, en principio, la técnica utilizada para reducir el tamaño de la memoria ROM puede aplicarse nuevamente en diferentes etapas, lo que equivale a un cierto mariposeo de los datos, efectuado en la mayoría de los algoritmos de transformadas rápidas; sin embargo, el ahorro logrado en tiempo no es suficientemente redituable como para equiparar el incremento de las irregularidades en que se incurre [MING89], lo cual dificulta y complica notablemente el diseño de la arquitectura del procesador propio del sistema a implantar.

Ahora bien, a fin de ilustar el procedimiento descrito en la expresión (46), para el cálculo de la DCT, la figura 3.2 muestra la disposición básica de un arreglo para efectuar el cómputo de la DCT, en el caso de contar tan sólo con 8 puntos. Como se puede apreciar en esa figura, es necesaria la presencia de 8 procesadores de aritmética distribuida, en los cuales el tamaño de la memoria ROM es reducido, si se considera y aprovecha la simetría por columnas de la matriz asociada a la DCT.

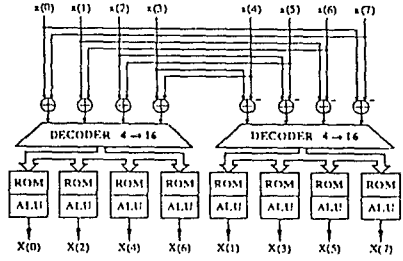


Figura 3.2 Cálculo de la DCT de 8 puntos usando 8 procesadores en paralelo. El tamaño del ROM se reduce debido a la simetría por columnas.

Por otra parte, conviene recordar que debido a la naturaleza del algoritmo MSDCT, lo que se ha expuesto hasta ahora es igualmente aplicable tanto al cálculo de la transformada directa, así como la transformada inversa DCT; razón por la cual la estructura básica que se tiene descrita, no presenta variaciones considerables para uno u otro caso, lo que constituye una de las principales ventajas de este algoritmo puesto que la arquitectura a desarrollar es fundamentalmente la misma para las dos transformaciones, como podrá constatarse en secciones posteriores.

IV. ASPECTOS GENERALES DE LOS PROCESADORES ORIENTADOS AL CALCULO DE LA DCT.

Antes de analizar con detalle la organización genérica de los procesadores que se utilizan exprofeso para el cálculo de la DCT bidimensional, conviene tomar en cuenta la resolución y, por ende, la precisión con que deberá operar el sistema a desarrollar. Una adecuada selección de la resolución acarrea consigo el establecimiento de varias características, tanto físicas como operativas, de los diferentes elementos que intervienen en la implantación de la DCT en medios electrónicos.

Cabe señalar que algunas de las implantaciones electrónicas de la DCT en dispositivos VLSI, consideran datos de entrada con una precisión entre 8 y 9 bits, manejándose cantidades ya sea sin signo o en complemento a dos. Al cabo de la primera etapa de transformación de la DCT (16x1 DCT), generalmente se mantiene una precisión interna entre 12 y 16 bits; produciéndose como resultado final, datos de salida con una precisión entre 12 y 14 bits dependiendo del dispositivo a utilizar. En la mayoría de los casos, los coeficientes precalculados para las tablas de asociación de datos (look-up tables) se consideran con una precisión entre 9 y 14 bits [MING89].

Ahora bien, al momento de pretender utilizar el algoritmo MSDCT para el cálculo de la transformada coseno, conviene recordar que se busca aprovechar la misma estructura, tanto para la transformada directa como para la transformada inversa. Adicionalmente a esto, es pertinente tomar en cuenta que este proyecto está orientado al tratamiento de señales de video digitalizadas; razón por la cual se considera que las entradas y las salidas cuentan con una longitud de palabra igual a 8 ó 12 bits, dependiendo si se trata, respectivamente, de la DCT directa (FDCT) o de la DCT inversa (IDCT); manejándose, mediante una precisión interna de 12 bits, los diferentes coeficientes precalculados. Es oportuno mencionar que en algunos modelos y prototipos experimentales, se considera que tanto las entradas como las salidas son de 16 bits, en tanto que los coeficientes precalculados son de 10 bits [DEFI89].

Por otra parte, y de acuerdo con lo expuesto en secciones anteriores, los procesadores de aritmética distribuida son de naturaleza serie; sin embargo, se requiere almacenar ciertos datos intermedios, con una resolución fija y en paralelo, a fin de efectuar con ellos operaciones tales como la transposición en memoria, de la información procesada en la unidades de DA. Esto último conduce a pensar que las tres partes constitutivas esenciales para todo dispositivo VLSI destinado al cálculo de la DCT bidimensional son:

i) Banco de procesadores de DCT para una dimensión (1D-DCT).

ii) Memoria para resultados intermedios (IRM).

iii) Banco de Registros de corrimiento (SRB).

La adecuada relación de estos tres elementos da lugar al diagrama a bloques que se muestra en la figura 4.1, el cual básicamente describe la estructura interna que se debe considerar en todo procesador dedicado a la obtención de la DCT para el caso bidimensional.

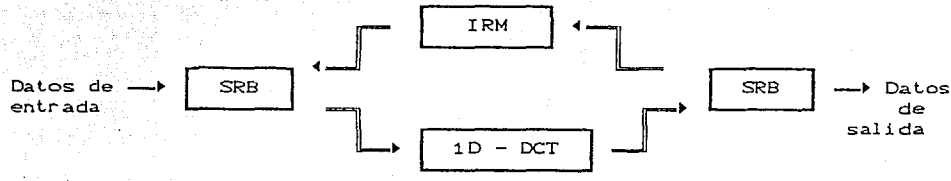


Figura 4.1 Diagrama a bloques del procesador por utilizar.

Con base en este último diagrama, se puede esbozar la arquitectura genérica del procesador destinado a ser la parte operativa del sistema por desarrollar, con el fin de lograr la implantación física de la DCT para el tratamiento de imágenes digitales. La estructura propia de dicha arquitectura genérica se muestra gráficamente en la figura 4.2.

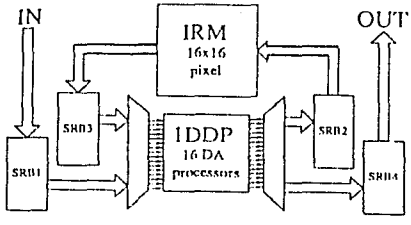


Figura 4.2 Arquitectura global de un procesador para el cálculo de la DCT bidimensional (2D-DCT)

Cabe mencionar que, en términos generales, la parte medular de este subsistema se encuentra en el banco de procesadores encargados de efectuar la transformada DCT unidimensional [DEFI89]. En el caso particular de la aplicación en estudio, ese banco de procesadores está constituido por 16 procesadores de aritmética distribuida, mismos que permiten efectuar simultáneamente el cálculo concurrente de 16 productos internos.

En lo que se refiere a la memoria para resultados intermedios (IRM), se puede afirmar que se trata de una RAM de alta velocidad, la cual está destinada a retener la información obtenida justo después de efectuar la primera transformación unidimensional del tipo 1D-DCT [DEFI89]. Conviene mencionar que la memoria IRM juega un papel estratégico muy importante en el procesador por especificar, puesto que en esta sección es donde precisamente se efectúa, de manera implícita, la transposición matricial que permite obtener la transformada coseno discreta bidimensional.

Como también se desprende de analizar la figura 4.2, existen dentro de esta arquitectura cuatro bancos de registros de corrimiento, de 16 bits cada uno, destinados a enlazar adecuadamente los distintos módulos operativos en que se subdivide al procesador.

En secciones posteriores, se profundizará en el análisis de las características que deban presentar los diferentes módulos operativos que constituyen al sistema en estudio.

Por otra parte, las principales ventajas que ofrece emplear la arquitectura genérica que aquí se plantea son [DEFI89]:

-Contar con el máximo paralelismo posible (16 procesadores en paralelo).

-Permitir un uso óptimo de la circuitería disponible, v.g. todas las partes del sistema se encuentran activas en todo instante.

-Permitir que todas las partes del sistema actúen síncronamente.

-Contar con una estructura completamente del tipo pipeline, permitiéndose una alta velocidad así como un gran rendimiento en el procesamiento.

-Permitir que se efectúen las operaciones propias de los productos internos de tal manera que se genere el menor error posible en la cuantización.

Antes de continuar, es interesante notar que el haber estandarizado la capacidad de los registros internos del sistema en 16 bits cada uno, es ventajoso aún en el caso de pretender emplear caracteres con una menor longitud en bits; la afirmación anterior tiene lugar si se considera que también en esos casos existen suficientes bits de guarda o protección, mismos que permiten la adecuada detección de los errores de sobreflujo, así como contar con una buena lógica de corrección [DEFI89].

V. ELEMENTOS ARQUITECTONICOS DE LOS PROCESADORES ORIENTADOS AL CALCULO DE LA DCT BIDIMENSIONAL.

Como ya se ha mencionado, la arquitectura genérica de esta clase de procesadores tiene como base la estructura que se desprenda de interrelacionar los bancos de procesadores 1D-DCT, los bancos de registros de corrimiento y la memoria de resultados intermedios. Al considerar la importancia que tiene para este proyecto el lograr un mejor entendimiento del comportamiento de cada uno de esos elementos, conviene entonces precisar a continuación ciertas características que resultan de interés para el diseño de un sistema orientado a soportar el cálculo de la DCT bidimensional.

BANCOS DE REGISTROS DE CORRIMIENTO.

Después de tomar en cuenta la organización general del sistema, ilustrada mediante la figura 4.2, se considera que para los cuatro bancos de registros de corrimiento disponibles, tan solo se emplean dos tipos diferentes de elementos: aquellos registros del tipo (entrada paralelo)-(salida serie) y aquellos registros del tipo (entrada serie)-(salida paralelo). En este caso, los registros SRB1 y SRB3 caen en el primero de los tipos y su función respectiva es capturar los datos provenientes tanto del exterior del procesador como de la memoria de resultados intermedios. Por otra parte, los registros SRB2 y SRB4 caen en la segunda categoría y su función es retener, alternadamente, los resultados de los cálculos efectuados para los diversos productos internos en los procesadores 1DDP [DEFI89], mismos que son de naturaleza serie.

Conviene mencionar que en esta aplicación en particular, es patente la necesidad de mantener una adecuada regularidad tanto en las acciones de control como en las comunicaciones locales, internas y externas, para cada bloque operativo. Por tanto, dichas acciones deben ser eficientemente coordinadas por la unidad de control, propia del sistema por desarrollar.

Antes de concluir el tratamiento de este punto, es oportuno notar que los bancos de registros SRB1 y SRB4 se comunican al exterior del sistema de cálculo de la DCT; sin embargo, dichos registros no se comunican con la memoria de resultados intermedios. Asimismo, los registros SRB2 y SRB3 se comunican únicamente con la memoria IRM; sin embargo, esos registros no se comunican directamente con el exterior del sistema [DEFI89], por lo cual se tiene, mediante estos registros de corrimiento, un acoplamiento estrecho entre la IRM y el conjunto de procesadores de DA. Cabe señalar que tanto los registros de corrimiento como la IRM permiten afirmar que cada procesador de DA cuenta con un banco de memoria local propia.

Finalmente, es curioso notar que si se considera desarrollar esta clase de bancos mediante el uso de dispositivos MSI, similares a los registros de corrimiento del tipo TTL, entonces serán requeridos aproximadamente 128 circuitos integrados semejantes. Tal cantidad resulta de multiplicar el total de bancos del sistema (4), por la cantidad de registros existentes en cada banco (16), y este último resultado se multiplica a su vez por dos, si se toma en cuenta el uso de registros convencionales de 8 bits cada uno.

PROCESADORES DE ARITMETICA DISTRIBUIDA.

Como ya se ha mencionado, los procesadores de aritmética distribuida constan básicamente de dos partes: La ALU y la ROM de soporte. Resulta evidente que de entre esos dos elementos, la sección más importante es indiscutiblemente la ALU [DEFI89], ya que de su adecuado diseño se desprenden la mayor parte de las características operativas de los procesadores de DA. Lo anterior acarrea consigo la estipulación de las principales condiciones en que se desenvolverá el sistema encargado de efectuar el procesamiento digital de las señales a tratar durante este proyecto. Respecto al bloque de ROM requerido por cada procesador de DA, conviene decir que su diseño se ve notablemente influenciado por el comportamiento propio de los parámetros que intervienen en la unidad aritmética.

Cabe mencionar que la ALU a considerar para este proyecto, debe soportar eficientemente el cálculo del producto interno requerido por la DCT, lo que se logrará mediante el uso de un algoritmo basado únicamente en sumas y corrimientos. A fin de analizar el diseño de la ALU para los procesadores 1DDCT, conviene tener en cuenta que existen diferentes esquemas con los que es factible lograr la implantación de un arreglo de multiplicadores, como el requerido para la unidad aritmética en cuestión. Entre esos esquemas se encuentran el de *Sumador con Ahorro de Acarreo (Carry Save Adder* CSA), *el Arbol de Wallace* y *el esquema de Dadda* [KUNG88].

Ahora bien, con el propósito de entender adecuadamente la ALU requerida para los procesadores de aritmética distribuida, es de particular interés el tener en cuenta la técnica de ahorro de acarreo (CSA), puesto que mediante ella es factible diseñar eficientemente esa clase de dispositivos. En lo que respecta al término de *"ahorro de acarreo"*, éste proviene de la manera en que los bits de acarreo se propagan al momento de obtener los resultados parciales en el proceso de adición [KUNG88] propio de los procesadores de DA. Aunado a lo anterior, se puede afirmar que la principal característica de los sumadores con ahorro de acarreo, es la eliminación del tiempo de propagación del bit

de acarreo, puesto que se tienen trayectorias de cómputo con un retardo mínimo [DEFI89].

La figura 5.1 ilustra la manera en la que se podría lograr la implantación física de la ALU requerida, considerando la técnica CSA.

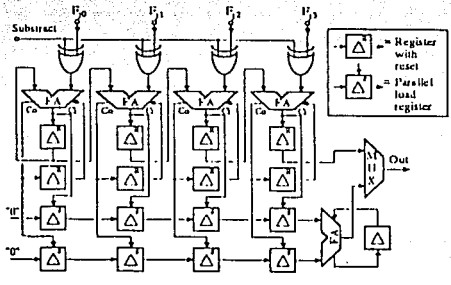


figura 5.1 Sumador con ahorro de acarreo, empleando registros pipeline

Es conveniente mencionar que después de efectuar el procesamiento en un arreglo de sumadores CSA, el resultado producido está en el formato de ahorro de acarreo; razón por la cual es necesario pasar por una etapa adicional de sumado, a fin de obtener el producto final. En este caso, el uso de un sumador convencional de acarreo sería inadecuado debido a su limitada rapidez de operación. Por tal motivo, frecuentemente son utilizadas técnicas de anticipación de acarreo (*carry look-ahead*) a efecto de acelerar el proceso de adición. La anticipación del acarreo surge a partir del fenómeno en el cual estos bits pueden ser generados o predecidos aún antes de que se obtengan las respectivas sumas en una cierta etapa del proceso. Estos acarreos anticipados, mismos que son entradas de todas las posiciones de los bits de un sumador paralelo, son generados por algún tipo de circuitería lógica adicional, lo que se traduce en una constante de tiempo propia de la adición, independientemente de la longitud del sumador [KUNG88].

Con lo expuesto hasta aquí, se puede afirmar que en un procesador de aritmética distribuida, el resultado no se encuentra directamente disponible justo después de que el último bit de las entradas ha sido introducido, por vía de un corrimiento, al ALU; debido esto al paralelismo masivo de la estructura a emplear. Por otra parte, todo contenido de los registros internos debe ser vaciado al exterior, mediante corrimientos, antes de que el cálculo de un nuevo producto interno pueda tener lugar. El efectuar este vaciado operativamente tendrá un costo de W_c ciclos de reloj adicionales (recordando que W_c representa el ancho de palabra de la ROM); sin embargo, y a fin de aumentar la tasa de trabajo útil del sistema, se introduce el uso de registros pipeline, mismos que cuentan con un cargado en paralelo.

A partir de lo anterior, conviene aclarar que cuando se ha calculado el último bit de un producto interno, entoces el contenido de los registros de suma y acarreo se copia en los registros pipeline, con el propósito de que el cálculo del próximo producto interno tenga lugar inmediatamente. Respecto a los registros pipeline, se puede decir que estos son del tipo entrada paralelo salida serie. Además, se emplea un sumador completo (full adder) en la etapa de salida, a efecto de sumar adecuadamente los resultados parciales así como los acarreos existentes. Finalmente y con objeto de completar el diseño del procesador, es necesario utilizar un multiplexor para seleccionar la parte deseada del resultado [DEFI89].

MEMORIA DE RESULTADOS INTERMEDIOS.

También conocida como memoria de transposición, juega un papel estratégico muy importante en cualquier sistema desarrollado tendiente a efectuar la DCT bidimensional. Lo anterior se afirma con base en que una vez efectuada la transformada 1D-DCT, el bloque de 256 registros con datos intermedios es transpuesto antes de efectuar la segunda etapa de transformación 1D-DCT. Recuérdese que este caso se limita a la transformación de bloques matriciales de orden 16x16; razón por la cual la RAM de transposición contiene un total de 256 localidades, mismas que deben permitir la escritura y lectura simultánea mientras se efectúa el proceso de transposición [MING89].

Por otra parte, si se considera que los resultados intermedios del bloque actualmente en proceso, son continuamente escritos en la memoria y estos datos provienen del cálculo de la primera DCT de 16x1, efectuándose esto mientras los resultados intermedios del bloque previo están siendo leídos para el cálculo de la segunda DCT de 16x1, entonces las operaciones de lectura/escritura deben ser arregladas de tal suerte que la información no sea destruida antes de ser leída u obtenida por algún dispositivo exterior. Es factible lograr lo anterior mediante un arreglo que contemple las señales de control tanto de lectura/escritura, como las de direccionamiento, de tal manera que cada muestra sea escrita en la misma localidad de la cual se leyó la muestra del bloque previo. A fin de lograr la transposición matricial, los datos son leídos exteriormente por columnas si el bloque previo fue escrito en la RAM por renglones y viceversa [MING89].

Es importante mencionar que la secuencia de direcciones para lectura y escritura, es generada por un contador de ocho bits. El cambio de la secuencia (lectura/escritura por columna o por renglón) se logra al intercambiar los cuatro bits más significativos con los cuatro bits

menos significativos. El contador se puede implantar mediante un arreglo formado por un sumador y un registro de corrimiento. Conviene aclarar que la señal de inicio de bloque sirve para poner en condiciones iniciales a las direcciones que intervienen en la operación de lectura/escritura [MING89].

Dada la naturaleza del procesamiento a efectuar, las celdas de memoria podrían no requerir de un refresco de información, lo que conduce a pensar que esta clase de elemento puede ser implantado mediante memorias del tipo estático o, en caso dado, dinámico, considerando tener una escasa lógica de refresco. Es prudente aclarar que si bien se está realizando un procesamiento serie, la memoria a emplear puede tener entradas en paralelo si se considera la salida simultánea de los diferentes elementos de procesamiento involucrados.

Como se desprende de lo anterior, es factible lograr la implantación de este tipo de memoria de transposición mediante el arreglo eficiente de dispositivos convencionales de memoria RAM, tanto estática como dinámica, sin embargo no está por demás considerar la rapidez con que se debe efectuar el procesamiento para el cálculo de la DCT bidimensional.

Ahora bien, después de analizar los diversos elementos que constituyen la estructura de un procesador genérico orientado al cálculo de la DCT bidimensional, cabe señalar que varios microprocesadores comerciales se ajustan estrechamente a lo que hasta aquí ha sido mencionado, contando con los tres elementos básicos antes descritos, dentro de su organización interna. Son ejemplo de tales procesadores comerciales los fabricados por la compañía INMOS del grupo SGS-Thomson Microelectronics; entre los que se encuentran los dispositivos IMS A121 (2D-DCT de 8x8), STV 3208 (2D-DCT 8x8) y STV 3200 (2D-DCT 16x16), de los cuales se anexan copias de sus hojas de especificación técnica en el apéndice I.

A partir de lo expuesto hasta aquí, se llega a la conclusión de que para poder diseñar adecuadamente la arquitectura de un sistema digital que mediante dispositivos VLSI obtenga físicamente la DCT bidimensional de imágenes digitales, es conveniente precisar ciertos conceptos vinculados con la forma de manejar los algoritmos destinados a ser implantados en procesadores en paralelo o en arreglo (parallel and array processors), a fin de lograr un pleno entendimiento de las fases de diseño que restan por ser tratadas en las secciones siguientes de este estudio.

VI. ALGORITMOS PARA PROCESADORES EN PARALELO Y EN ARREGLO.

En las secciones previas se ha establecido un marco teórico alrededor de los esquemas de transformación más utilizados en el procesamiento digital de imágenes; asimismo, han sido introducidos los conceptos básicos de una arquitectura genérica que soporte la obtención de la transformada DCT bidimensional. Sin embargo, a fin de caracterizar adecuadamente el modelo a seguir para el sistema de cómputo orientado a satisfacer el objetivo planteado, conviene hacer hincapié en ciertas peculiaridades que es necesario tomar en cuenta al momento de implantar una arquitectura compuesta por múltiples procesadores, mismos que pueden estar dispuestos en paralelo o en arreglo (Array Processors). Para esto último, resulta importante entender que un algoritmo en arreglo se define como el conjunto de reglas que se utilizan para resolver un determinado problema mediante un número finito de pasos desarrollados en múltiples procesadores interconectados todos ellos entre sí. Por lo tanto, un algortimo en arreglo depende estrechamente de las características propias de la máquina que se esté utilizando, así como de las estrategias empleadas para la interconexión entre los procesadores [KUNG88]. Además, un arreglo de procesadores puede ser clasificado como una arquitectura tipo SIMD de acuerdo al modelo propuesto por Flynn, considerándose esta organización como una computadora vectorial implantada mediante un conjunto de elementos de procesamiento idénticamente sincronizados, capaces de ejecutar simultáneamente la misma operación sobre diferentes datos de entrada y, a menos que los elementos de procesamiento actúen en paralelo, las unidades pueden ser programadas para ignorar alguna instrucción en particular. Por su parte, los modelos SIMD pueden variar en dos aspectos: El número de procesadores puede ser fijo o ilimitado, y los procesadores pueden intercomunicarse entre sí ya sea por vía de memoria compartida, una red conectada en malla, una red tipo pirámide, una red perfectamente mezclada, una red conectada en cubo o una red con ciclos cúbicamente interconectados [QUIN88].

Ahora bien, en lo que se refiere a la concurrencia en un algoritmo en arreglo, ésta es muy importante para lograr la mayor cantidad de trabajo útil por unidad de tiempo; sobre todo al momento de emplear procesadores en arreglo con tecnología VLSI. En términos generales, la concurrencia frecuentemente se logra aplicando el principio de modularidad al descomponer un problema en subtareas independientes (ejecutables en paralelo), o en subtareas dependientes que puedan ser ejecutadas con cierto translape aplicando lo que se conoce como pipeline. Sin embargo, conviene tener en cuenta que el grado de concurrencia varía significativamente de una técnica a otra [KUNG88].

Por otra parte, el punto crucial que más afecta a la eficiencia del procesamiento en paralelo es el que se refiere a las comunicaciones y, en particular, a los esquemas de transferencia de datos entre los diferentes elementos de procesamiento que constituyen las redes de interconexión de gran escala, propias de los sistemas en arreglo. Consecuentemente a esto, efectuar un análisis orientado a las comunicaciones de los algoritmos concurrentes, resultaría útil al momento de mapear los algoritmos sobre los arreglos de procesadores. Ahora bien, debido a los alcances de este estudio es posible avocarse a una clase especial de algoritmos denominados recursivos y localmente

independientes. Entendiéndose que en un algoritmo recursivo, todos los procesadores realizan tareas casi idénticas, adicionalmente a que cada elemento de procesamiento ejecuta, repetidamente, un conjunto fijo de tareas sobre la secuencia de datos disponibles [KUNG88].

Es importante mencionar que el diseño efectivo de un algoritmo debe iniciar con un completo entendimiento de las especificaciones del problema a resolver, a fin de posteriormente efectuar un análisis matemático y poder así efectuar un análisis algorítmico de optimización y concurrencia. Con el propósito de maximizar el procesamiento en paralelo, así como el pipeline, conviene utilizar una gráfica de dependencias, puesto que ésta se convierte en una efectiva herramienta que muestra el total de las dependencias entre los datos que intervienen en la ejecución de un cierto algoritmo. Cabe mencionar que dichas dependencias constituyen las mayores restricciones a las que se ve sometida una secuencia de procesamiento [KUNG88] y limitan la explotación de la concurrencia potencialmente disponible en un arreglo de procesadores. Por tal motivo, y antes de efectuar la implantación tanto de un algoritmo como de una arquitectura, es necesario tener en cuenta una metodología y criterios apropiados que permitan lograr el diseño eficiente de un arreglo de procesadores.

De acuerdo a lo anterior, es oportuno considerar que la efectividad de mapear las actividades de un algoritmo sobre un arreglo de procesadores, se encuentra directamente relacionada con la manera en la cual se descompone dicho algoritmo. A partir de esto, se puede afirmar que dos algoritmos distintos con un rendimiento semejante dentro de una computadora secuencial, pueden comportarse totalmente de manera diferente en un contexto destinado al procesamiento en arreglo [KUNG88].

En lo que respecta al análisis convencional de algoritmos secuenciales, la complejidad de un algoritmo depende de los cálculos y la memoria requerida, siendo el factor más importante la contabilización de los cálculos a desarrollar. Por ejemplo, y como ya se ha mencionado previamente, se sabe que el algoritmo de la FFT es superior al de la DFT en términos de la cantidad de operaciones: NlogN operaciones de la FFT contra N^2 operaciones de la DFT. Lo cual es una comparación muy válida en una máquina tradicional del tipo Von Neumann, donde cada operación se encuentra acompañada de un factor de utilización de componentes uniforme en cada ciclo de escritura y/o captura de memoria. Sin embargo, en un contexto formado por esquemas de procesamiento en paralelo, debe de contemplarse que dicho factor no es uniforme debido a la gran cantidad de elementos de procesamiento que son utilizados, considerando que el tiempo de utilización de componentes depende críticamente tanto de la disponibilidad como de lo accesible de los recursos de cómputo a utilizar. Cabe señalar que una menor cantidad de operaciones no implican necesariamente un menor tiempo de procesamiento, lo que en otras palabras significa que la simple contabilización de operaciones no puede ser considerada como un método efectivo para medir el rendimiento de un procesador [KUNG88].

VI.1 ASPECTOS QUE INTERVIENEN EN EL DISEÑO DE ALGORITMOS EN ARREGLO PARA VLSI.

Con base en lo anterior, resulta evidente que se requiere de nuevos criterios para determinar la eficiencia de un cierto algoritmo por programar, puesto que la simple contabilización de operaciones adoptada por las máquinas secuenciales no es del todo adecuada en un esquema de procesamiento paralelo. Aunado a esto, es necesario considerar lo restrictivo de los problemas de comunicaciones asociados con la tecnología VLSI, lo que conduce a tomar muy en cuenta el aspecto potencialmente significativo que representan los costos de interconexión. Es importante mencionar que dentro de los criterios utilizados en la moderna computación mediante arreglos de procesadores, deben de ser incluídos algunos factores de influencia en el procesamiento, como lo son el grado de paralelismo y la tasa de pipeline; entendiéndose por esa tasa el recíproco del intervalo del tiempo que transcurre entre dos entradas consecutivas de datos [KUNG88].

A fin de precisar un poco más estos dos últimos conceptos, cabe señalar que el procesamiento en paralelo y el pipeline son dos métodos que mejoran la concurrencia de un algoritmo; sin embargo, el procesamiento en paralelo es un tipo de procesamiento de información donde se enfatiza la manipulación concurrente de datos pertenecientes a uno o más procedimientos que resuelven un solo problema, en tanto que el pipeline incrementa la concurrencia al dividir la computación en un determinado número de pasos, siendo esto contrastante con el paralelismo el cual utiliza multiples recursos para incrementar la concurrencia [QUIN88]

Por otra parte, al momento de diseñar un algoritmo en arreglo para un procesamiento en arreglo, es conveniente que sean considerados en la secuencia de diseño, aspectos tales como la computación involucrada; las comunicaciones locales y modulares; la memoria y los dispositivos de entrada/salida que sean requeridos. De entre todos esos aspectos, es importante conocer y delimitar las siguientes características generales:

Máximo paralelismo.

La experiencia ha comprobado que dos algoritmos diseñados con un rendimiento semejante en una computadora secuencial convencional, pueden comportarse totalmente diferentes en un medio de procesamiento en paralelo. Un algoritmo puede verse favorecido si expresa un alto grado de paralelismo, el cual es explotable por los arreglos de cómputo [KUNG88] con que cuente el sistema en uso. Esto permite afirmar que antes de evaluar la potencialidad de un algoritmo en cuanto a la aplicación por resolver, es necesario plantear el esquema de cómputo que lo va a soportar, a fin de explotar al máximo posible el paralelismo del sistema así como los recursos existentes tanto de hardware como de software.

Pipeline máximo.

La gran mayoría de los algoritmos para el procesamiento de señales demandan una gran cantidad de trabajo útil por unidad de tiempo, siendo computacionalmente intensivos (Comparados con respecto a sus requerimientos de entrada/salida). La explotación del efecto pipeline a menudo es lograda con naturalidad en redes de elementos de procesamiento, formadas con gran regularidad y conectadas localmente; por lo tanto, la mayor parte de la concurrencia necesaria en un procesamiento en arreglo, se deriva directamente del pipeline. A partir de lo cual es claro que para maximizar la tasa de trabajo útil por unidad de tiempo, es necesario seleccionar el mejor de todos los algoritmos así como el mejor de los arreglos, dependiendo de la aplicación por resolver. Lo anterior conduce a afirmar que los dispositivos VLSI que efectivamente están constituídos con arreglos de procesadores, presentan el máximo pipeline posible; razón por la cual requieren de algoritmos bien estructurados que cuenten con movimientos predecibles de datos. Cabe mencionar que los métodos iterativos que presentan ramificaciones dinámicas, y que son dependientes de los datos producidos durante el procesamiento, son los menos adecuados para ser implantados en arquitecturas del tipo pipeline [KUNG88].

Balance entre los cómputos, las comunicaciones y la memoria.

Un buen algoritmo en arreglo, debería ofrecer un adecuado balance entre diferentes anchos de banda en que se incurre en las diversas jerarquías de las comunicaciones que intervienen en el sistema de cómputo, a fin de evitar el drenado de datos o innecesarios cuellos de botella al momento de procesar información. El balancear los cálculos a desarrollar, con los variados anchos de banda que afectan a las comunicaciones, es crucial para la efectividad en la operación de un arreglo constituído con elementos de procesamiento. La tecnología actual permite mejorar sin dificultad alguna, el ancho de banda de los cómputos a efectuar; sin embargo, es interesante resaltar que resulta mucho más difícil incrementar el ancho de banda de los dispositivos de entrada/salida. En este caso, las técnicas del tipo pipeline son especialmente adecuadas para lograr el balance entre los cálculos y las entradas/salidas, puesto que bajo esos esquemas los datos tienden a distibuirse sobre tantos elementos de procesamiento como es posible, antes de dejar el arreglo; lo cual reduce el ancho de banda de las entradas/salidas para las comunicaciones externas. En problemas de cómputo con ciertas restricciones tales como las multiplicaciones matriciales, los ordenamientos y el mariposeo de la FFT, etc., si el ancho de banda de los cálculos se incrementa mientras que el ancho de banda de las entradas/salidas permanece constante, entonces el tamaño de la memoria tiene necesariamente que incrementarse a fin de obtener el balance entre los cómputos y las entradas/salidas por realizar [KUNG88].

Compromisos entre los cómputos y las comunicaciones.

Con el propósito de hacer que la red de interconexión en un sistema en arreglo sea práctica, eficiente y suficiente, se deben permitir esquemas de comunicación regulares. Cabe mencionar que los factores clave que afectan a la regularidad en las comunicaciones incluyen aspectos tales como la localidad o la globalidad de las comunicaciones, el ruteo estático o dinámico así como la interconexión entre módulos con datos dependientes contra la iterconexión entre módulos con datos independientes. El criterio a seguir en estos casos, debe considerar maximizar el compromiso que se establece entre los costos de interconexión y el trabajo útil por unidad de tiempo. A fin de satisfacer las restricciones en el aspecto de comunicaciones impuestas por la tecnología VLSI, recientemente se le ha dado énfasis al desarrollo de algoritmos que tomen en cuenta los principios de localidad y recursividad. En términos generales, se prefieren los algoritmos que requieren una red estática, sobre los algoritmos que requiren de una red dinámica puesto que estas últimas son más difíciles de construir [KUNG88].

Rendimiento numérico y efectos de cuantización.

El comportamiento numérico depende de varios factores entre los que se encuentran la longitud de la palabra y el tipo de algoritmos empleados. A manera de ejemplo, en el caso matricial se prefiere una descomposición del tipo QR sobre la descomposición LU para resolver sistemas lineales puesto que la primera cuenta con un comportamiento numérico más estable. Sin embargo, el precio que se paga en la descomposición QR es efectuar mayor cantidad de operaciones que en el caso LU. A menudo se aprovecha el hecho de efectuar más operaciones para mejorar el rendimiento numérico total; pero conviene tener en cuenta que el compromiso existente entre la computación y el comportamiento numérico, es un problema dependiente por lo que no hay una regla general que aplicar [KUNG88].

Si bien, conocer cada uno de estos aspectos es importante al definir la secuencia de pasos a seguir al momento de diseñar tanto el algoritmo como la organización del sistema que soporte el procesamiento deseado, esto no es suficiente, puesto que también se requiere considerar algunos aspectos relacionados con las arquitecturas de los sistemas en los que efectivamente están involucrados tanto los procesadores como los algoritmos en arreglo. Es importante mencionar que tales aspectos deben de considerar la forma en la cual se mapea un algortimo en arreglo sobre un conjunto de procesadores en arreglo, lo cual será tratado en las próximas secciones de este estudio.

VI.2 ARQUITECTURAS Y ALGORITMOS SISTOLICOS.

Al tomar en cuenta tanto los fines que persigue este estudio junto con los aspectos descritos en la sección anterior, se llega a concluir que la arquitectura que mejor soporte la obtención de la DCT bidimensional deberá ser capaz de manejar dispositivos VLSI dispuestos en arreglos de procesadores, contando con algoritmos que puedan manejar adecuadamente tal disposición. Por tal motivo, es conveniente analizar qué tipos de sistemas obtienen los mejores beneficios de los procesadores en arreglo y en paralelo que fueron tratados durante las secciones IV y V del presente estudio, lo que conduce, sin lugar a dudas, al análisis de las arquitecturas y de los algoritmos de tipo sistólico, mismos que serán tratados a continuación.

En términos generales, los procesadores sistólicos se consideran como una nueva clase de arquitectura de los procesadores pipeline que estén dispuestos en arreglo. Por otra parte, y de acuerdo con Kung y Leirserson, un sistema sistólico es una red de procesadores que rítmicamente efectúan cálculos y pasan los resultados al resto del sistema de cómputo. Es oportuno señalar que los sistemas sistólicos consideran en gran medida las importantes propiedades de modularidad, regularidad, interconexión local, alto grado de paralelismo y multiprocesamiento altamente sincronizado [KUNG88]. Cabe añadir que un algoritmo sistólico es un tipo especial de un algoritmo tipo pipeline, caracterizándose por tres atributos que son: Primero, el flujo de datos es rítmico y regular; segundo, los datos pueden fluir en más de una dirección; tercero, los cálculos efectuados por cada sección son esencialmente idénticos. Al igual que un algoritmo tipo pipeline, un algoritmo sistólico requiere implícitamente la adecuada sincronización entre las partes del proceso que producen datos y las que los consumen [QUIN88].

Por otra parte, el diseño de los arreglos sistólicos difiere del que se realiza en las computadoras convencionales tipo Von Neumann dado el alto grado de pipeline que se logra en los primeros. Al ver esto con más detalle quiere decir que una vez que se ha traído un dato de la memoria, éste puede ser efectivamente utilizado por cada elemento de procesamiento por el que va pasando al momento de ser ''bombeado'' de elemento a elemento por todo el arreglo. Esto último resulta especialmente útil para una amplia gama de cálculos donde se realizan múltiples operaciones sobre cada dato, siendo éstas, además, en forma repetitiva. Por tal motivo, los arreglos de esta naturaleza reducen el clásico problema del cuello de botella que se forma en el acceso a la memoria de una máquina tipo Von Neumann [KUNG88].

En cuanto a las tareas de cómputo, éstas pueden ser clasificadas conceptualmente en dos grandes familias: procedimientos limitados por los cálculos y procedimientos limitados por las interacciones de entrada y salida. Si en un procedimiento de cómputo la cantidad de operaciones es mayor que la cantidad de interacciones de entrada salida, entonces ese procedimiento está limitado por los cálculos; en caso contrario

está limitado por la interacción entrada/salida. A manera de ejemplo puede considerarse que un procedimiento para efectuar la multiplicación cae en el primero de los casos, en tanto que un procedimiento que realice la adición matricial cae en el segundo. A fin de acelerar los procedimientos limitados por las interacciones de las entradas y salidas, se requiere de un aumento en el ancho de banda de la respuesta de las memorias; lo que resulta difícil con la tecnología que actualmente está disponible. En tanto que acelerar los procedimientos limitados por operaciones de cálculo, a menudo puede lograrse mediante el uso de arreglos sistólicos. La configuración básica de un arreglo sistólico es la que se muestra en la figura 6.1, donde al reemplazar un procesador simple por un arreglo unidimensional o bidimensional de procesadores, se puede alcanzar una mayor cantidad de trabajo útil por unidad de tiempo, sin la necesidad de incrementar el ancho de banda de las memorias.

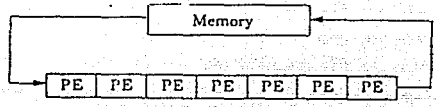


Figura 6.1 Configuración básica de un arreglo sistólico.

Como se desprende de lo anterior, pueden existir varias definiciones de lo que se entiende como un arreglo sistólico, sin embargo, a fin de dar una definición coherente con lo planteado hasta aquí, conviene adoptar el siguiente enunciado:

Un arreglo sistólico es una red de elementos de procesamiento que posee las características siguientes [KUNG88]:

Sincronía. Los datos son rítmicamente procesados y pasados por el sistema de acuerdo a la operación de un reloj global del mismo.

Modularidad y regularidad. El arreglo consiste de unidades modulares de procesamiento que cuentan con interconexiones homogéneas, pudiéndo así extenderse indefinidamente la red del arreglo.

Localidad espacial y temporal. Cuando el arreglo presenta una estructura de interconexión dependiente de las comunicaciones locales, entonces se tiene la localidad espacial. Por otra parte, cuando se logra al menos un retardo de una unidad de tiempo para completar las transacciones entre un nodo y el siguiente, entonces se tiene una localidad temporal.

Aceleración en el tiempo de ejecución.(Pipelinability; O(M) execution-time speedup). Cuando el arreglo cuenta con una tasa lineal de aceleración en el tiempo de ejecución, entonces se debe alcanzar un factor O(M) mejor en términos de la rapidez del procesamiento, siendo M la cantidad de elementos de procesamiento del sistema; en este caso, la eficiencia del arreglo se mide de acuerdo a la expresión:

$$\text{Factor de aceleración} = O(M) = \frac{T_S}{T_P} \qquad (47)$$

siendo Ts el tiempo de procesamiento en un procesador simple y Tp el tiempo de procesamiento en el arreglo de procesadores.

Ahora bien, a fin de sentar las bases para efectivamente diseñar un sistema que cuente con un arreglo de procesadores, orientado a manejar los datos en forma sistólica, es conveniente considerar que dichos sistemas habitualmente cuentan con cuatro componentes esenciales mismos que son [KUNG85]:

- La computadora maestra (también llamada Host)
- El subsistema de interfase que incluye la memoria de interfase y una unidad de control.
- La red de conexiones inluyéndose las conexiones entre los mismos elementos de procesamiento, así como las conexiones entre dichos elementos y la memoria.
- El arreglo de procesadores que comprende al total de elementos de procesamiento con memoria local.

Considerando la importancia que para el presente estudio tienen los cuatro elementos antes descritos, concernientes a un sistema en arreglo, serán abordados cada uno de ellos con un poco más de detalle a continuación.

La computadora principal (Host Computer).

La computadora principal debe proveer al sistema del almacenamiento masivo de datos, así mismo se encarga de la administarción y el formateo de la información. De igual forma este elemento determina la sincronía del programa encargado de controlar tanto al subsistema de interfase como a la red de conexión, además de generar y cargar los códigos que sean requeridos por cada elemento de procesamiento. Es oportuno mencionar que es una tarea difícil identificar la computadora principal que sea adecuada para un arreglo de procesadores rápido [KUNG85].

Subsistema de interfase.

El subsistema de interfase, que se conecta a la computadora principal por su canal o bus de comunicación (Host bus), tiene la función de enviar o recibir los datos que se van a procesar o ya se procesaron dentro del arreglo. Habitualmente este subsistema consta de un controlador y memoria de interfase (buffer) que facilitan el intercambio de información y datos. Por su parte, el controlador monitorea de acuerdo a la sincronía de un programa, tanto al mismo subsistema como al arreglo de procesadores. De igual forma, el subsistema de interfase se encarga de proveer un adecuado apoyo a la circuitería electrónica, a fin de soportar muchas de las operaciones comunes para la administración de los datos. También aquí cabe señalar que otra tarea difícil al momento de diseñar un sistema, es administrar los bloques de datos, asegurándose de que la memoria esté disponible a fin de balancear el reducido ancho de banda del sistema para operaciones de entrada/salida, junto con el gran ancho de banda de los procesadores en arreglo [KUNG85].

Red de conexiones.

La red de conexiones provee al sistema del conjunto de rutas que existen entre los procesadores y los módulos de memoria, con el propósito de satisfacer ciertas necesidades de comunicación. Al incorporar ciertas interconexiones estructuradas de acuerdo a un determinado mapa, se puede mejorar significativamente el rendimiento en la rapidez del arreglo [KUNG85].

Arreglo de procesadores.

Por facilidad, generalmente se considera sólo un arreglo dentro del sistema; sin embargo, contar con varios arreglos en un mismo sistema resulta atractivo actualmente. Lo anterior se traduce, por ejemplo, en que cuando un problema puede ser reducido a varios subproblemas que pueden ser ejecutados uno después del otro, sería entonces útil que cada subproblema se realizara en su propio arreglo de procesadores, por lo que se utilizaría la red para facilitar el efecto pipeline entre arreglos. Lo anterior sugiere un esquema pipeline al nivel de arreglos, lo que aceleraría el procesamiento notablemente [KUNG85].

La figura 6.2 muestra la forma en la que habitualmente se relacionan cada uno de estos cuatro elementos tanto internamente como con la computadora principal, a fin de poder soportar un sistema tipo sistólico.

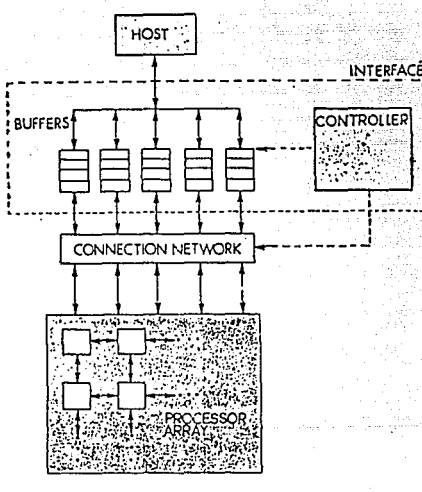


Figura 6.2 Estructura básica de un sistema en arreglo.

A partir de todo lo tratado hasta este punto, es posible proceder a limitar y especificar el tipo de sistema al cual se orinetarán las secciones restantes del presente estudio, lo que se hará a continuación.

VII. CONSIDERACIONES PREVIAS AL DISEÑO DE UNA ARQUITECTURA APTA PARA EL CALCULO DE LA DCT BIDIMESIONAL.

A partir del marco teórico establecido en las secciones previas, es posible esbozar las principales características que deben estar presentes en el diseño por realizar y de acuerdo al objetivo originalmente planteado para este estudio. Por tal motivo, es necesario considerar que para especificar el diseño del sistema, éste se aplicará al tratamiento de imágenes digitales que cuenten con un formato matricial de 256 renglones por 256 columnas, donde cada pixel cuenta con 8 bits en paralelo. Resulta oportuno aclarar que por restricciones operativas impuestas por la DEPFI para otros proyectos semejantes, cada imagen a transformar se fraccionará en submatrices que cuenten con 16 renglones, 16 columnas y 8 bits por pixel para el caso directo ó 12 bits por pixel para el caso inverso; por lo cual en lo que resta del presente estudio nos limitaremos exclusivamente al tratamiento de imágenes cuya información se encuentra en matrices de orden 16x16, con elementos de 8 ó 12 bits.

Por otra parte, considerando que se pretende diseñar un sistema digital que posea una alta eficiencia tanto en los aspectos temporal como espacial, y con el propósito de que éste pueda ser implantado para un ambiente donde se utilicen computadoras de tipo personal, entonces resulta claro entender que el algoritmo seleccionado para el cálculo de la DCT es el desarrollado por Hou [HSIE87], el cual, como se recordará de la sección II.2.2, es el antecedente para la implantación en dispositivos VLSI del algoritmo conocido como MSDCT. Resulta oportuno resaltar el hecho de que el algoritmo de Hou es el que han seleccionado algunos fabricantes de circuitos integrados puesto que, en su versión MSDCT, permite desarrollar una estructura interna que hace posible la obtención de las transformadas DCT e IDCT exactamente con el mismo dispositivo; lo que redunda en una arquitectura más sencilla y más rápida que otros esquemas de transformación.

Adicionalmente a lo anterior, y conforme a lo descrito durante la sección III, conviene recomendar el uso de procesadores de aritmética distribuida, en virtud de su inherente rapidez para efectuar operaciones repetitivas de cálculo aritmético como las que se requieren para la obtención de la DCT bidimensional, tanto directa como inversa.

Con base en lo mencionado en estos últimos dos párrafos junto con lo descrito en la sección V del presente estudio, se llega a la conclusión que el elemento central para el sistema por diseñar es, sin lugar a dudas, el procesador STV 3200 fabricado por INMOS, del cual se encuentran sus especificaciones técnicas en el apéndice I. Este dispositivo, aparte de utilizar el algoritmo de Hou para el cálculo de la DCT, tiene la característica de contar con un banco de 16 procesadores de aritmética distribuida, dispuestos en paralelo conforme a un arreglo de procesadores tipo pipeline; por lo cual se considera que el algoritmo MSDCT es un algoritmo recursivo y localmente independiente, según lo establecido en la sección VI de este escrito.

Otro aspecto importante de este dispositivo está en el hecho de que al desactivar momentáneamente su reloj principal, se suspende momentáneamente la ejecución del algoritmo de transformación, sin pérdida de información; por lo cual, si es seleccionada adecuadamente la arquitectura del sistema, la tasa de pipeline que se establece en la sección VI.1 puede ser considerablemente grande.

Ahora bien, y como se mencionó al inicio de esta sección, las imágenes a tratar están en un formato matricial de 256 renglones por 256 columnas, con 8 bits por pixel, lo que se traduce en la necesidad de destinar 64 Kbytes de memoria principal para el almacenamiento de una sola imagen digital. Considerando que el sistema por diseñar está contemplado para interactuar con computadoras personales, éste deberá ser capaz de explotar al máximo posible el uso tanto de la memoria principal de dichas computadoras, como las bondades de sus canales de comunicación (presentes en las ranuras de expansión). Esta dos últimas restricciones no deberán interferir ni con la operación de otros dispositivos instalados en la misma computadora, ni obstruir la ejecución de algún procedimiento propio del software de esas computadoras.

Con base en este último párrafo, y considerando lo descrito en la sección VI.1, en lo que se refiere al máximo paralelismo y al balance que debe existir entre los cómputos, las comunicaciones y la memoria, se llega a concluir que la mejor manera de intercambiar los datos entre la computadora personal y el sistema por diseñar es la técnica de *Acceso Directo de Memoria*, mejor conocida como DMA. Esta técnica, además de ser lo suficientemente rápida como para alcanzar la tasa de pipeline que demanda el STV 3200, permite desarrollar programas más simples para su administración operativa, como se verá más adelante.

Por otra parte, después de analizar las características del STV 3200 en lo relativo a la operación de su pipeline, junto con las arquitecturas más comunes para los procesadores en paralelo dispuestos en un arreglo tipo red, y considerando lo descrito en la sección VI.2, se llega a la conclusión que la arquitectura que mejor aprovecha el pipeline del STV 3200, es una arquitectura sistólica, misma que debe ser controlada por un algoritmo que eficientemente le administre los datos que requiera, lo que se logra con cierta sencillez en una estructura basada en el uso de la técnica de DMA.

Es importante mencionar que, dadas las características del equipo con el que se cuenta en los laboratorios de la DEPFI, se contempla como el Host de la arquitectura sistólica a una computadora personal del tipo AT, en tanto que el subsistema de interfase cae en lo que se conoce como una tarjeta de expansión prototipo para esa clase de computadoras. En lo que se refiere a la red de conexiones, ésta se construirá con diversos dispositivos integrados que permitan un adecuado control del STV 3200, ya que éste es considerado como el arreglo de procesadores propio de la arquitectura sistólica en cuestión.

Como se desprende de lo anterior, los puntos tratados en esta sección delimitan el diseño del sistema destinado a soportar la mejor implantación de la DCT bidimensional en medios electrónicos, por lo cual en la siguiente sección es posible proceder a relizar dicho diseño.

VIII. DISEÑO DE UNA ARQUITECTURA PARA EL CALCULO DE LA DCT BIDIMENSIONAL SOBRE DATOS DIGITALES EN FORMATO MATRICIAL.

Con el propósito de establecer claramente los lineamientos que regirán durante esta sección, es conveniente señalar que la técnica de diseño seleccionada es la que se conoce como *Diseño Funcional Descendente* (también conocida como TOP-DOWN), misma que se apega estrictamente a los principios de modularidad, regularidad, conectividad y localidad propios de todo diseño de sistemas con microprocesadores.

Ahora bien, una vez que se ha establecido en la sección anterior que la parte medular del sistema es precisamente el dispositivo STV 3200, y antes de continuar con el diseño, se hace necesario especificar la organización de memoria que requiere este dispositivo, para lo cual considérese que una imagen digital por transformar (una imagen a la cual se le va a aplicar la DCT bidimensional directa) es almacenada, mediante 64 KB, en un formato matricial de 256 renglones por 256 columnas; si se toma a ésta como una hipermatriz formada a su vez por submatrices de orden 16x16, entonces se llega a que dicha hipermatriz contará con 256 submatrices diferentes. De acuerdo a lo anterior, cada renglón y cada columna de la hipermatriz constarán de 16 submatrices diferentes, dispuestas como se muestra en la figura 8.1, donde se puede apreciar la localización de las diversas submatrices de orden 16x16.

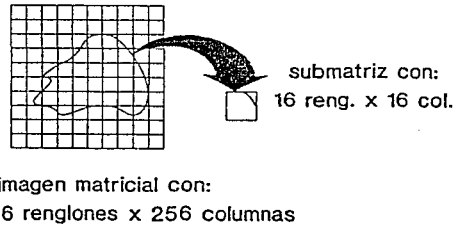


Figura 8.1 Segmentación de una imagen digital en submatrices.

Por otra parte, si para el caso directo se considera que cada submatriz de 16x16 se organiza en la memoria de la computadora personal como un conjunto de 256 localidades de 8 bits cada una de ellas, estando ubicadas en direcciones contiguas, entonces resulta explicable el hecho de que cada renglón de la hipermatriz representa un bloque de memoria de 4 KB. Sin embargo, al momento de revisar las especificaciones que hace INMOS para las memorias que operan con el STV 3200, se llega a la conclusión que se requiere una memoria del tipo estático, de alta velocidad y fácil manejo; para lo cual este fabricante recomienda su memoria IMS 1635, misma que cuenta con una organización de 8Kx8 y un tiempo máximo de respuesta igual a 15, 20 ó 25 nanosegundos, dependiendo

del modelo seleccionado. En caso que se desee operar el STV 3200 a 16 MHz, se recomienda el modelo de 15 nanosegundos. La información técnica de esta memoria se puede consultar en el apéndice II del presente estudio.

Es necesario destacar el hecho de que, de acuerdo a lo visto en secciones previas, el tiempo de acceso de las memorias que apoyan a un arreglo de procesadores en paralelo es un factor de capital importancia para alcanzar la tasa de pipeline propuesta para dicho arreglo, puesto que de ese tiempo depende se formen o no los cuellos de botella de las comunicaciones locales e intramodulares; lo que puede redundar en que se degrade notablemente el tiempo de procesamiento de todo el arreglo.

Además, en este caso el uso de la memoria IMS 1635 permite captar en un solo dispositivo, dos renglones consecutivos de la hipermatriz que contiene a la imagen original en tratamiento o, lo que es lo mismo, se pueden caputar en el mismo circuito hasta 32 submatrices consecutivas, siendo cada una de ellas de orden 16x16 y 8 bits por pixel, para el caso de la transformación directa (DCT). Para el caso de la transformación inversa (IDCT), es necesario utilizar dos dispositivos de memoria puestos en paralelo, a fin de retener simultáneamente los 12 bits que constituyen a cada pixel.

A partir de esto último, y considerando además que el sistema se pretende implantar en una tarjeta de expansión para computadoras personales tipo AT (por lo cual existen restricciones de espacio disponible), se llega entonces a la conclusión de que solamente existirán dos bancos de memoria para retener los datos que requiere o genera el STV 3200. El primer banco será de 8Kx8, en tanto que el segundo banco será de 8Kx16. Cada uno de estos bancos estará formado por memorias IMS 1635 conforme a lo ya comentado.

De acuerdo con lo tratado hasta aquí, es posible plantear un diagrama que resuma el flujo de la información que será manejada por el sistema que se está diseñando. Dicho diagrama se muestra en las figuras 8.2a y 8.2b, las cuales corresponden, respectivamente, al caso de la transformada directa y la transformada inversa.

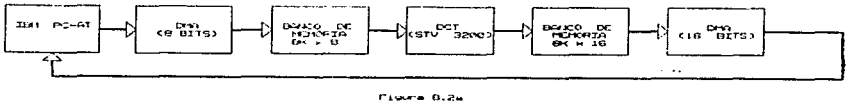


Figura 8.2a

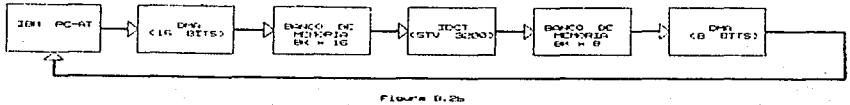


Figura 8.2b

Figura 8.2 (a)Flujo que sigue la información para la DCT.
(b)Flujo que sigue la información para la IDCT.

Como ya se ha mencionado, el sistema por diseñar se pretende implantar en una tarjeta de expansión para computadoras tipo AT, motivo por el cual la arquitectura que se seleccione deberá tomar en cuenta las características operativas que impone IBM a tales tarjetas, entre las que se encuentran la dirección de habilitación y el método de comunicación con la misma PC. En este caso, IBM facilita en su *Manual de Referencia Técnica* (Technical Reference Manual) información sobre la tarjeta prototipo que se debe de emplear para la clase de computadoras seleccionada; dicha información puede ser consultada en el apéndice III. En lo que respecta a nuestro proyecto, 300H se define como la dirección con la que se identificará al registro de control de la tarjeta prototipo dentro del mapa de puertos de la computadora personal, en tanto que 302H se define como la dirección para el registro de datos ; la comunicación que se establece entre la PC y la tarjeta prototipo es en paralelo, de acuerdo a alguno de los esquemas de DMA disponibles en la misma computadora personal. Adicionalmente a esto último, se toma como política de diseño, adoptar medidas para que el sistema aquí desarrollado cuente con un bit de habilitación para que la tarjeta prototipo sólo responda a comandos de control en la dirección 300H del mapa de puertos, considerando, de acuerdo a lo ya expuesto, que esta localidad es un registro de control; en tanto que se considera que la localidad 302H corresponde a la dirección propia para el manejo de datos que existan entre la PC y el sistema en cuestión.

La información de estos últimos párrafos nos permite esbozar un diagrama a bloques que refleje la organización básica del sistema destinado a soportar el cálculo de la DCT bidimensional en medios electrónicos, mismo que se muestra en el figura 8.3

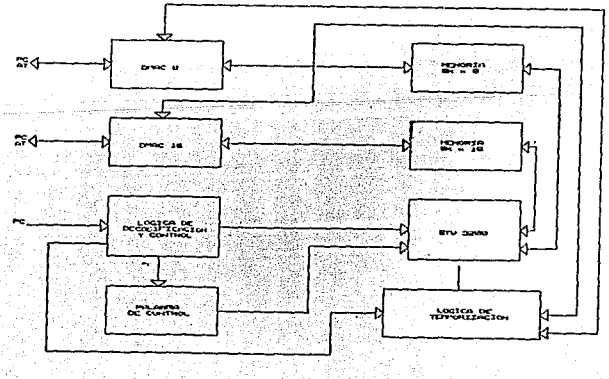


Figura 8.3 Diagrama a bloques general del sistema para el cálculo de la DCT en medios electrónicos.

Por otra parte, y dado que la tecnica de DMA tiene gran importancia dentro de este proyecto, conviene abordarla con mas detalle a continuación. La técnica de DMA se puede aplicar dentro de la arquitectura de una computadora personal, con el propósito de apoyar las transferencias de datos entre la memoria y algún dispositivo periférico de gran rapidez. Al utilizar esta técnica en una computadora personal convencional, es posible transferir datos a una tasa de 476 kilobytes por segundo, con lo cual se aprovecha aproximadamente la mitad del ancho de banda disponible en el canal de la PC. Es oportuno mencionar que este tipo de transferencias de alta velocidad tienen como consecuencia natural el hecho de que se degrade el tiempo de ejecución del programa actual en la PC, llegándose a consumir hasta la mitad del tiempo promedio de ejecución cuando no existen esta clase de accesos. Por otra parte, cabe añadir que los ciclos de DMA de una computadora personal, están diseñados de tal manera que no se puede dejar bloqueado al microprocesador maestro de la PC.

Adicionalmente a lo anterior, es importante mencionar el hecho de que un ciclo de DMA involucra las operaciones de lectura y escritura tanto de memoria como de puertos durante el mismo ciclo; además, los canales de la PC no son controlados por el microprocesador principal, sino por un controlador de DMA y algunos dispositivos que lo apoyan. Cabe señalar que un acceso de DMA en una computadora personal, no es iniciado ni por su procesador central ni por su controlador de DMA, sino por la solicitud que hace algún dispositivo periférico mediante las señales que se encuentran disponibles en la ranura de expansión y que se denominan como DRQi (donde i=1,2,...,7). Un dispositivo periférico realiza esta solicitud de un ciclo de DMA, mediante la transición positiva (flanco positivo) que tenga lugar en alguna de las señales DRQi. Una vez que se ha efectuado la solicitud para un acceso de DMA, entonces el periférico solicitante reconoce que se inicia el ciclo cuando recibe de la computadora personal, mediante la señal DACKi (donde i=1,2,...,7 respectivamente), una transición negativa (flanco negativo).

Con base en la información proporcionada por IBM, una computadora personal del tipo AT cuenta con siete canales para soportar accesos de DMA; sin embargo, dichos canales son controlados mediante dos circuitos controladores 8237A-5, marca INTEL, donde cada dispositivo se encarga de atender hasta cuatro canales. La información que proporciona IBM sobre los ciclos de DMA es la que está en su *Manual de Referencia Técnica*, del cual se encontrarán copias en el apéndice IV del presente estudio. Asimismo, y con base en dicha información, se llega a la conclusión que para los fines que persigue este estudio, deben ser seleccionados los canales 0 y 5 para soportar, respectivamente, las transferencias de 8 y 16 bits que requiere el sistema para el cálculo de la DCT y la IDCT bidimensionales; obedeciendo dicha selección al hecho de que esos canales son los de más alta prioridad de acuerdo a la cantidad de bits en paralelo que se manejen en cada caso.

Cabe mencionar que, de acuerdo al esquema de una arquitectura sistólica, el módulo que administra los accesos de DMA cae en lo que correspondería a la red de interconexiones que requiere dicha arquitectura.

Ahora bien, con base en lo anterior es posible efectuar el diseño de los módulos que se ubican en la tarjeta prototipo y que son los encargados del manejo de las señales de activación de los ciclos de DMA de la computadora personal. Se debe considerar que dichos modulos estan especializados dependiendo de la operación que se pretenda realizar; por lo cual existen dos módulos para solicitar la escritura de datos en la memoria de la PC (Un módulo se encarga de la escritura por el canal 0 que es de 8 bits, en tanto que el otro módulo se encarga de la escritura por el canal 5 que es de 16 bits); existen, además, dos módulos de lectura que le solicitan datos a la computadora personal (Un módulo se encarga de la lectura de datos de la PC por el canal 0, en tanto que el otro de los módulos se encarga de la lectura de datos por el canal 5 del la misma PC).

Es importante tener en cuenta para el diseño de esos cuatro módulos, que el ciclo de DMA lo inicia la tarjeta prototipo mediante el flanco de una transición bajo-alto en la señal DRQi respectiva; además, dichos módulos deben considerar el comportamiento de la señal DACKi que genera la misma computadora personal. Por otra parte, también se debe contemplar que los canales por los cuales realmente circulan los datos, deben estar debidamente acoplados al canal de la ranura de la PC, a fin de evitar posibles problemas de sincronía o disparidad de impedancias entre la computadora personal y la tarjeta prototipo. Asimismo, se debe considerar, conforme a lo ya mencionado, que la tarjeta prototipo, y por ende sus módulos de interfase, se activan dependiendo del estado de un bit de habilitación propio de la dirección y puerto en uso.

Las figuras 8.4a y 8.4b respectivamente muestran los diagramas de los circuitos que pueden ser utilizados como módulos de interfase para la lectura de datos de 8 ó 16 bits, provenientes de los canales 0 y 5 de la computadora personal para el manejo de DMA. En este caso, la interfase se activa por un bit de habilitación propio, por lo que el canal de la PC puede ser reutilizado por otros dispositivos cuando este módulo no está en operación. A fin de transmitir un dato de uno o dos bytes según sea el caso, y asumiendo que los dispositivos encargados del manejo del DMA han sido previamente programados en forma adecuada, la interfase simplemente aplica el dato al puerto de entrada respectivo y hace una transición positiva en la señal de solicitud DRQi. Cuando ocurre esa transición, quedan fijos los datos de entrada en el registro de datos del canal, solicitándose entonces un ciclo de DMA. En el momento en que la respectiva señal DACKi toma un estado activo, se transfiere un dato a la memoria del sistema de la computadora personal, borrándose la solicitud de DMA hecha mediante la señal DRQi (Es decir, se le da *reset* a la señal de *request*). Estas acciones se repetirán cada vez que la señal DRQi sufra una transición positiva y mientras la cuenta terminal del DMA no se haya alcanzado.

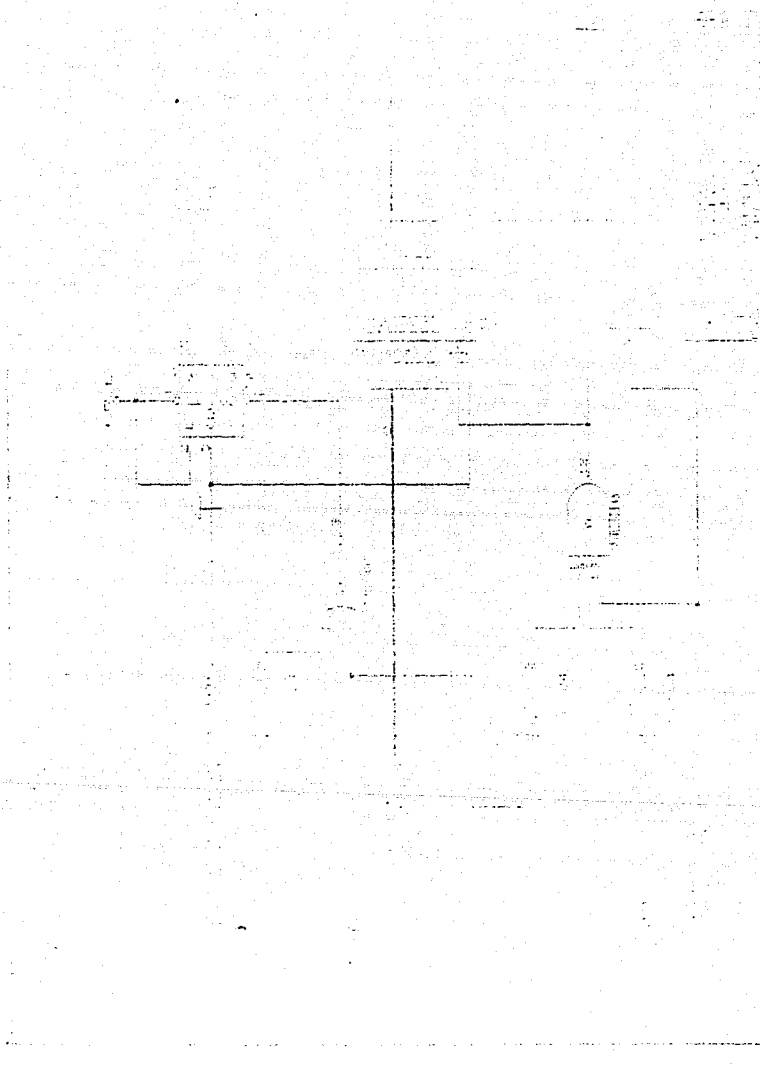


Figura 8.4a Circuito básico para transferir, mediante DMA, datos de 8 bits de la tarjeta prototipo a la PC.

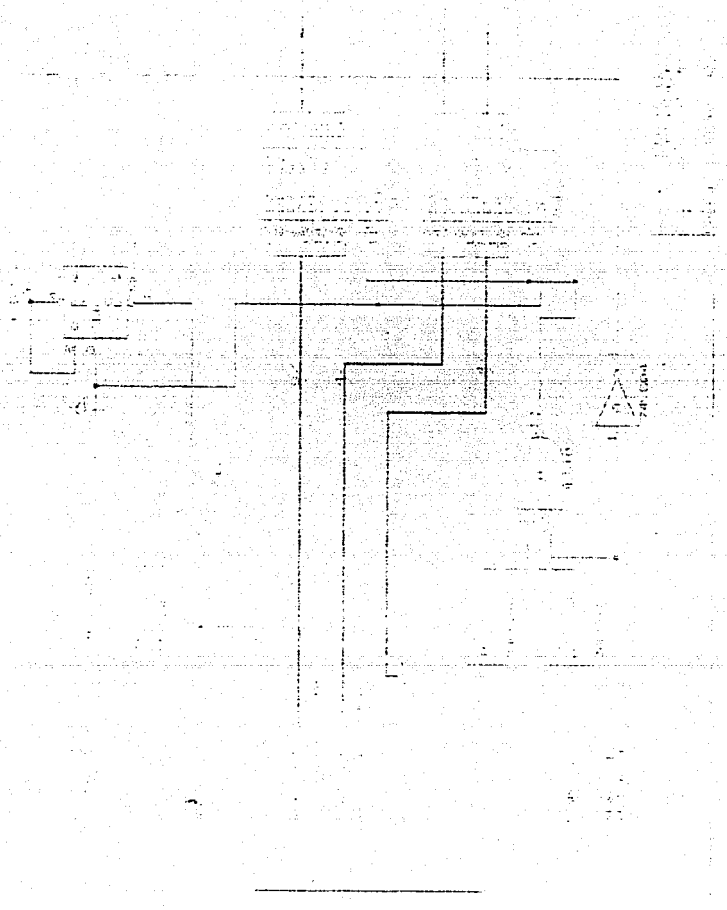


Figura 8.4b Circuito básico para transferir, mediante DMA, datos de 16 bits de la tarjeta prototipo a la PC.

Por su parte, las figuras 8.5a y 8.5b muestran, respectivamente, los diagramas de los circuitos que pueden ser empleados en los módulos que bajo el control del DMA, pueden transferir datos de 8 ó 16 bits provenientes de la memoria de la PC y que están destinados a ser utilizados en la tarjeta prototipo. Para solicitar un dato de la memoria de la computadora personal, basta con poner en un estado alto la señal de solicitud DRQi, con lo cual serán cargados el registro del puerto en uso, al instante en el que la señal DACKi se encuentre en un estado activo. Lo anterior asume que el canal de DMA ha sido habilitado por el bit de activación respectivo, propio del puerto de entrada salida que se esté utilizando, además de considerar que los dispositivos para el manejo del DMA han sido previamente programados en forma adecuada. En este caso, al igual que en los circuitos anteriores, se forza a que la señal DACKi borre la solicitud de DMA; así mismo, la operación del circuito se repite toda vez que la señal DRQi presente un flanco positivo y no se haya alcanzado la cuenta terminal del total de datos a transferir por el canal mediante el acceso de DMA.

Resulta oportuno mencionar que para los alcances del presente estudio, y considerando que el microprocesador STV 3200 maneja datos de 8 ó 12 bits en paralelo, se ha adoptado como política de diseño forzar al valor de cero lógico el estado de los 4 bits más significativos de un dato de 12 bits, siempre que éste se encuentre codificado en la memoria de la PC mediante una palabra de 16 bits. Por tal motivo, las figuras 8.4b y 8.5b muestran ciertas conexiones a tierra en los cuatro bits más significativos del canal de datos de 16 bits.

Una vez que se han desarrollado los módulos que accionan al ciclo de DMA en la tarjeta prototipo utilizada dentro de este diseño, y considerando que en párrafos previos se han seleccionado tanto el dispositivo con el que se calculará la DCT bidimensional junto con las memorias que soportan su operación, surge entonces la necesidad de diseñar la lógica que se encargue del control tanto del STV 3200 como de los bancos de memoria que requiere dicho dispositivo dentro de la misma tarjeta prototipo en uso; por lo cual, a continuación se realizará el diseño de dicha lógica de control.

Como punto de partida para este módulo debe considerárse que las memorias habitualmente se controlan mediante un apuntador de memoria, el cual básicamente está constituido por contadores binarios que poseen la capacidad de carga o borrado de su contenido. Por tal motivo, y tomando en cuenta que existen varios tipos de contadores disponibles en lógica TTL, se seleccionó de entre todos los posibles al contador modelo 74LS193, por sus características operativas puesto que resulto ser el que mejor se apegaba a los requerimientos de diseño planteado, al momento de poder cargar o borrar su contenido, dependiendo del flanco de sus señales de control. Adicionalmente a esto, puede realizar su conteo en forma ascendente o descendente, dependiendo del estado de sus líneas de reloj. Por otra parte, este contador además de permitir la propagación de datos mediante una cascada de contadores, también permite detener la cuenta que se esté efectuando, dependiendo de las señales de habilitación que posee. En el apéndice V se pueden consultar las especificaciones técnicas de este dispositivo.

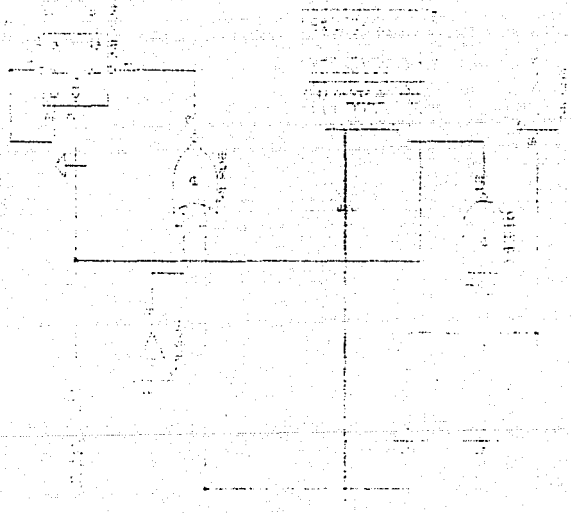


Figura 8.5a Circuito básico para transferir, mediante DMA, datos de 8 bits de la PC a la tarjeta prototipo.

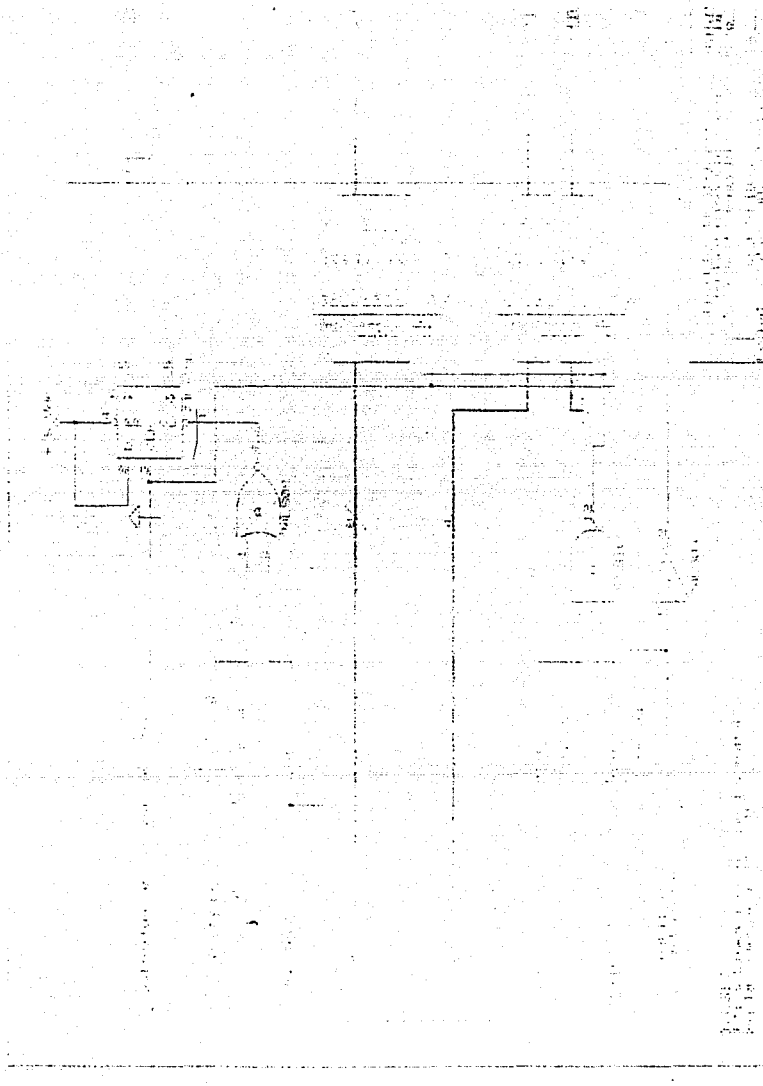


Figura 8.5b Circuito básico para transferir, mediante DMA, datos de 16 bits de la PC a la tarjeta prototipo.

Ahora bien, considerando que existen dos bloques independientes de memoria (el que maneja los datos de 8 bits y el que maneja los de 16 bits) se hace necesario que cada uno de los dos bloques posea su propio apuntador de memoria; además, tomando en cuenta que ambos bloques de memoria tienen exactamente la misma cantidad de registros, entonces se puede aplicar el principio de regularidad y diseñar los dos apuntadores que se requieren, con idéntica estructura basada en el contador 74LS193. En este caso también se debe de contemplar el hecho de que los apuntadores de memoria deben operar a dos frecuencias, una que depende de la tasa de transmisión por DMA y otra que es la propia de la operación nominal del STV 3200, es decir, 16 MHz.

Cuando se esté realizando alguna de las transferencias de tipo DMA, se recomienda tomar como señales de reloj para los contadores a las señales DACKi, puesto que ellas sólo cambian de estado cuando se ha transferido un dato entre la PC y el sistema periférico. Por otra parte, en el caso de la operación nominal de la tarjeta prototipo, ésta deberá poseer su propio reloj de 16 MHz para poder impulsar tanto a los apuntadores de memoria como al mismo STV 3200. En ese caso, la lógica encargada de apoyar a los apuntadores de memoria deberá ser capaz de efectuar las transiciones que tengan lugar entre las dos señales ya comentadas, y que actúan como relojes de los contadores utilizados.

Una característica importante que debe poseer el diseño del módulo en que nos encontrámos, consiste en su capacidad de ser borrado el contenido de los contadores toda vez que ocurra: una escritura en la localidad de puertos 300H, una transición entre los relojes de los contadores (lo que se debe entender como la transición entre un ciclo de DMA y un ciclo nominal de trabajo para el STV 3200) o la presencia de una señal que indique el reestablecimiento de las condiciones iniciales de operación (es decir, la presencia activa de la señal *RESET DRV* en el bus de la PC).

Otro aspecto que también se debe considerar para el diseño del módulo de los apuntadores de memoria, consiste en la capacidad de direccionamiento que requieren las memorias. En este caso, y como ya se ha mencionado, las memorias seleccionadas poseen localidades por un total de 8K, por lo cual se necesita que cada apuntador de memoria cuente con 4 contadores 74LS193, a fin de cubrir el total de direccionamiento. En la figura 8.6 se puede apreciar la estructura básica con la que se diseñaron los módulos apuntadores de memoria, mismos que cubren las especificaciones planteadas en los párrafos previos.

Por otra parte, y como se desprende de analizar la información técnica del STV 3200, existe una diferencia de tiempo entre el instante en que el STV 3200 recibe los datos para aplicarles alguna de las transformadas (DCT o IDCT), y el instante en que inicia la salida de los datos ya transformados. Dicha diferencia se calcula conforme a una expresión dada por el fabricante y, para el caso en que nos encontramos, se traduce en 386 pulsos de reloj principal (es decir 386 pulsos del reloj que opera a

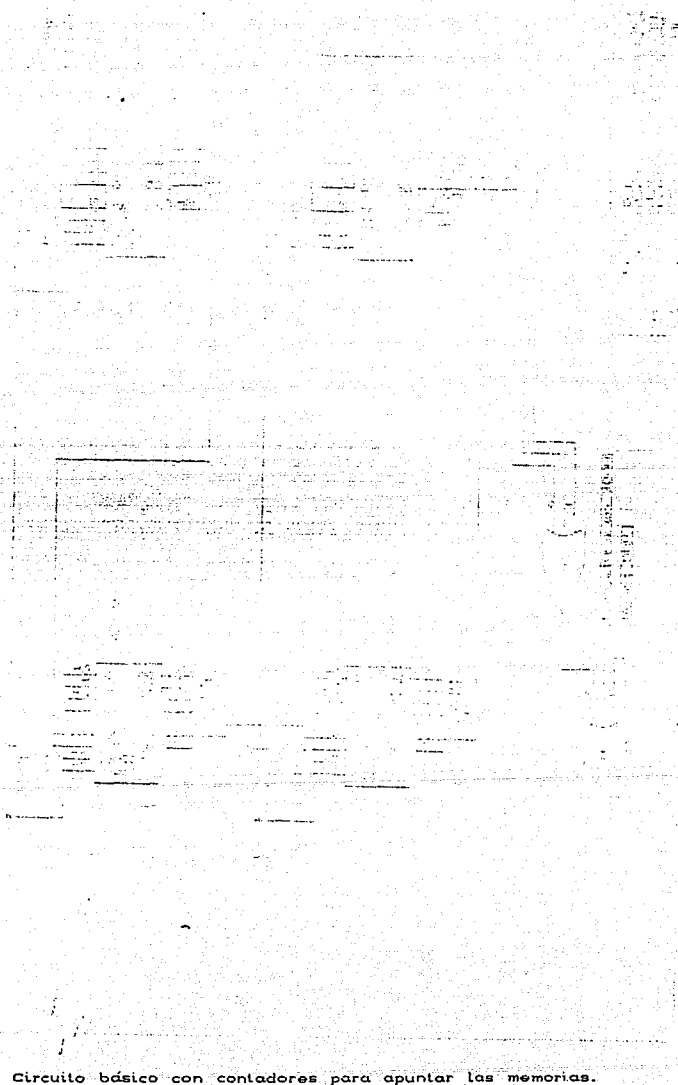


Figura 8.6 Circuito básico con contadores para apuntar las memorias.

16 MHz). Con el propósito de poder sincronizar adecuadamente la ubicación de las localidades respectivas en los dos bancos de memorias, se requiere consumir, mediante contadores, la diferencia de los 386 pulsos de reloj. Por tal motivo, es necesario incluir una lógica que se encargue de manejar a estos contadores auxiliares. En la figura 8.7 se puede apreciar el diagrama propio de dicha lógica. Cabe señalar que se sigue explotando en este circuito el principio de regularidad en el diseño de los módulos operativos, puesto que, como se aprecia en la figura 8.7, se continua centrando el diseño en contadores 74LS193.

Colateralmente al diseño de los apuntadores de memoria, y con el propósito de apoyar a los módulos encargados de las transferencias de DMA entre la PC y la tarjeta prototipo, surge la necesidad de definir un registro en la localidad 300H, que haga las veces de un registro de control para el STV 3200 puesto que este dispositivo no cuenta con un registro propio que el usuario pueda afectar externamente. Se decidió la dirección 300H por ser la primera con la que se activa la tarjeta prototipo dentro del mapa de puertos de la computadora personal. En este caso, y después de analizar la experiencia obtenida con los contadores utilizados en otros módulos, se llegó a la conclusión de emplear para la implantación del registro de control a los mismos tipos de contadores 74LS193, en virtud de las características ya comentadas en este mismo escrito.

En cuanto a la estructura del registro de control, se decidió que esta fuera hecha con dos contadores 74LS193, con el propósito de poder soportar hasta 8 condiciones diferentes simultáneamente. Para controlar de la línea de carga de los contadores, se decidió combinar mediante una lógica a las señales IOW, SA0, SA1, SA2, SA3 y SA4, las cuales están presentes en el bus de la computadora personal. En cuanto a las líneas SDi (i=0,...,7) se decidió conectarlas a las entradas de datos de los contadores, conforme se muestra en la figura 8.8. En esta misma figura se muestra como se conectaron las salidas de los contadores al resto del sistema diseñado. Es interesante notar que los contadores fueron inhibidos para el conteo descendente; sin embargo, para el conteo ascendente, se optó por coordinar a cada uno de los dos contadores con las señales respectivas DACK0 y DACK5. La principal razón para haber planteado esta opción es poder forzar el cambio de estado en alguna de las señales de control que se generan a partir de las salidas de los contadores que definen al registro de control. Por otra parte, la figura 8.8 ilustra cómo se controla mediante este registro la operación del STV 3200, en particular en lo que se refiere al cálculo de la transformada directa o inversa.

Como se desprende de analizar la figura 8.8, el registro de control es realmente un ardid electrónico simple, que está pensado tanto para retener temporalmente ciertas señales, como para impulsar el cambio de estado de otras; sin embargo, su simpleza no debe engañar puesto que es una de las partes medulares para el control operativo del STV 3200.

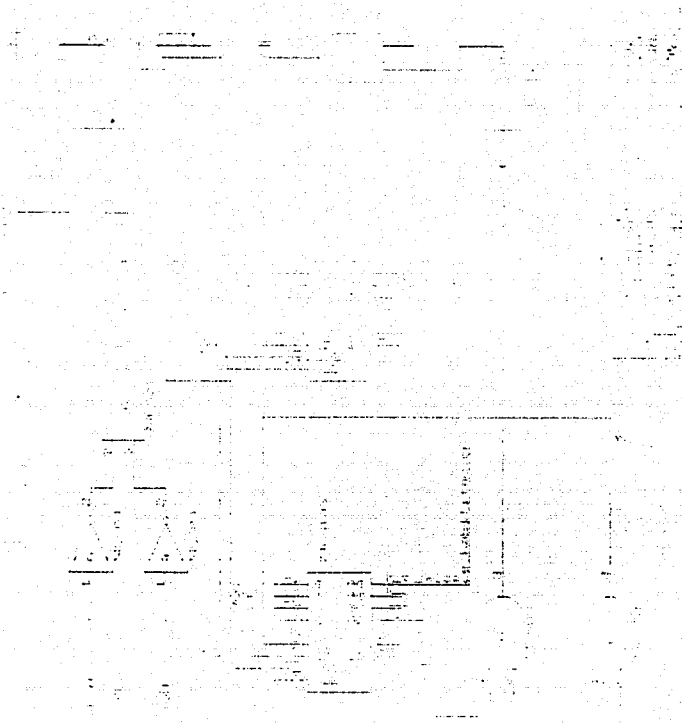


Figura 8.7 Circuito básico con contadores para el retardo de tiempo.

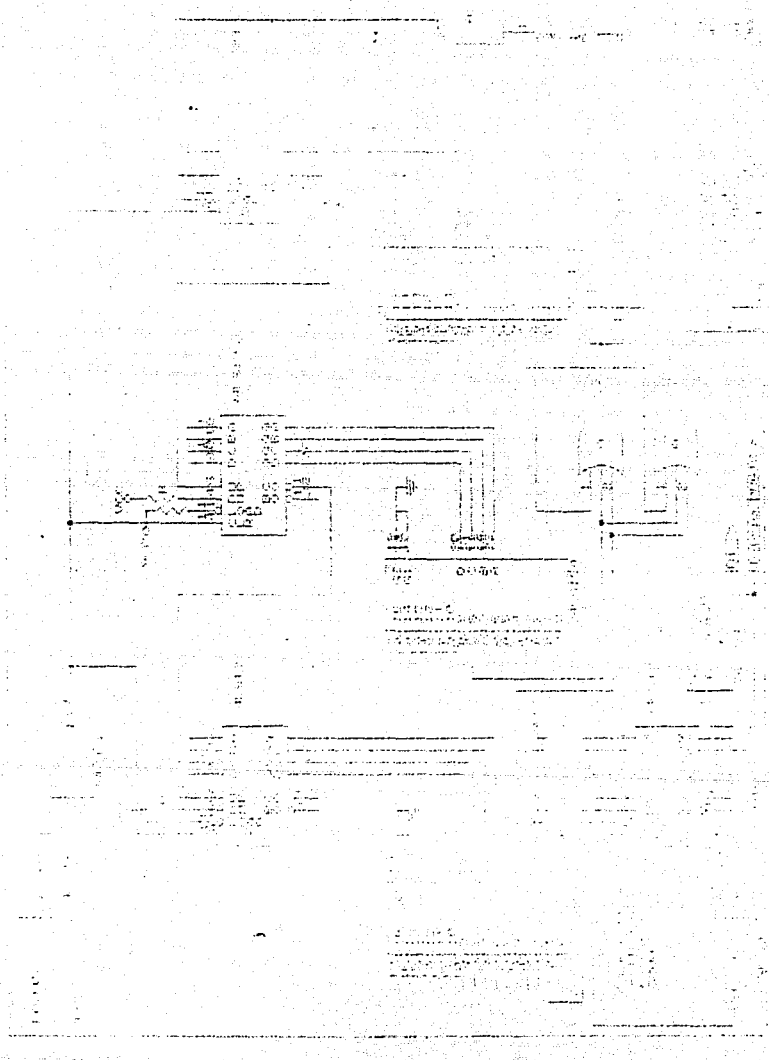


Figura 8.8 Circuito básico con contadores para emular un registro de control.

Despues de haber discutido cada uno de los módulos que constituyen al sistema propuesto y que esta orientado al cálculo de la DCT bidimensional, se;llega entonces a que la arquitectura global es la que se muestra en la figura 8.9, la cual puede obtener las transformadas DCT para el caso directo e inverso. Cabe recordar que, con base en lo expuesto en secciones previas, esta es una arquitectura sistólica que optimiza el cálculo recursivo requerido para la transformación de imágenes digitales bidimensionales.

Por otra parte, y en lo que se refiere al software externo que requiere para su operación la arquitectura propia de este sistema digital, se puede afirmar que éste depende de la aplicación que se esté trabajando en el momento y, en todo caso, las únicas necesidades que existen desde el punto de vista de programación, son: definir adecuadamente las páginas de memoria involucradas con las transferencias de tipo DMA conforme lo indica IBM en su *Manual de Referencia Técnica*; definir el contenido del registro de control del STV 3200, a fin de establecer tanto el sentido de la transformación, la selección del reloj de operación y la habilitación de la tarjeta prototipo para su dirección de datos (302H).

No Hay Hoja.

66, 67

IX. EVALUACION DE LA ARQUITECTURA PROPUESTA.

A fin de establecer un marco de referencia que permita evaluar la arquitectura propuesta durante las secciones previas, conviene partir del análisis de los resultados obtenidos por Moctezuma y García en los trabajos que realizaron para la DEPFI sobre el tratamiento de imágenes digitales. En esos trabajos, Moctezuma y García utilizaron como una herramienta más para la consecución de sus objetivos, técnicas de software para efectuar el cálculo de las transformadas DCT e IDCT; debido a lo complicado y tardado de los cálculos que se requerían para evaluar directamente dichas transformadas, se plantearon dos estrategias para evaluar la DCT y la IDCT, una mediante lo que llamaron el método MCI y otra mediante lo que denominaron el método MCR; dados los recursos con los que contaban y el giro de su proyecto, finalmente adoptaron el método MCI para su investigación.

El método MCI propuesto por Moctezuma y García para la obtención de la DCT y la IDCT, recibe su nombre en virtud de ser un Método de Cálculo Indirecto (de ahí el MCI) basado en técnicas ya conocidas para la obtención de la FFT, centradas en la que se conoce como Split-Radix FFT. Por su parte, el método MCR se refiere al Método de Cálculo Recursivo, el cual no es otra cosa que la implantación del método de Hou por técnicas de software. Cabe señalar que para la ejecución de ambos algoritmos, se utilizó un microprocesador para DSP como lo es el TMS320C25, considerando que se le destina un tiempo de 100 nanosegundos a la ejecución de cada instrucción de tal procesador [GARC90].

Con base en las premisas expuestas, Moctezuma y García obtuvieron los siguientes resultados al transformar una imagen digital que se encontraba en un formato matricial de 256x256 y con 8 bits por pixel [GARC90].

	Unidad	Método MCI		Método MCR	
		DCT	IDCT	DCT	IDCT
Longitud de las operaciones	bits	32	16	32	32
Memoria de programa	palabras	595	640	608	608
Memoria de datos	palabras	272	224	4287	4287
Tiempo de ejecución por imagen.	segundos	3.69	3.8	9.17	9.17

Resulta oportuno señalar que estos resultados fueron obtenidos una vez que ya se habían proporcionado los datos al procesador TMS320C25, requiriendo este procesador de su propio programa, el cual les consumía entre 595 y 608 palabras de memoria de programación. Por otra parte, también les fue necesario expander los datos desde 8 bits por pixel hasta 32 bits, según se muestra en la tabla anterior; requiriéndose también de una memoria auxiliar para datos intermedios, misma que variaba de 224 a 4287 palabras.

Con lo anterior se tienen suficientes datos para evaluar algunos parámetros de la arquitectura diseñada durante el proyecto ''Nepohualtzintzin''. Como punto inicial, conviene partir del tiempo que consume depositar datos en la tarjeta prototipo mediante la técnica de DMA. En este caso, se debe recordar que el diseño de la IBM-PC considera normalmente 6 ciclos de reloj para ejecutar completamente un ciclo de DMA. Si se toma en cuenta el caso típico para una computadora tipo AT, donde el reloj es de 12 MHz, entonces un ciclo de DMA consume, en promedio, 500 nanosegundos (Este tiempo varía dependiendo del reloj maestro de la PC). Si durante un ciclo de DMA se transfiere tan solo un dato que puede ser un byte o una plabra de 16 bits, entonces para transferir los 8K que requiere este diseño, se consumen 4.096 milisegundos.

A partir de este último párrafo, podemos estimar el tiempo que requiere la arquitectura aquí propuesta, para obtener la DCT bidimensional sobre datos o pixels de 8 bits. En este caso, y considerando que transferir datos entre la PC y el sistema consume un tiempo igual a 4.096 milisegundos, entonces una vez efectuada esta transferencia el procesador STV 3200 comienza a operar a 16 MHz, por lo cual aumenta la rapidez del sistema en forma global. Ahora bien, considerando la información del apéndice I, el STV 3200 consume 642 ciclos de su reloj (el cual oscila a 16 MHz) para dar el último dato ya transformado de una matriz de 16x16 que se le haya suministrado; tal matriz se interpreta físicamente como una cadena ordenada en forma consecutiva y con 256 caractéres. A 16MHz, esos 642 ciclos se traducen en 40.125 microsegundos, con lo cual se puede afirmar que ese es el tiempo que consume transformar en forma directa una submatriz de orden 16x16 y con 8 bits por pixel. Sin embargo, las memorias que se tienen son de 8K de capacidad, lo que implica que se pueden almacenar exacta y simultáneamente 32 submatrices de orden 16x16; en tal caso, el tiempo que consume el STV 3200 en transformar esas 32 subatrices (sin detenerse dado que son bloques consecutivos) es de 1.284 milisegundos. Resulta oportuno mencionar que durante ese tiempo la arquitectura propuesta opera a 16 MHz y no se permite el vaciado del pipeline del procesador STV 3200.

El tiempo acumulado hasta este punto del proceso para el cálculo de la DCT directa, es de 5.380 milisegundos, encontrándose los datos ya listos para ser devueltos a la PC, pero ahora en un formato de 16 bits. Para lograr lo anterior, se vuelve a recurrir a la técnica de DMA; sin embargo, gracias a las bondades de la arquitectura de la PC, se pudieron efectuar transferencias de 16 bits mediante el canal 5 del canal disponible en la PC y que habitualmente se destina a tales transferencias. En este caso, el tiempo total que se consume en la PC para lograr este ciclo de DMA, es exactamente igual que el anterior, es decir 4.096 milisegundos para 8K de palabras; por lo cual el tiempo total que consume transformar las 32 submatrices contenidas en memoria, es igual a 9.476 milisegundos.

Ahora bien, se debe recordar que una imagen digital se ha organizado en submatrices de orden 16x16, y por restricciones de las memorias en uso se deben agrupar 32 submatrices de 16x16, teniéndose por tal motivo 8 grupos que contienen al total de los datos de la imagen original (la cual se encuentra en un formato matricial de 256x256). En tal caso, y si se posee un eficiente esquema de control de los datos, de tal manera que se permita pasar instantáneamente de un grupo a otro, entonces el tiempo que consume calcular la DCT de una determinada imagen mediante la arquitectura propuesta, es tan solo de 75.808 milisegundos. Cabe recordar que para el esquema de control de DMA por software, es factible definir en la PC-AT (según lo indica IBM mediante la información que se encuentra en el apéndice IV) páginas de hasta 128K, por lo cual es posible actualizar instantánemente el apuntador del grupo respectivo de submatrices a transformar.

Por otra parte, y en lo que respecta al cálculo de la IDCT, se puede afirmar que gracias al algoritmo seleccionado en la arquitectura propuesta, el tiempo que se requiere para su obtención es exactamente el mismo que el que consumió el cálculo de la DCT directa, siendo esto una de las principales ventajas que presenta esta nueva arquitectura.

Ahora bien, con la idea de evaluar objetivamente el rendimiento del arreglo de procesadores que reside en el STV 3200, de acuerdo a lo que se menciona en la sección VI.2 del presente estudio, mediante la expresión (47), y auxiliándonos de los datos vertidos en esta sección (incluyendo los datos obtenidos por Moctezuma y García), es posible calcular el factor de aceleración OCM), estimado para esta arquitectura:

$$OCM)_{DCT} = \frac{3.69}{75.808 \times 10^{-3}} = 48.6756 \quad \text{Considerando el método MCI [GARC90]}$$

$$OCM)_{IDCT} = \frac{3.8}{75.808 \times 10^{-3}} = 50.1266 \quad \text{Considerando el método MCI [GARC90]}$$

$$CCMD_{IDCT} = \frac{9.17}{75.808 \times 10^{-3}} = 120.9635 \text{ Considerando el método MCR [GARC90]}$$

$$CCMD_{IDCT} = \frac{9.17}{75.808 \times 10^{-3}} = 120.9635 \text{ Considerando el método MCR [GARC90]}$$

Como se desprende de lo anterior, la arquitectura propuesta acelera entre 48 y 120 veces la arquitectura de un sistema centrado en el TMS 320C25. Sin embargo, también se presentan ciertas ventajas desde el punto de vista de recursos, es decir, en tanto que esta arquitectura no requiere de que se le programe en particular, la del TMS si lo requiere; adicionalmente a esto, en las operaciones que efectúa el TMS es necesario hacer ajustes de precisión, en tanto que con el STV 3200 ya se tiene fija la precisión de los cálculos, previniéndose inclusive algunas condiciones de saturación. Por otra parte, una de las desventajas de esta arquitectura es no poder variar sus parámetros, en caso de que así lo requiriera el esquema de codificación y compresión de imágenes en uso. Otra característica importante de la arquitectura propuesta es la poca cantidad de circuitos integrados que requiere ya que en su totalidad puede ser implantada en una tarjeta prototipo como ya se mencionó.

Un aspecto interesante de notar, y que es resultado del presente estudio, se da en el hecho de observar que el algoritmo de Hou es poco eficiente desde el punto de vista de software (su ejecución consume 9.17 segundos vs. 3.69 del Split-radix); sin embargo, este mismo algoritmo es muy eficiente desde el punto de vista de hardware, puesto que permite optimizar el uso del espacio disponible dentro de un dispositivo VLSI, lo cual acarrea consigo el hecho de poder implantarlo físicamente con una complejidad relativamente baja y una eficiencia bastante alta, según lo muestran los datos contenidos en el apéndice I.

Por todo lo anterior, se puede afirmar que la evaluación de la arquitectura planteada resulta favorable y estimula para contemplar el empleo de esta clase de dispositivos en futuras aplicaciones generadas por la DEPFI.

X. CONCLUSIONES Y COMENTARIOS.

Después de analizar diversos algoritmos de transformaciones lineales aplicables al tratamiento de imágenes digitales, así como estudiar variadas arquitecturas electrónicas que soportan los cálculos requeridos por dichos algoritmos, se pudo llegar finalmente a proponer un esquema que, mediante el apoyo de computadoras personales, pudiera efectuar ágilmente la *Transformada Discreta Coseno* (DCT) para el caso bidimensional. Si bien la arquitectura aquí propuesta no es la panacea del procesamiento digital de señales, si es una solución digna de analizar puesto que permite ejecutar con gran rapidez y mediante técnicas relativamente simples de hardware, un algoritmo que tradicionalmente se desarrollaba por software, apoyándose en técnicas de cálculo indirecto como la DFT o la FFT.

Es interesante resaltar que la filosofía de diseño se basó en los principios que rigen al diseño de procesadores de gran capacidad, considerando lo que es el procesamiento en paralelo, el pipelining y el comportamiento sistólico. Asimismo, resultó interesante idear a la tarjeta prototipo como un procesador especializado del cual sólo se contaba, en forma inicial, con el equivalente de su unidad de lógica-aritmética (en este caso el STV 3200); requiriéndose desarrollarle tanto su registro de control como algunos apuntadores de memoria que, en los casos de procesadores convencionales, hacen las veces del MBR.

Otro aspecto de especial relevancia es, sin lugar a dudas, el que se refiere al pipeline del procesador STV 3200 y su alta tasa de procesamiento, ya que gracias a esto fue necesario definir la arquitectura del sistema como una arquitectura sistólica, requiriéndose, por ende, hacer uso de las transferencias por DMA, entre la computadora personal y el sistema aquí desarrollado, puesto que sólo así era posible acercarse un poco a la altísima tasa de pipeline que teóricamente puede manejar dicho procesador.

Por otra parte, no se debe olvidar que uno de los propósitos del presente estudio fue analizar diferentes algoritmos de cálculo de la DCT bidimensional, a fin de establecer aquel que mejor permitiera su implantación electrónica; por lo que a raíz de estudiar diversas técnicas de transformación, tal como se hizo durante las secciones II., II.1, II.2, II.2.1 y II.2.2, se establece que los algoritmos de Hou y Modificado Simétrico (MSDCT) son los que mejores características presentan para ser implantados en medios electrónicos puesto que, debido a su estructura recursiva y buen aprovechamiento de tablas de búsqueda de datos (look-up tables), ahorran tiempo de procesamiento para el procesador central, evitándose efectuar tardadas operaciones de multiplicación y adición con datos de diferentes resoluciones internas

cada vez. Adicionalmente a esto, contar con un banco de procesadores es inherentemente más rápido que contar con un solo procesador central en el sistema y, si a esto último se le añade el hecho de que dicho banco posea internamente algunos bits de protección para los cálculos, entonces se puede superar en buena medida el problema de los puntos de saturación, el cual frecuentemente se presenta en algoritmos de software.

Por último, el presente estudio permitió durante su desarrollo, integrar diversas experiencias y criterios de diseño, con el propósito de obtener un esquema de procesamiento de gran rendimiento y alta velocidad, lo cual se tradujo en el sistema cuyo diseño se llevó al cabo en las secciones previas. Si bien, es posible afirmar que tal vez ésta no sea la solución óptima, si es una solución factible de ser realizada físicamente a un costo relativamente bajo, con una baja densidad de circuitos integrados por tarjeta de circuito impreso y un tiempo de procesamiento corto, lo que hace al sistema propuesto muy atractivo desde el punto de vista de ingeniería.

B I B L I O G R A F I A

[DEFI89] I. Defilippis, U. Sjöström, M. Ansorge, F. Pelladini.
''A 2-Dimensional 16 Point Discrete Cosine Transform Chip for Real Time Video Applications'' *Douzieme Colloque GRETSI. Juan-Les-Pins 12 au Juiin 1989.*

[MING89] Ming-Ting Sun, Ting-Chung Chen Albert M. Gottlieb.
''VLSI Implementation of 16x16 Discrete Cosine Trsnform''.
IEEE Trans. on Circuits and Syst. Vol.36 No.4 april 1989.

[WHIT89] Stanley A. White.
''Applications of Distributed Arithmetic to Digital Signal Processing: A Tutorial Review''.
IEEE Magazine July 1989.

[GORD88] Gordon R. Lang, MOez Dharssi, Fred M. Longstaff.
Philip S. Longstaff, Peter A.S. Metford, Malcolm T. Rimmer.
''An Opptimum Parallel Architecture for High-Speed Real-Time Digital Signal Procesing'' *Motorola Information Systems. February 1989.*

[HSIE87] Hsieh S. Hou.
''A fastt Recursive algorithm for Computing the Discrete Cosine Transorm''.
IEEE Trans. Acuost. Speech and Sign. Proc. Vol.ASSP 35 No.10 October 1987.

[VETT84] Martin Vetterli, Henri J. Nussbaumer.
''Simple FFT and DCT Algorithms with Reduced Number of Operations'' *Signal Processing Vol.6 August 1984.*

[KUNG85] S. Y. Kung.
''VLSI Array Processors''.
IEEE ASSP Magazine July 1985.

[KUNG88] S. Y. Kung.
''VLSI Array Processors''.
Prentice Hall.
Euglewood Cliffs, New Jersey 1988.

[GONZ87] Rafael C. González y Paul Wintz.
''Digital Image Processing''.
Addison-Wesley Publishing Company USA 1987.

[CAVA85] Joseph J. F. Cavanagh.
''Digital Computter Arithmetic Design and Implementation''.
Mc Graw Hill Book Company, Sngapore 1985.

[HWAN86] Kai Hwang, Faye A. Briggs.
''Computer Architecture and Parallel Processing''.
Mc Graw Hill Book Co. 2nd. edition USA 1986.

[INMOS89] ''The Digital Signal Processing Databook''.
1a. edición.
INMOS: SAS-THOMPSON MICROELECTRONICS, USA 1989.

[APRU86] G. Apruzzese, G. Frauly.
''IBM-PC del Laboratorio a la Industria''.
Ediciones TEcnicas Rede, S.A.
Barcelona, Espana 1986.

[QUIN88] Michael J. Quinn.
''Designing Effiicient Algorithms for Parallel Computers''.
Mc Graw Hill Book Company, 2nd edition.
Singapore 1988.

[CAHI85] S. J. Cahill.
''Designing Microprocessor-Based Digital Circuitry''.
Prentice Hall International, USA 1985.

[GARC90] García Garduño Victor, Moctezuma Flores Miguel.
''Implantación de un sistema de compresión de imágenes en un microprocesador de señales''
Tesis para obtener el grado de Maestria en Ingenieria, que otorga la DEPFI. México 1990.

A P E N D I C E

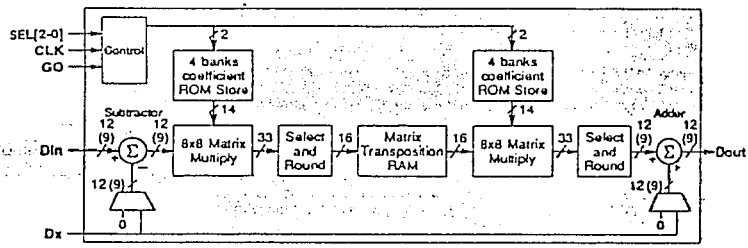
I

DISPOSITIVOS INMOS PARA EL CALCULO DE LA DCT BIDIMENSIONAL

IMS A121
2-D Discrete Cosine Transform Image Processor

inmos®

Advance information



FEATURES

- 8 × 8 Transform size.
- 8 × 8 DCT calculation time = 3.2μs.
- DC to 20 MHz pixel rate.
- 9 bit add/subtract input.
- 12 bit input/output.
- 14 bit fixed coefficients.
- Multifunction capability (DCT, IDCT, Filter).
- Full internal precision, for each dimension.
- Fully synchronous interface.
- High speed CMOS implementation.
- TTL compatible.
- Single +5V ± 10%.
- Power dissipation < 1.5 Watt.
- 44 pin plastic package.

DESCRIPTION

The IMS A121 is a device for computing the Discrete Cosine Transform (DCT & IDCT). It will also function as a 2-D linear filter or perform matrix transposition. These 4 functions operate on blocks of data with a fixed size of 64 samples (8 × 8). The IMS A121 has other functions aimed specifically at the implementation of video codecs; on-chip subtraction and addition functions may be selected to reduce system chip count.

The main computation is performed by two identical multiplication arrays, each of which perform an 8 × 8 matrix multiplication in 64 cycles, with no internal rounding. The DCT/filter coefficients (14 bit) are stored in 4 banks of fixed ROM. The intermediate 8 × 8 matrix result is rounded to 16 bits and stored in the transposition RAM between each multiplication array. The device is fully pipelined with data sampled on the input at the clock frequency and the resultant output appearing 128 clock cycles later.

5.1 OVERALL DEVICE OPERATION

The IMS A121 is a device for computing the Discrete Cosine Transform (DCT) and the Inverse Discrete Cosine Transform (IDCT). It can also perform a simple low-pass filter operation.

The IMS A121 processes blocks of data which are 64 samples long and represent an 8 x 8 matrix. Data is sampled on the Din port every cycle and data is output every cycle on the Dout port.

The GO signal is used to indicate the start of a block. When it is sampled high the data on the Din port is the first sample of the block. The mode select signals SEL[2-0] are sampled at the same time. The remainder of the block of data is sampled on the Din port for the subsequent 63 cycles and during this time the GO signal and the SEL port are ignored. Each consecutive group of eight samples is treated as a column, eight such columns making a block.

The computation is in two stages, between which the block of 64 intermediate samples is stored in the transposition RAM. The transposition RAM serves a dual function of storing the intermediate results and transposing the data from column order into row order. This permits the two matrix computation elements to be identical although the first stage does the column computations and the second stage does the row computations.

Data is output on the Dout port in blocks of 64 samples. However, each consecutive group of eight samples now represents a row because of the internal transposition of data. The first sample of the block is output on the Dout port 128 cycles after the first sample of the block was sampled on the Din port.

An auxillary port, Dx is provided. The data on the Dx port is optionally subtracted from the data on the Din port (DCT mode) or added to the output (IDCT mode).

The IMS A121 views input data in column order and (because of the internal transposition) output data in row order. However, this convention is only used to define the arithmetic which the IMS A121 performs. The system in which the IMS A121 is a component may well view the data going into the IMS A121 in row order and the data coming out in column order.

5.1.1 The fixed ROM coefficients

There are four sets of fixed ROM coefficients, each corresponding to one of the four possible functions the device can perform. The two main functions which the device can perform are the DCT and the IDCT. The other two functions provide assistance for the implementation of a video codec. The filter function is provided at very little overhead because the device is essentially a 2-D filter. The transposition function which is a unity multiplication, enables a simple method of switching out the filter without any external logic.

5.1.2 Number formats

All numbers input to the IMS A121 are *signed integers*. The Din and Dout ports use 12 bit signed integers, while the Dx port uses 9 bit signed integers. In both cases the number format is *twos complement binary*. *Little Endian* format is assumed throughout, so that, for example, Din[0] is the least significant bit of the Din port and Din[11] the most significant (sign) bit. When a nine bit number is transfered over one of the 12 bit ports the most significant nine bits are used. The lowest three bits of the Din port are ignored and the lowest three bits of the Dout port will be zero.

5.1.3 Internal Bit-field Selectors and Rounding

The transforms are implemented by a matrix multiplication with no truncation or rounding. This yields a 33 bit result, with bit-field selectors provided to select the parts of the result which are of interest. 16 bits are selected from the output of the first matrix multiplication, which are stored in the matrix transposition RAM. Either 9 bits or 12 bits are selected from the output of the second matrix multiplication (depending on the selected mode).

Bits below the selected range are discarded although the result is *rounded* not truncated. This is a simple round towards $+\infty$; if the most significant bit of those bits which have been discarded is set then one is added to the bits which are retained.

5.1.4 Overflow, Saturation and Clipping

Overflow can occur in the subtraction unit, the two bit-field selectors or the addition unit. Overflow occurs whenever there are insufficient bits in the result to represent the number. When overflow occurs the result is replaced by the most positive or the most negative number which can be represented (depending on the sign of the correct result).

The device will normally be used in a feedback system. If either positive or negative overflow occurs, then inaccuracies have been introduced. However, the system will remain stable.

In some of the IDCT modes the output is clipped so that all results are positive and all negative numbers are replaced by zero. This ensures that the output is a valid (8-bit) pixel, between 0 and 255.

5.1.5 Subtraction with the DCT function

When the IMS A121 is used to perform the DCT, it is possible to enable the on-chip subtraction unit, so that before the DCT the data on the Dx port is subtracted from the data on the Din port. The data is presented to the Dx port at exactly the same time as to the Din port.

In DCT mode the data on the Din port is a nine bit number (the lowest 3 of 12 bits are ignored). The result of the subtraction is saturated to nine bits before being passed to the matrix multiplier.

5.1.6 Addition with the IDCT function

When the IMS A121 is used to perform the IDCT, it is possible to enable the on-chip addition unit, so that after the IDCT of the data has been done, the result may be added to the data on the Dx port. The timing requires careful consideration because of the latency of the device (128 cycles). The first sample of a block must be presented on the Dx port 124 cycles after the first sample was presented to Din. The data presented to the Dx port should be transposed and is thus in the same order as it will come out of Dout four cycles later.

The result of the addition is saturated to nine bits and then clipped so that all negative numbers are replaced by zero. The nine bit result is presented on Dout[11-3], while Dout[2-0] will be zero. Dout[11] will be zero because all the numbers are positive.

Two modes are provided which perform the IDCT *without* addition. One of these modes disables the adder completely so that nine bit signed results appear on Dout. The other mode does NOT add on the value on the Dx port but still clips the result so that only positive values appear on Dout.

5.1.7 Resetting

The IMS A121 does not have a reset pin. At power-on the internal state will be undefined and as a result the first three blocks processed are not guaranteed correct. GO must be held low for at least 63 cycles to ensure that when it does go high it is interpreted as the start of a block.

5.2 DCT FUNCTION

The DCT function is selected when SEL[2-0]=000 or 100 (mode 0 or 4).

5.2.1 Internal number format

The input for the DCT is a 9 bit signed integer in the range −256 to +255. This is either an external input or the output of the on-chip subtractor depending on SEL[2-0]. The input is multiplied in the matrix multiplication array by 14 bit signed fixed point numbers in the range -2 to $(2-2^{-12})$. The accumulated result of 8 multiply operations is a 26 bit signed integer, the bottom 8 bits of which are rounded (see section 5.1.3) and the top 2 bits used to saturate the output (see section 5.1.4). The result of the first matrix multiply is stored as a 16 bit signed integer and the second matrix multiply performed in exactly the same manner, yielding 33 bit results. The output rounds the bottom 19 bits, saturates the top 2 bits giving a 12 bit signed integer in the range −2048 to +2047.

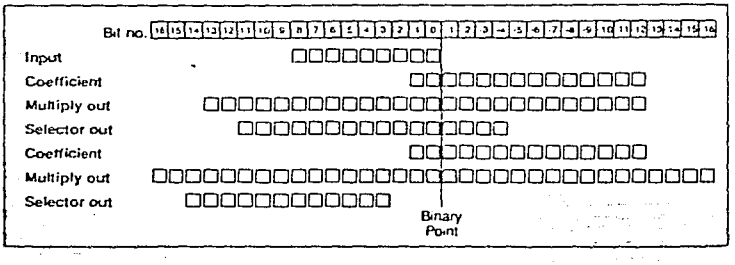
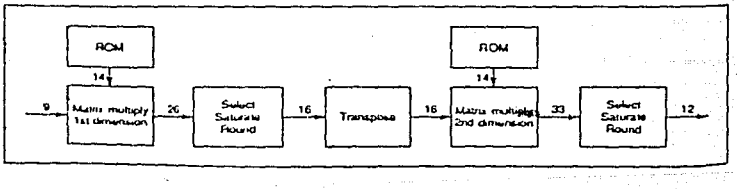


Figure 5.1 DCT internal number format

5.2.2 Internal data flow



5.2.3 The mathematical basis for the DCT

The 1 dimensional equation for the DCT is as follows:

$$\text{Forward transform} \quad X(k) = \sqrt{\frac{2}{N}}\, c(k) \sum_{m=0}^{N-1} x(m) \cos\left[\frac{(2m+1)k\pi}{2N}\right] \qquad k = 0, 1, \cdots, N-1$$

$$\text{where} \quad c(k) = \begin{cases} \frac{1}{\sqrt{2}} & \text{for} \quad k = 0 \\ 1 & \text{for} \quad k = 1 \cdots N-1 \end{cases}$$

where $x(m)$ represent the input samples and $X(k)$ is the resulting output. The special case for the IMS A121 is with $N = 8$ and the actual filter coefficients are then calculated. The following equation is used to calculate the actual filter coefficients.

$$\text{DCT coefficients} \quad Coeff_{km} = \sqrt{2}c(k) \cos\left[\frac{(2m+1)k\pi}{2N}\right]$$

It should be noticed that the coefficients are $2\sqrt{2}$ times bigger than in the forward transform equation. This means that the output after the 2 dimensional DCT is 8 times too big (The 1 dimensional transform is applied twice giving $(2\sqrt{2})^2$ magnitude increase). This is in accordance with the 3 bit shift of the output data necessary to give the correct 12 bit signed output.

5.2.4 DCT coefficients

1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
1.3870	1.1759	0.7857	0.2759	−0.2759	−0.7857	−1.1759	−1.3870
1.3066	0.5412	−0.5412	−1.3066	−1.3066	−0.5412	0.5412	1.3066
1.1759	−0.2759	−1.3870	−0.7857	0.7857	1.3870	0.2759	−1.1759
1.0000	−1.0000	−1.0000	1.0000	1.0000	−1.0000	−1.0000	1.0000
0.7857	−1.3870	0.2759	1.1759	−1.1759	−0.2759	1.3870	−0.7857
0.5412	−1.3066	1.3066	−0.5412	−0.5412	1.3066	−1.3066	0.5412
0.2759	−0.7857	1.1759	−1.3870	1.3870	−1.1759	0.7857	−0.2759

5.2.5 DCT coefficients (14 bit signed integers)

4096	4096	4096	4096	4096	4096	4096	4096
5681	4816	3218	1130	−1130	−3218	−4816	−5681
5352	2217	−2217	−5352	−5352	−2217	2217	5352
4816	−1130	−5681	−3218	3218	5681	1130	−4816
4096	−4096	−4096	4096	4096	−4096	−4096	4096
3218	−5681	1130	4816	−4816	−1130	5681	−3218
2217	−5352	5352	−2217	−2217	5352	−5352	2217
1130	−3218	4816	−5681	5681	−4816	3218	−1130

5.3 IDCT FUNCTION

The IDCT function is selected when SEL[2-0]=001, 101 or 111 (modes 1, 5 or 7).

5.3.1 Internal number format

The input for the IDCT is a 12 bit signed integer in the range −2048 to +2047. The input is multiplied in the matrix multiplication array by 14 bit signed fixed point numbers in the range -2 to $2 - 2^{-12}$. The accumulated result of 8 multiply operations is a 29 bit signed integer, the bottom 8 bits of which are rounded (see section 5.1.3) and the top 5 bits used to saturate the output (see section 5.1.4). The result of the first matrix multiply is stored as a 16 bit signed integer and the second matrix multiply performed in exactly the same manner, yielding 33 bit results. The output rounds the bottom 19 bits, saturates the top 5 bits giving a 9 bit signed integer in the range −256 to +255

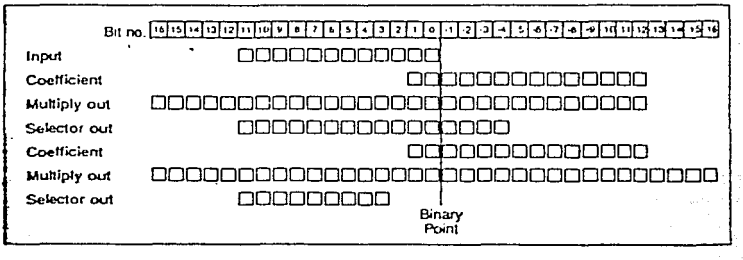
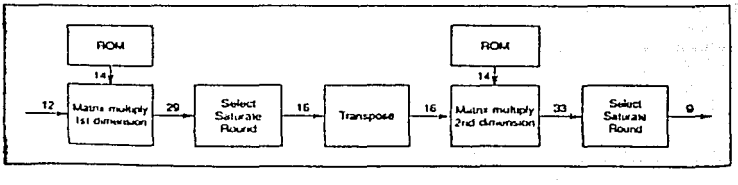


Figure 5.2 IDCT internal number format

5.3.2 Internal data flow



5.3.3 The mathematical basis for the IDCT

The 1 dimensional equation for the IDCT is as follows:

$$\text{Inverse transform} \quad x(m) = \sqrt{\frac{2}{N}} \sum_{k=0}^{N-1} X(k)c(k)\cos\left[\frac{(2m+1)k\pi}{2N}\right] \qquad m = 0, 1, \cdots, N-1$$

$$\text{where} \quad c(k) = \begin{cases} \frac{1}{\sqrt{2}} & \text{for} \quad k = 0 \\ 1 & \text{for} \quad k = 1 \cdots N-1 \end{cases}$$

where $x(m)$ represent the output samples and $X(k)$ is the input. The special case for the IMS A121 is for $N = 8$ and the actual filter coefficients are then calculated. The following equation is used to calculate the actual filter coefficients.

$$\text{IDCT coefficients} \quad Coeff_{m,k} = \sqrt{2}c(k)\cos\left[\frac{(2m+1)k\pi}{2N}\right]$$

It should be noticed that the coefficients are $2\sqrt{2}$ times bigger than in the inverse transform equation. This means that the output after the 2 dimensional IDCT is 8 times too big (The 1 dimensional transform is applied twice giving $(2\sqrt{2})^2$ magnitude increase). This is in accordance with the 3 bit shift of the output data necessary to give the correct result.

5.3.4 IDCT coefficients

1.0000	1.3870	1.3066	1.1759	1.0000	0.7857	0.5412	0.2759
1.0000	1.1759	0.5412	−0.2759	−1.0000	−1.3870	−1.3066	−0.7857
1.0000	0.7857	−0.5412	−1.3870	−1.0000	0.2759	1.3066	1.1759
1.0000	0.2759	−1.3066	−0.7857	1.0000	1.1759	−0.5412	−1.3870
1.0000	−0.2759	−1.3066	0.7857	1.0000	−1.1759	−0.5412	1.3870
1.0000	−0.7857	−0.5412	1.3870	−1.0000	−0.2759	1.3066	−1.1759
1.0000	−1.1759	0.5412	0.2759	−1.0000	1.3870	−1.3066	0.7857
1.0000	−1.3870	1.3066	−1.1759	1.0000	−0.7857	0.5412	−0.2759

5.3.5 IDCT coefficients (14 bit signed integers)

4096	5681	5352	4816	4096	3218	2217	1130
4096	4816	2217	−1130	−4096	−5681	−5352	−3218
4096	3218	−2217	−5681	−4096	1130	5352	4816
4096	1130	−5352	−3218	4096	4816	−2217	−5681
4096	−1130	−5352	3218	4096	−4816	−2217	5681
4096	−3218	−2217	5681	−4096	−1130	5352	−4816
4096	−4816	2217	1130	−4096	5681	−5352	3218
4096	−5681	5352	−4816	4096	−3218	2217	−1130

5.4 FILTER FUNCTION

The filter function is selected with SEL[2-0]=010. (mode 2)

This filter is intended to be used for image data, taking 9 bit signed input data and giving a 9 bit signed result.

5.4.1 Internal number format

The input to the filter is a 9 bit signed integer in the range −256 to +255. The input is multiplied in the matrix multiplication array by 14 bit signed fixed-point numbers in the range -2 to $2-2^{-12}$. The accumulated result of 8 multiply operations is a 26 bit signed integer, the bottom 5 bits of which are rounded (see section 5.1.3) and the top 5 bits are used to saturate the output (see section 5.1.4). The result of the first matrix multiply is stored as a 16 bit signed integer and the second matrix multiply performed in exactly the same manner, yielding 33 bit results. The output rounds the bottom 19 bits, saturates the top 5 bits giving a 9 bit signed integer in the range −256 to +255.

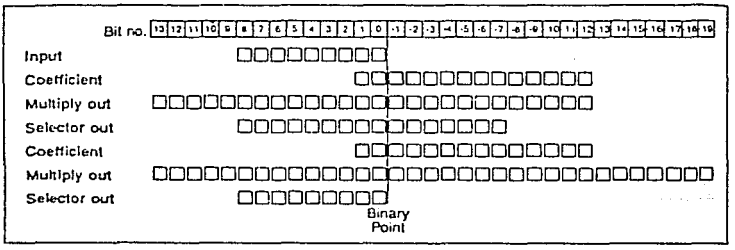
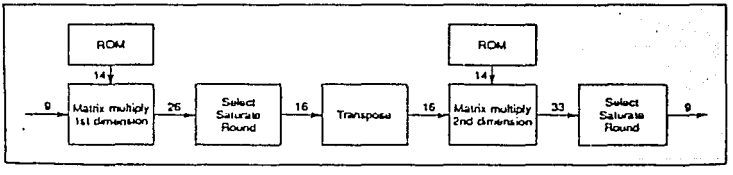


Figure 5.3 Filter and Transpose internal number format

5.4.2 Internal data flow



5.4.3 Definition of filter

The filter is a simple $\frac{1}{4}-\frac{1}{2}-\frac{1}{4}$ filter applied in both dimensions which means that the overall filter kernel is:

$$\frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$$

i.e. an output pixel is calculated from the corresponding pixel in the input field and its eight closest neighbours by evaluating

$$\tfrac{1}{16}(4 \times \text{pixel} + 2 \times (\textstyle\sum \text{four adjacent pixels}) + 1 \times (\textstyle\sum \text{four diagonal pixels}))$$

However, at the block edges, where some of the pixels would fall outside the block boundary, the filter is modified to 0–1–0 which means that along the edge the kernel would be:

$$\tfrac{1}{16}\begin{bmatrix} 0 & 0 & 0 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix} \text{ (rotated to suit)}$$

and the corner pixels are passed through unmodified.

5.4.4 Filter coefficients

$$\begin{bmatrix} 1.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 \\ 0.2500 & 0.5000 & 0.2500 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 \\ 0.0000 & 0.2500 & 0.5000 & 0.2500 & 0.0000 & 0.0000 & 0.0000 & 0.0000 \\ 0.0000 & 0.0000 & 0.2500 & 0.5000 & 0.2500 & 0.0000 & 0.0000 & 0.0000 \\ 0.0000 & 0.0000 & 0.0000 & 0.2500 & 0.5000 & 0.2500 & 0.0000 & 0.0000 \\ 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.2500 & 0.5000 & 0.2500 & 0.0000 \\ 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.2500 & 0.5000 & 0.2500 \\ 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 1.0000 \end{bmatrix}$$

5.4.5 Filter coefficients (14 bit signed integers)

$$\begin{bmatrix} 4096 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1024 & 2048 & 1024 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1024 & 2048 & 1024 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1024 & 2048 & 1024 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1024 & 2048 & 1024 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1024 & 2048 & 1024 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1024 & 2048 & 1024 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 4096 \end{bmatrix}$$

5.5 TRANSPOSER FUNCTION

The transposition function is selected with SEL[2-0]=011. (mode 3)

This is intended to be used for filtering image data, taking 9 bit signed input data and giving a 9 bit signed result. Data is passed through unmodified and is intended to be used in conjunction with the filter function (SEL[2-0]=010), so that by toggling SEL[0] the filter can be switched in and out.

5.5.1 Internal number format and data flow

The internal number format and data flow for the transpose function are the same as for the filter function. Refer to sections 5.4.1 and 5.4.2.

5.5.2 Transposition coefficients

$$\begin{bmatrix} 1.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 \\ 0.0000 & 1.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 \\ 0.0000 & 0.0000 & 1.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 \\ 0.0000 & 0.0000 & 0.0000 & 1.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 \\ 0.0000 & 0.0000 & 0.0000 & 0.0000 & 1.0000 & 0.0000 & 0.0000 & 0.0000 \\ 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 1.0000 & 0.0000 & 0.0000 \\ 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 1.0000 & 0.0000 \\ 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 1.0000 \end{bmatrix}$$

5.5.3 Transposition coefficients (14 bit signed integers)

$$\begin{bmatrix} 4096 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 4096 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 4096 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 4096 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 4096 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 4096 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 4096 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 4096 \end{bmatrix}$$

5.6 PIN DESIGNATIONS

System services

Pin	In/out	Function
VCC, GND		Power supply and return
CLK	In	Input clock

Synchronous input/output

Pin	In/out	Function
GO	In/out	Initiate input/computation/output cycle
Din[11-0]	In	Data input port
Dout[11-0]	Out	Data output port
Dx[11-3]	In	Addition/subtraction port
SEL[2-0]	In	Mode select input port

5.6.1 System services

Power

Power is supplied to the device via the VCC and GND pins. Several of each are provided to minimise inductance within the package. All supply pins must be connected. The supply must be decoupled close to the chip by at least one 100nF low inductance (e.g. ceramic) capacitor between VCC and GND. Four layer boards are recommended; if two layer boards are used, extra care should be taken in decoupling.

Input voltages must not exceed specification with respect to VCC and GND.

CLK

The clock input signal CLK controls the timing of input and the output on the three dedicated interfaces, and controls the progress of data through the addition/subtraction units, multipliers and transposition RAM. Since the IMS A121 is fully static, the clock can be stopped in either phase without corrupting data.

5.6.2 Synchronous input/output

GO

The GO signal is active high and is sampled on the rising edge of the input clock. If the device is processing a previous block of data, the GO signal is ignored. Otherwise, the processing of a block of 64 pixels commences and the GO signal is ignored for a further 63 cycles. Data is always assumed to be valid for the 64 cycles from the start of a major cycle. Blocks of data may be processed at any time and with any spacing between the major blocks, by toggling the GO signal as necessary.

Din[11-0]

The data input port is sampled 64 times on successive clock cycles, commencing when GO is sampled high. Data must be valid on the rising edge of CLK for each of the 64 cycles. The block of data may be considered as an 8 × 8 matrix, where each group of 8 samples represents a column, and the 8 columns are sampled consecutively until the block is complete. The data is *twos complement, Little Endian* so that Din[11] gives sign information, and Din[0] is the least significant bit.

Dout[11-0]

The data output port will be valid for periods spanning 64 clock cycles. The data will be valid on the rising edge of the clock, exactly 128 cycles (the latency) after the data was sampled on the input. This output data, which may be considered as an 8 x 8 matrix, is transposed with respect to the input data. The data is *twos complement, Little Endian* like the input data.

Blocks of data may follow directly after one another so that the first data of a block is presented exactly 64 cycles after the first data of the preceding block. However, if there is a gap between blocks zero will appear on the data output port between blocks of data.

Dx[11-3]

The addition/subtraction port is sampled on each clock cycle in exactly the same way as the data input port. The data on this port will either be subtracted from the signal on the data input port before matrix multiplication, or, added to the result of matrix multiplication prior to output. The addition and subtraction functions can never be used together. The function selected is determined by the SEL[2-0] signals. The data is *twos complement, Little Endian* like the Din/Dout data. Note however, that although the Dx port has a different width, Dx[10] has the same bitwise significance as Din[10]/Dout[10].

The timing of data on the Dx port is different depending on the selected mode.

In the case of subtraction in the DCT mode, SEL[1-0]=00, data is presented on the Dx port on the same cycle as the corresponding data (from which it will be subtracted) is presented on the Din port.

In the case of addition in the IDCT mode, SEL[1-0]=01, data is presented on the Dx port exactly 4 cycles before the corresponding data (to which it will have been added) appears on the Dout port.

SEL[2-0]

The mode select input port is sampled on the rising edge of CLK, when GO is active, at the start of a block of data. This fixes the selected mode for the entire block of data.

SEL[2-0]	Mode	Function	PreSubtract	PostAdd	Clipping	Din width	Dout width
000	0	DCT	Disabled	Disabled	Disabled	9	12
001	1	IDCT	Disabled	Disabled	Disabled	12	9
010	2	Filter	Disabled	Disabled	Disabled	9	9
011	3	Transpose	Disabled	Disabled	Disabled	9	9
100	4	DCT	Enabled	Disabled	Disabled	9	12
101	5	IDCT	Disabled	Enabled	Enabled	12	9
110	6	Reserved – Do not use					
111	7	IDCT	Disabled	Disabled	Enabled	12	9

5.7 ELECTRICAL SPECIFICATION

5.7.1 DC electrical characteristics

Absolute maximum ratings

Symbol	Parameter	Min	Max	Units	Notes (1)
VCC	DC supply voltage	0	7.0	V	2
VI, VO	Voltage on input and output pins	−1.0	VCC+0.5	V	2
TA	Temperature under bias	−40	85	°C	2
TS	Storage temperature	−65	150	°C	2
PDmax	Power dissipation		1.5	W	2

1 All voltages are with respect to GND.

2 This is a stress rating only and functional operation of the device at these or any other conditions above those indicated in the operational sections of this specification is not implied. Stresses greater than those listed may cause permanent damage to the device. Exposure to absolute maximum rating conditions for extended periods may affect reliability.

DC operating conditions

Symbol	Parameter	Min.	Nom.	Max.	Units	Notes (1)
VCC	DC supply Voltage	4.5	5.0	5.5	V	
VIH	Input Logic '1' Voltage	2.0		VCC+0.5	V	2
VIL	Input Logic '0' Voltage	−0.5		0.8	V	2
TA	Ambient Operating Temperature	0		70	°C	3

Notes

1 All voltages are with respect to GND. All GND pins must be connected to GND.

2 Input signal transients 10 ns wide, are permitted in the voltage ranges GND − 0.5 V to GND − 1.0 V and VCC + 0.5 V to VCC + 1.0 V.

3 400 linear ft/min transverse air flow.

DC characteristics

Symbol	Parameter	Min.	Max.	Units	Notes (1,2)
VOH	Output Logic '1' Voltage	2.4	VCC	V	IO ≤ −4.4 mA
VOL	Output Logic '0' Voltage	0	0.4	V	IO ≤ 4.4 mA
IIN	Input leakage current (any input)		±10	μA	3
ICC	Average power supply current		300	mA	4

Notes

1 All voltages are with respect to GND. All GND pins must be connected to GND.

2 Under the conditions specified by the DC operating conditions.

3 VCC = VCC(max), GND ≤ VIN ≤ VCC

4 This applies at 20 MHz and will be less at slower clock rates

5.7.2 A.C. timing characteristics

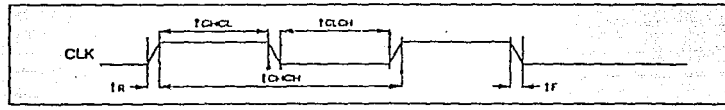
All timings are given for a load of 30pF unless otherwise stated.

Clock requirements

Symbol	Parameter	Min	Max	Units	Notes
t_{CHCL}	Clock Pulse High Width	20		ns	
t_{CLCH}	Clock Pulse Low Width	20		ns	
t_{CHCH}	Clock Period	50		ns	
t_R	Clock rise time	0	50	ns	1
t_F	Clock fall time	0	50	ns	1

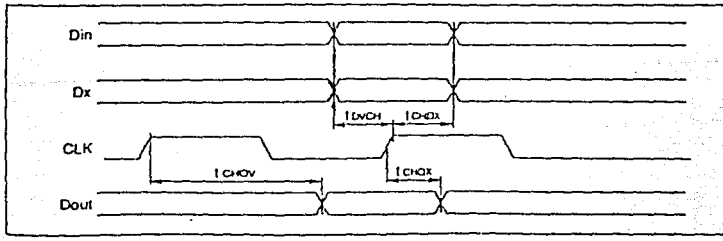
Notes

1 The clock edges should be monotonic between VIL and VIH.



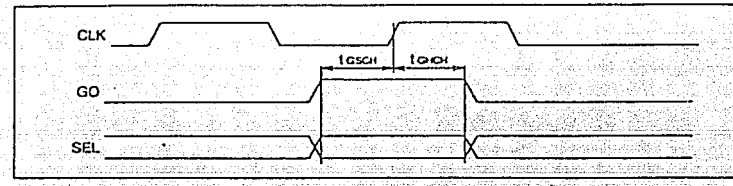
Synchronous input and output (Din, Dout, Dx)

Symbol	Parameter	Min	Max	Units	Notes
t_{CHOV}	CLK high to Dout Valid		38	ns	
t_{CHOX}	Dout hold time after CLK	2		ns	
t_{DVCH}	Din/Dx setup time to CLK high	10		ns	
t_{CHDX}	Din/Dx hold time to CLK high	0		ns	



Synchronous control (GO, SEL[2-0])

Symbol	Parameter	Min	Max	Units	Notes
tGSCH	GO/SEL hold to clock high	0		ns	
tGSCH	GO/SEL setup to clock high	10		ns	



Overall data timing

5.8 PACKAGE SPECIFICATIONS

5.8.1 44 pin PLCC package

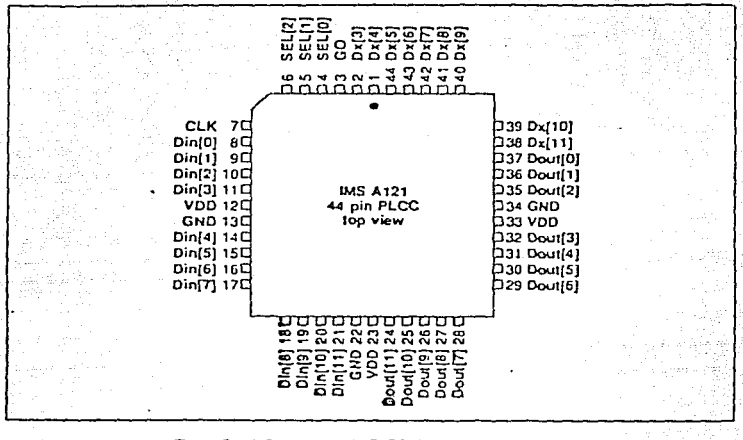


Figure 5.4 IMS A121 44 pin PLCC J-bend package pinout

Note

All VCC pins must be connected to the 5 Volt power supply.
All GND pins must be connected to ground.

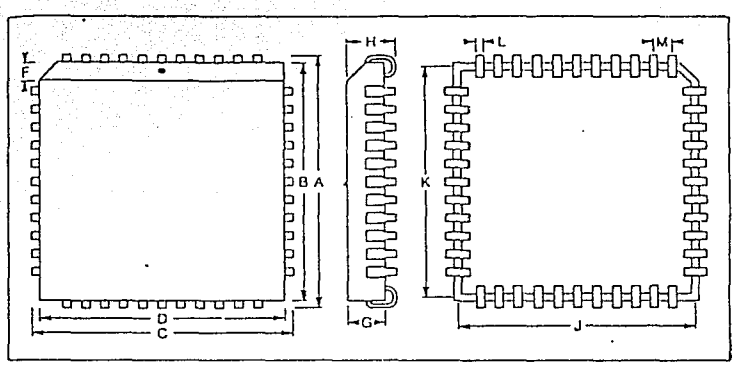


Figure 5.5 44 pin PLCC J-bend package dimensions

DIM	Millimetres		Inches		Notes
	NOM	TOL	NOM	TOL	
A	17.577	±	0.692	±	
B	16.612	±	0.654	±	
C	17.577	±	0.692	±	
D	16.612	±	0.654	±	
F	1.143		0.045		
G	3.861		0.152		
H	4.369	±	0.172	±	
J	15.748	±	0.620	±	
K	15.748	±	0.620	±	
L	0.457		0.018		
M	1.270		0.050		

Table 5.1 44 pin PLCC J-bend package dimensions

PLCC thermal characteristics

Symbol	Parameter	Min	Nom	Max	Units	Notes
θ JA	Junction to ambient thermal resistance				°C /W	1,2

Notes

1 Measured at 400 linear ft/min transverse air flow.

2 This parameter is sampled and not 100% tested.

5.9 ORDERING DETAILS

The following table indicates the designation of the IMS A121 variants.

INMOS designation	Package	Clock speed	Military/commercial
IMS A121-J20S	Plastic LCC	20 MHz	commercial

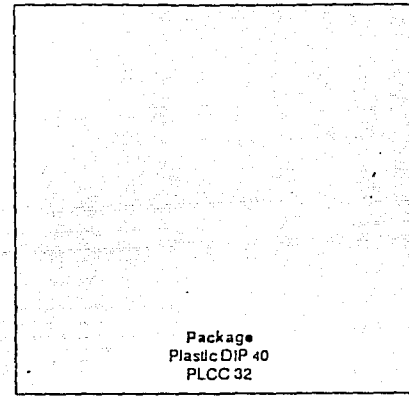
SGS-THOMSON MICROELECTRONICS

STV3208

8 x 8 DISCRETE COSINE TRANSFORM (DCT)

TENTATIVE DATA

- 0 TO 40 MHz PIXEL RATE IN SINGLE PRECISION MODE,
- 0 TO 20 MHz PIXEL RATE IN DOUBLE PRECISION MODE
- FORWARD AND INVERSE 8 x 8 TRANSFORM
- 9-BIT TWO'S COMPLEMENT PIXEL FORMAT
- 12-BIT TWO'S COMPLEMENT COEFFICIENT FORMAT
- OPTIMIZED ACCURACY FOR 8-BIT TWO'S COMPLEMENT PIXEL FORMAT
- SUPPORTS CCITT H261 PROPOSED RECOMMENDATION
- ZIG ZAG SCANNING FOR COEFFICIENT BLOCKS
- FULLY TTL AND CMOS COMPATIBLE
- CMOS TECHNOLOGY
- SINGLE +5 VOLT POWER SUPPLY
- MAXIMUM POWER DISSIPATION : 750mW AT 40MHz



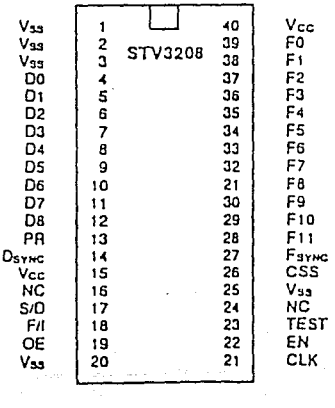
DESCRIPTION

The STV3208 is a dedicated circuit for the 8 x 8 discrete cosine transform (DCT) computation. Two-dimensional forward DCT (FDCT) or inverse DCT (IDCT) is performed for 8 x 8 block sizes and a pixel rate up to 40MHz. The circuit architecture is fully bidirectional with 9-bit magnitude pixel data bus and a 12-bit magnitude coefficient data bus programmed as input or output depending on the selection of FDCT or IDCT.

	FDCT	IDCT	Data Format
Pixel Bus	Input	Output	9-bit 2's Complement
Coefficient Bus	Output	Input	12-bit 2's Complement

For the forward transform, the input pixels are coded on 9-bit 2's complement and the output coefficients are coded on 12-bit 2's complement. For the inverse transform, the data format is identical with the coefficients used as input and the pixels used as output.
A double precision mode allows to meet the proposed CCITT requirements for IDCT accuracy for a pixel rate up to 20MHz. An optimized mode allows to process 8-bit 2's complement pixels with improved accuracy.

PIN CONNECTIONS



DESCRIPTION

The STV3208 is a dedicated circuit for the 8*8 discrete cosine transform (DCT) computation. Two-dimensional forward DCT (FDCT) or inverse DCT (IDCT) is performed for 8*8 block sizes and a pixel rate up to 40 MHz. The circuit architecture is fully bidirectionnal with a 9-bit magnitude pixel data bus and a 12-bit magnitude coefficient data bus programmed as input or output depending on the selection of FDCT or IDCT.

	FDCT	IDCT	Data Format
Pixel Bus	Input	Output	9-bit 2's Complement
Coefficient Bus	Output	Input	12-bit 2's Complement

For the forward transform, the input pixels are coded on 9-bit 2's complement and the output coefficients are coded on 12-bit 2's complement. For the inverse transform, the data format is identical with the coefficients used as input and the pixels used as output.

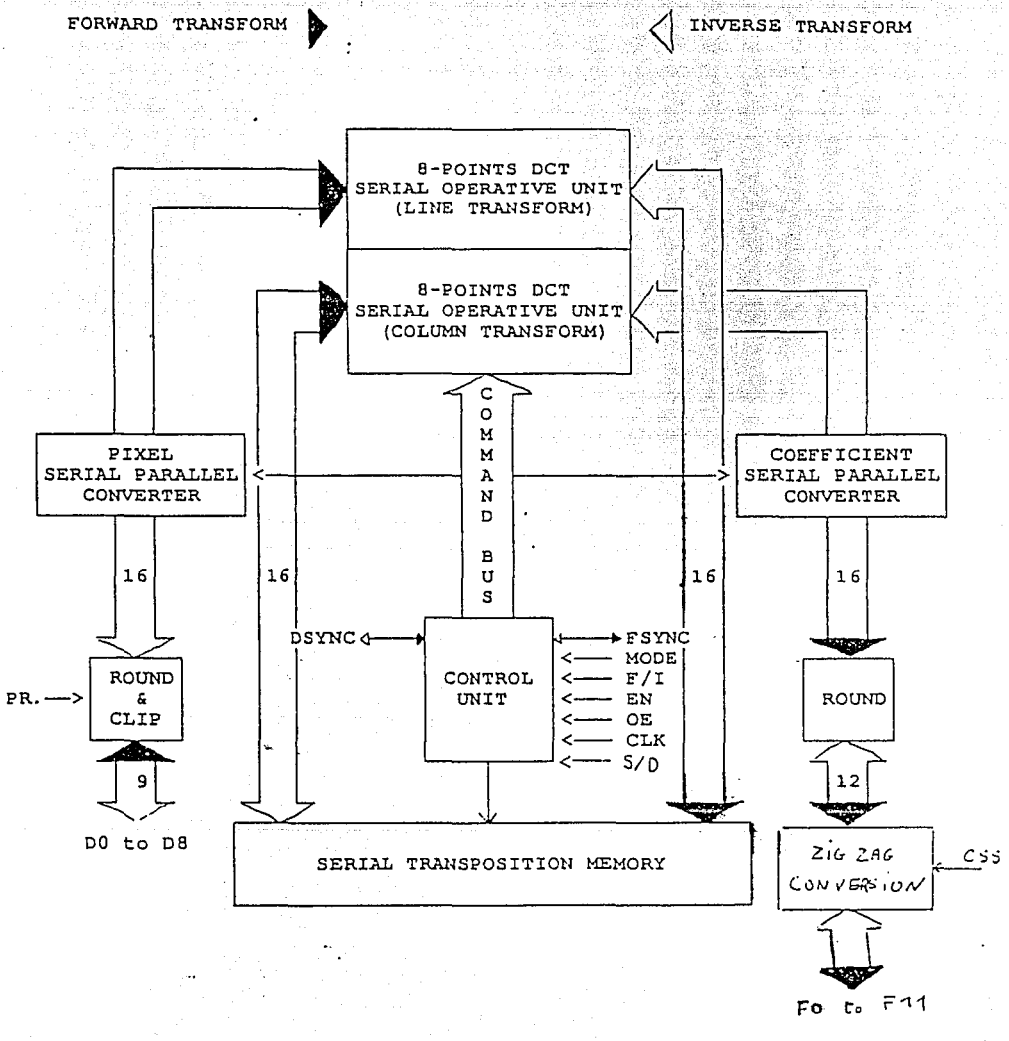
A double precision mode allows to meet the CCITT requirements for IDCT accuracy for a pixel rate up to 20 Mhz.

An optimized mode allows to process 8-bit 2's complement pixels with improved accuracy.

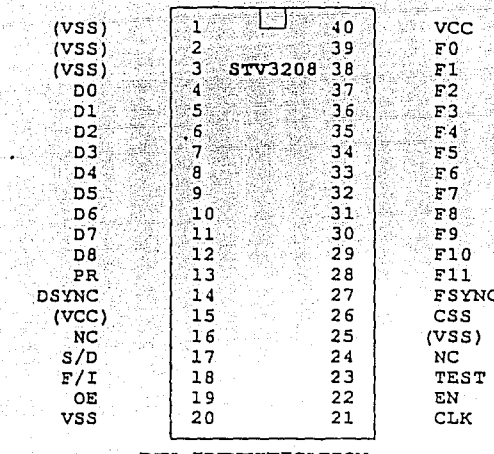
FEATURES

* 0 to 40 MHz Pixel rate in single precision mode,
  0 to 20 MHz Pixel rate in double precision mode
* Forward and Inverse 8*8 Transform
* 9-bit Two's Complement Pixel Format
* 12-bit Two's Complement Coefficient Format
* Optimized accuracy for 8-bit Two's Complement Pixel Format
* CCITT compatible
* ZIG ZAG scanning for coefficient blocks
* Fully TTL and CMOS Compatible
* CMOS Technology
* Single +5 Volt Power Supply
* Maximum Power Dissipation : 750mW at 40MHz

FUNCTIONAL BLOCK DIAGRAM



PIN CONFIGURATION



PIN IDENTIFICATION

N°	SYMBOL	TYPE	FUNCTION DESCRIPTION
4-12	D0 to D8	I/O	PIXEL DATA BUS
13	PR	Input	PIXEL RANGE SELECTION
14	DSYNC	I/O	PIXEL BLOCK SYNCHRONIZATION SIGNAL
17	S/D	Input	SINGLE/DOUBLE PRECISION SELECTION
18	F/I	Input	FDCT/IDCT SELECTION
19	OE	Input	OUTPUT THREE-STATE CONTROL
21	CLK	Input	CLOCK SIGNAL
22	EN	Input	CLOCK ENABLE SIGNAL
23	TEST	Input	TEST MODE SELECTION
26	CSS	Input	ZIG ZAG SELECTION
27	FSYNC	I/O	COEFFICIENTS BLOCK SYNCHRONIZATION SIGNAL
28-39	F0 to F11	I/O	COEFFICIENT DATA BUS
1,2,3,20,25	VSS	Power	GROUND
15,40	VCC	Power	POWER SUPPLY

FUNCTIONAL DESCRIPTION

EQUATIONS

The STV3208 performs 8*8 Two dimensional Discrete Cosine Transform according to the following formula :

Equations for 9-bit PIXEL DATA (PR pin set to low):

Forward Transform Equation:

$$F(u,v) = \text{Round}\left[\frac{1}{4} C(u)C(v) \sum_{i=0}^{7}\sum_{j=0}^{7} D(i,j)\cos\frac{(2*i+1)u\pi}{16}\cos\frac{(2*j+1)v\pi}{16}\right]$$

Inverse Transform equation:

$$D(i,j) = \text{Round}\left[\frac{1}{4} \sum_{u=0}^{7}\sum_{v=0}^{7} C(u)C(v)F(u,v)\cos\frac{(2*i+1)u\pi}{16}\cos\frac{(2*j+1)v\pi}{16}\right]$$

where $C(u) = 2^{-1/2}$ if $u=0$
$= 1$ otherwise

Equations for 8-bit PIXEL DATA (PR pin set to high):

Forward Transform Equation:

$$F(u,v) = \text{Round}\left[\frac{1}{2} C(u)C(v) \sum_{i=0}^{7}\sum_{j=0}^{7} D(i,j)\cos\frac{(2*i+1)u\pi}{16}\cos\frac{(2*j+1)v\pi}{16}\right]$$

Inverse Transform equation:

$$D(i,j) = \text{Round}\left[\frac{1}{8} \sum_{u=0}^{7}\sum_{v=0}^{7} C(u)C(v)F(u,v)\cos\frac{(2*i+1)u\pi}{16}\cos\frac{(2*j+1)v\pi}{16}\right]$$

where $C(u) = 2^{-1/2}$ if $u=0$
$= 1$ otherwise

DATA FLOW ORDERING

The pixel block is scanned column by column (ORDER 1) or line by line (ORDER 2). If CSS is high, the coefficient block is scanned with a zig zag order. Figure 2 shows the relation between pixels order and coefficient order.

	PIXEL ORDER										COEFFICIENT ORDER							
	1	9	17	25	33	41	49	57			1	2	6	7	15	16	28	29
	2	10	18	26	34	42	50	58			3	5	8	14	17	27	30	43
	3	11	19	27	35	43	51	59			4	9	13	18	26	31	42	44
	4	12	20	28	36	44	52	60			10	12	19	25	32	41	45	54
ORDER 1	5	13	21	29	37	45	53	61	<——>		11	20	24	33	40	46	53	55
	6	14	22	30	38	46	54	62			21	23	34	39	47	52	56	61
	7	15	23	31	39	47	55	63			22	35	38	48	51	57	60	62
	8	16	24	32	40	48	56	64			36	37	49	50	58	59	63	64
	1	2	3	4	5	6	7	8			1	3	4	10	11	21	22	36
	9	10	11	12	13	14	15	16			2	5	9	12	20	23	35	37
	17	18	19	20	21	22	23	24			6	8	13	19	24	34	38	49
	25	26	27	28	29	30	31	32			7	14	18	25	33	39	48	50
ORDER 2	33	34	35	36	37	38	39	40	<——>		15	17	26	32	40	47	51	58
	41	42	43	44	45	46	47	48			16	27	31	41	46	52	57	59
	49	50	51	52	53	54	55	56			28	30	42	45	53	56	60	63
	57	58	59	60	61	62	63	64			29	43	44	54	55	61	62	64

Figure 2 : Data Ordering (CSS high)

If CSS is low, the coefficient block is scanned line by line and the pixel block is scanned column by column, or the coefficient block is scanned column by column and the pixel block is scanned line by line.

	PIXEL ORDER										COEFFICIENT ORDER							
	1	9	17	25	33	41	49	57			1	2	3	4	5	6	7	8
	2	10	18	26	34	42	50	58			9	10	11	12	13	14	15	16
	3	11	19	27	35	43	51	59			17	18	19	20	21	22	23	24
	4	12	20	28	36	44	52	60			25	26	27	28	29	30	31	32
ORDER 1	5	13	21	29	37	45	53	61	<——>		33	34	35	36	37	38	39	40
	6	14	22	30	38	46	54	62			41	42	43	44	45	46	47	48
	7	15	23	31	39	47	55	63			49	50	51	52	53	54	55	56
	8	16	24	32	40	48	56	64			57	58	59	60	61	62	63	64
	1	2	3	4	5	6	7	8			1	9	17	25	33	41	49	57
	9	10	11	12	13	14	15	16			2	10	18	26	34	42	50	58
	17	18	19	20	21	22	23	24			3	11	19	27	35	43	51	59
	25	26	27	28	29	30	31	32			4	12	20	28	36	44	52	60
ORDER 2	33	34	35	36	37	38	39	40	<——>		5	13	21	29	37	45	53	61
	41	42	43	44	45	46	47	48			6	14	22	30	38	46	54	62
	49	50	51	52	53	54	55	56			7	15	23	31	39	47	55	63
	57	58	59	60	61	62	63	64			8	16	24	32	40	48	56	64

Figure 2' : Data Ordering (CSS low)

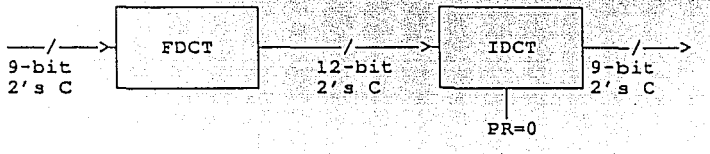
DATA FORMAT

Coefficients format is 12-bit 2's Complement, corresponding to the range -2048 to 2047.

There are 2 possible ranges for pixel data :

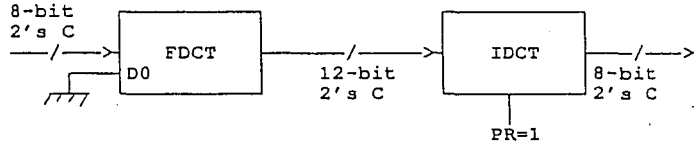
9-bit two's complement magnitude

The pixel data range is -256 to +255. In this case the PR pin must be set to 1 for IDCT. D8 is the most significant bit and D0 the least significant bit for the pixel data. A clipping to the range -256 to +255 is performed before outputting reconstructed pixels after an IDCT.



8-bit two's complement magnitude :

Pixel data range is -128 to +127. In this case D0 must be set to 0 and the PR pin must be set to 1 for IDCT. D8 is the most significant bit and D1 the least significant bit for the pixel data. A clipping to the range -128 to +127 is performed before outputting reconstructed pixels after an IDCT.



This mode may be used for intra picture coding. In this case, pixel data range is 0 to 255. For a FDCT, the most significant bit of input pixel data (D8) must be inverted before entering the chip. This is equivalent to substract 128 to the input pixel data. Note that this operation will only have effect on the DC value F(0,0). For an IDCT, the most significant bit of output pixel data (D7) must be inverted. This is equivalent to add 128 to the output pixel data.

BLOCK FLOW

Depending on the application, blocks may be entered in different way.

Latent period : the latent period between input data and corresponding output results is 170 clock cycles (if FDCT is selected) or 168 clock cycles (if IDCT is selected) in single precision mode (S/D pin set to 1). This means that the first data of a resulting block is provided 170 clock cycles (if FDCT is selected) or 168 clock cycles (if IDCT is selected) after the first data of the corresponding input block. The latent period is 109 cycles (if FDCT is selected) or 108 cycles (if IDCT is selected) in double precision mode (S/D pin set to 0).

LATENCY

	FORWARD DCT	INVERSE DCT
S/D = 1 SINGLE PRECISION	~~170~~ 171 CYCLES	~~168~~ 169 CYCLES
S/D = 0 DOUBLE PRECISION	~~109~~ 142 CYCLES	~~108~~ 141 CYCLES

Synchronization signals : an input block synchronization signal must be provided. The input pin for this signal is DSYNC if FDCT is selected and FSYNC if IDCT is selected. This signal must be active with the first data of each input block or group of blocks.
An output block synchronization signal is provided. The output pin for this signal is FSYNC if FDCT is selected and DSYNC if IDCT is selected. This signal is active with the first data of each output block or group of blocks.
The output synchronization signal is equal to the input synchronization signal shifted from the latent period.

Continuous block flow

Input data are entered continuously with one new item data at each clock cycle and output data is provided continuously with one new result data item at each clock cycle.
The input synchronization signal can be provided for each input block. In this case the output synchronization pulse is provided for each output block (figure 3). An other way is to provide a synchronization pulse only for the first block of a group of blocks. In this case, only one synchronization pulse is provided for the first output block (figure 4).

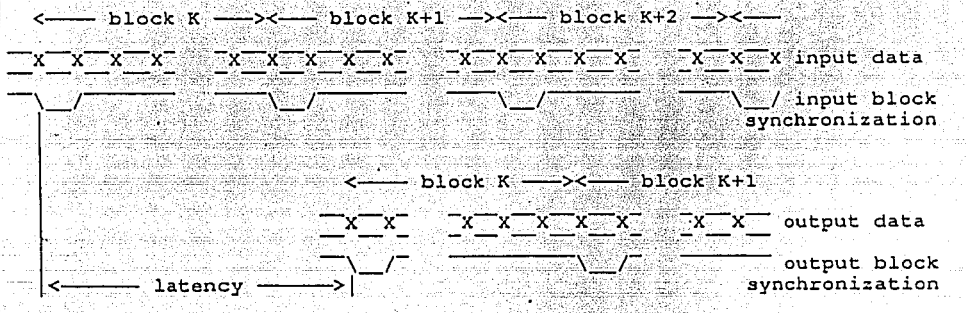


figure 3 : continuous block flow 1

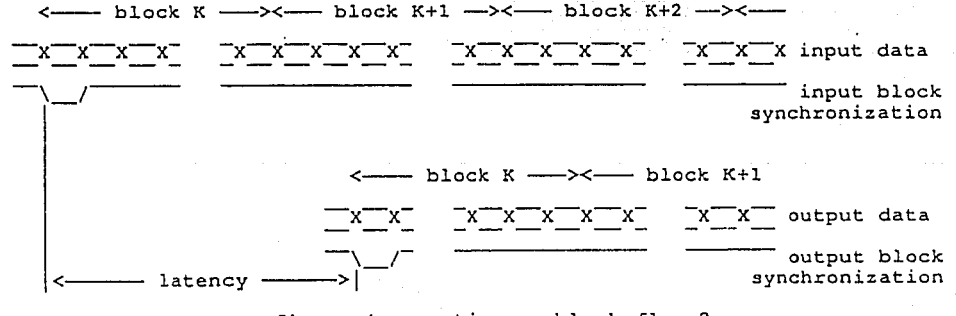


figure 4 : continuous block flow 2

Continuous block flow with bypass of irrelevant data

It is possible to process a block flow including irrelevant data (corresponding to line suppression for example) as if it was a continuous block flow. One way is to stop the clock signal during the irrelevant data occurrence. Another way is to use the Clock Enable Signal (EN) which allows to stop the chip internal clock during irrelevant data occurrence. (figure 5)

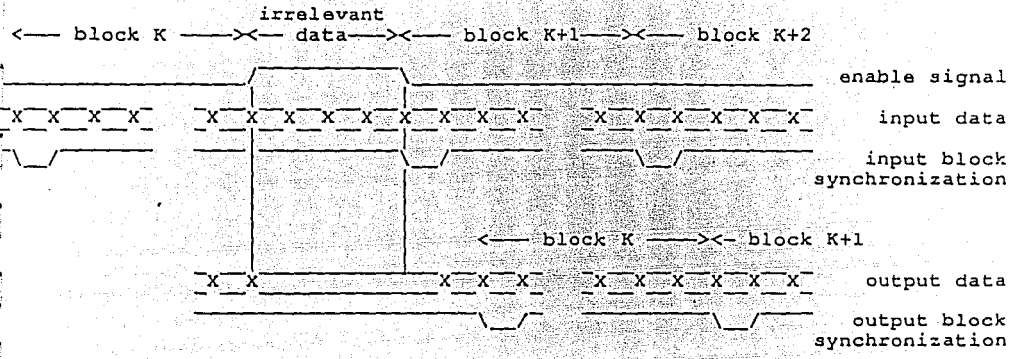


figure 5 : Continuous Block Flow with Irrelevant Data

Burst block flow

Single blocks (or groups of block) may not be contiguous. In other words, delay cycles between two blocks (or groups of block) may exist. During these delay cycles, the clock is still running and the chip continues to perform computations. The constraint is that the internal pipe line must not be broken when a new block occurs. To take this constraint into account, the number of delay cycles (NC) must respect one of the following conditions :

1 - the number of delay cycles (NC) is greater than or equal to the latency. In this case the pipe line is empty (all the relevant data has been outputted) when a new input block processing starts.

2 - the number of delay cycles (NC) is a multiple of 64. In this case, the input data always remains synchronous with the internal pipe line.

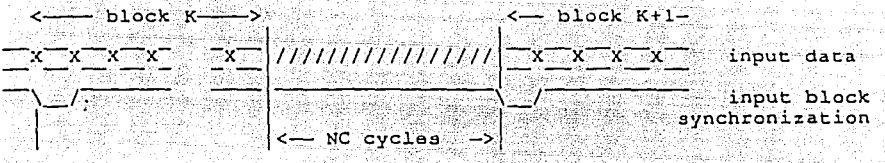


figure 6 : burst block flow

Mixed FDCT/IDCT

In some low frequency application, it could be cost effective to use only one chip to compute all the FDCT and IDCT required by the coding scheme. Blocks must be entered in a burst fashion with at least the latency time between the last pixel of input pixels for FDCT and the first pixel of input coefficients for IDCT. The same delay must be respected between the last pixel of input coefficients for IDCT and the first pixel of input pixels for FDCT.

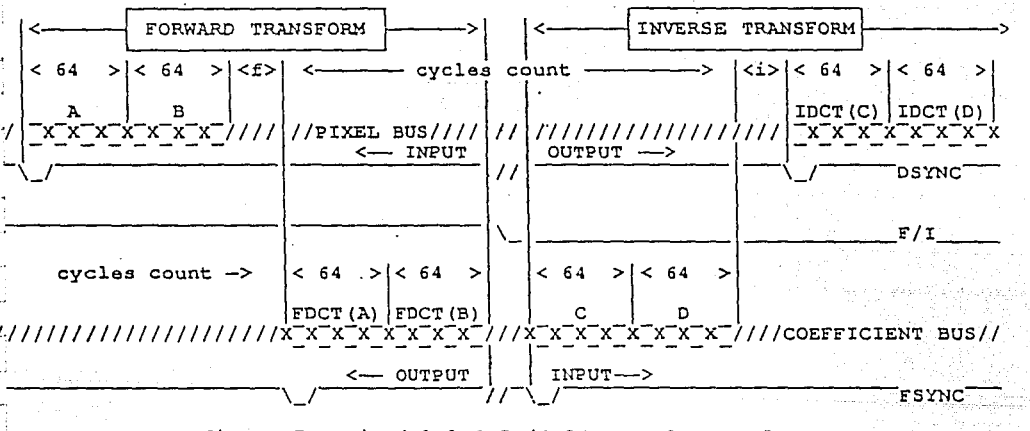


figure 7 : mixed 8x8 FDCT/IDCT example waveforms

PRECISION SELECTION

For single precision mode, the S/D pin must be low. In this case, the maximum rating for pixel is 40 Mhz.
For double precision mode, the S/D pin must be high. In this case, the maximum rating for pixel is 20 Mhz.

PINS DESCRIPTION

CLK : Clock signal

DATA PINS

D0 to D8 : 9-bit bidirectional Pixel data bus pins. Direction is programmed by the F/I pin :

F/I state	D0 to D8 direction
high	Input
low	Output

Data is loaded (when input) or settled (when output) on rising edge of CLK. D0 is the least significant bit and D8 the most significant one. Note that for the optimized mode for 8-bit 2's C pixel data, D1 is the least significant bit and D0 must be set to 0.

	MSB								LSB	
D8	D7	D6	D5	D4	D3	D2	D1	D0	Pin
-256	128	64	32	16	8	4	2	1	Weight

DSYNC : Pixel data block synchronization signal. This pin is bidirectional with the direction programmed by the F/I pin (like D0 to D8). DSYNC is active (low level) with the first pixel data of a block (or a group of blocks).

F0 to F11 : 12-bit bidirectional Coefficient data bus pins. Direction is programmed by the F/I pin :

F/I state	F0 to F11 direction
high	Output
low	Input

Data is loaded (when input) or settled (when output) on rising edge of CLK. F0 is the least significant bit and F11 the most significant one.

MSB											LSB	
F11	F10	F9	F8	F7	F6	F5	F4	F3	F2	F1	F0	Pin
-2048	1024	512	256	128	64	32	16	8	4	2	1	Weight

FSYNC : Coefficient data block synchronization signal. This pin is bidirectional with the direction programmed by the F/I pin (like F0 to F11). FSYNC is active (low level) with the first coefficient data of a block (or a group of blocks).

CONTROL PINS

F/I : Forward or inverse selection. When F/I is high, forward DCT is performed, when F/I is low, inverse DCT is performed.

S/D : Single or double precision. When S/D is high, single precision is selected. When S/D is low, double precision is selected with the result that the CCITT requirements for IDCT is met.

CSS : Coefficient Scanning Selection. When CSS is high, zig zag scanning of coefficient block is selected. When CSS is low, row scanning of coefficient block is selected.

PR : Pixel range selection. If PR is low, pixel range is -256 to +255. If PR is high, pixel range is -128 to +127.

OE : Output enable. This signal is active low. When OE is high, all outputs (defined by the F/I pin state) are forced to the high impedance state.

EN : Enable. This signal is active low. When EN is high, internal states of the chip are frozen. When EN becomes low, execution restarts. EN must change when CLK is high.

STATE	FUNCTION
F/I is High	Forward DCT
F/I is Low	Inverse DCT
S/D is High	Single Precision
S/D is Low	Double Precision (CCITT Standard)
CSS is High	Zig Zag Scanning of coefficients
CSS is Low	Row Scanning of coefficients
PR is High	8-bit 2's C Pixel Data
PR is Low	9-bit 2's C Pixel Data
OE is High	Outputs Active
OE is Low	High Impedance Outputs
EN is High	Internal Clock is stopped
EN is Low	Internal Clock runs

POWER SUPPLY AND GROUND PINS

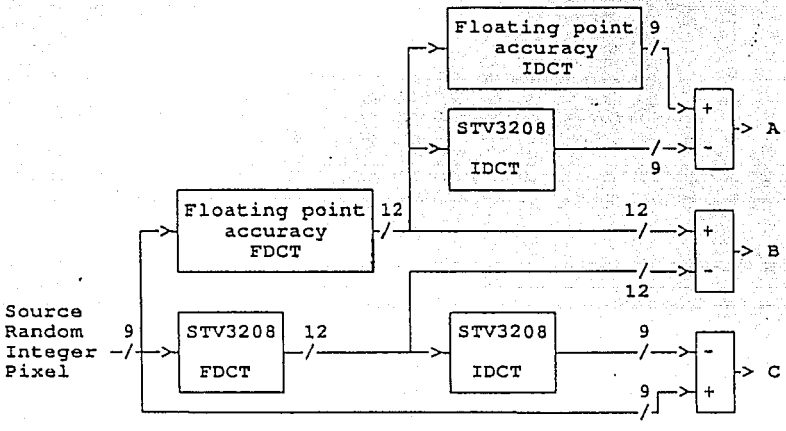
VCC : +5 Volt power supply.

VSS : ground potential.

OTHERS : test control. This pin is reserved and must be low in normal mode.

ACCURACY CHARACTERISTICS

The accuracy characteristics have been measured according to the following scheme :



A : characteristics of IDCT. Error between the IDCT computed with 64-bit floating point accuracy and the IDCT computed by the STV3208 is measured. Measures have been done according to the CCITT WGXV method. The main result is that for double precision mode, the standard is met :

	STV3208 single precision	STV3208 double precision	CCITT requiremen
Peak Error	1	1	1
Peak Mean Square Error	0.0403	0.0258	0.06
Overall Mean Square Error	0.0287	0.0200	0.02
Peak Mean Error	0.0125	0.0041	0.015
Overall Mean Error	0.0050	0.0000062	0.0015

	9-bit pixels		8-bit pixels	
	single precision	double precision	single precision	double precision
Exact value	97.1%	98.0%	99.96%	99.97%
Errors of ±1 LSB	2.9%	2.0%	0.04%	0.03%
Errors of ±2 LSB	0%	0%	0%	0%

B : characteristics of FDCT. Error between the FDCT computed with 64-bit floating point accuracy and the FDCT computed by the STV3208 is measured.

	9-bit pixels		8-bit pixels	
	single precision	double precision	single precision	double precision
Exact value	93.6%	96.9%	93.6%	96.9%
Errors of ±1 LSB	6.4%	4.1%	6.4%	4.1%
Errors of ±2 LSB	0%	0%	0%	0%

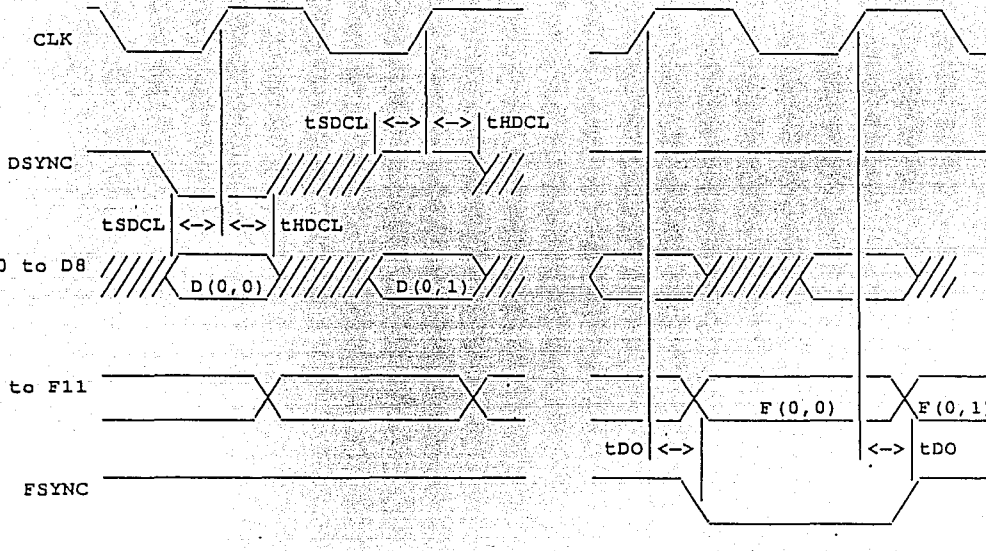
C : characteristics of FDCT followed by an IDCT. Error between the source picture and the FDCT computed by the STV3208 followed by an IDCT computed by the STV3208 is measured.

	9-bit pixels		8-bit pixels	
	single precision	double precision	single precision	double precision
Exact value	89.3%	90.6%	99.88%	99.92%
Errors of ±1 LSB	10.7%	9.4%	0.12%	0.08%
Errors of ±2 LSB	0%	0%	0%	0%

See annex for detailed results

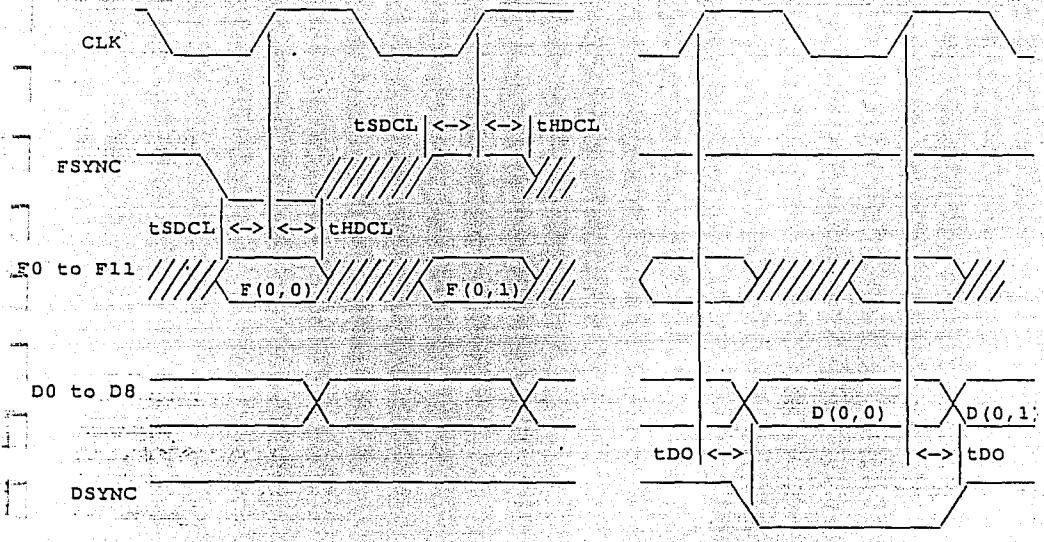
TIMING WAVEFORMS

nch Signals Timing Diagram for a Forward Transform



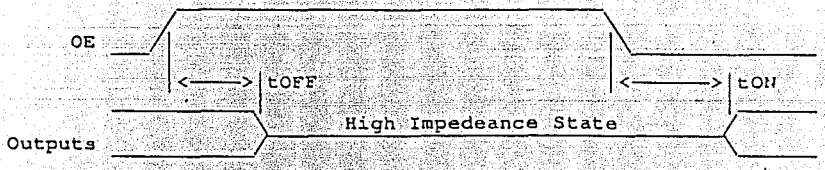
NOTE : FSYNC will be in unknown state during the first cycles after the power up.

ıch Signals Timing Diagram for an Inverse Transform

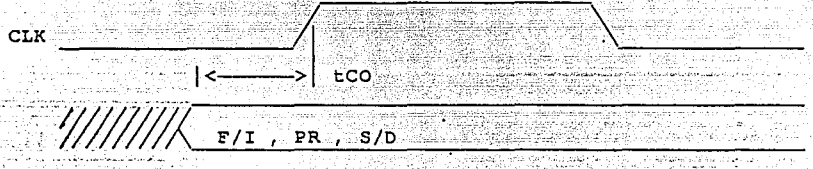


NOTE : DSYNC will be in unknown state during the first cycles after the power up.

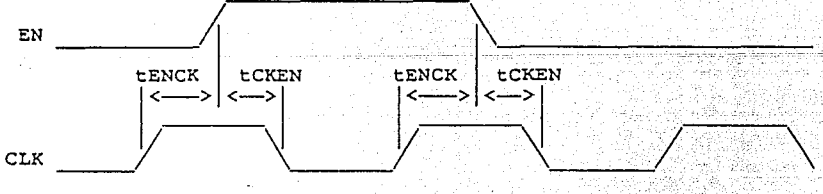
Output Enable Signal Timing Waveforms



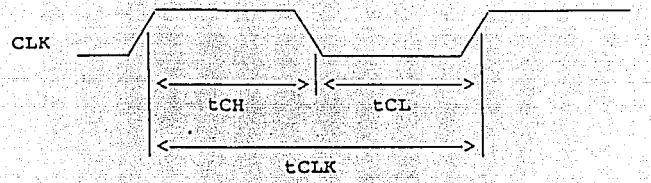
Control Static Signal Timing Waveforms



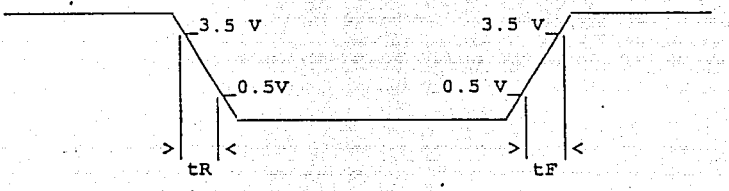
Enable Signal Timing Waveforms



Clock Timing Waveforms



Output Timing Waveforms



ELECTRICAL CHARACTERISTICS

Absolute maximum ratings

Supply voltage (VCC) : 6 Volt
Operating temperature range : 0 to 70 °C
Voltage on any pin relative to VSS : -0.5 to VCC+0.5 Volt

DC electrical characteristics

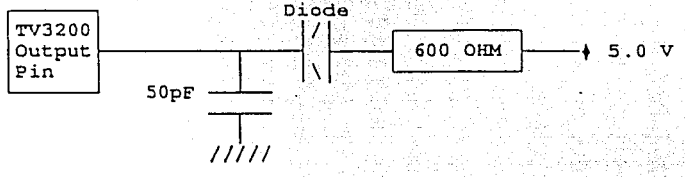
Operating Conditions : VSS = 0 Volt
TA = 0 to 70 °C
VCC = 5V±10% unless otherwise specified

PARAMETER	CONDITIONS	MIN	MAX	UNIT
Operating Voltage		4.5	5.5	V
Power Supply Ripple			0.5	V
Supply current : Fclk = 40 MHz Fclk = 0 MHz	Cload = 50pF on all outputs. All inputs at VCC or VSS		150 1	mA mA
Input Voltage Level : Logic Low Logic High	VCC = 5±0.5 V	 2	0.8	 V V
High Impedance input leakage : I/O Buffers Input Buffers	VIN = VSS to VCC	-5 -1	+5 +1	µA µA
Output Voltage Level : Logic Low Iload=6.4mA Logic High Iload=-1mA	VCC = 4.5 V	 2.7	0.4	 V V
Output Current Level : Source Vout = 0V Sink Vout = 5V	VCC = 5 V	 -25 45		 mA mA
Input capacitance (DIL PACKAGE) D0 to D8, F0 to F11 DSYNC, FSYNC all others	Voffset = 2.5 V F = 1MHz		 12 12 10	 pF pF pF

AC electrical characteristics

Operating Conditions : VSS = 0 Volt
TA = 0 to 70 °C
VCC = 5V±10% unless otherwise specified
Outputs Loads : capacitance = 50pF
Current Logic Low = 6.4mA

Test Load on outputs :



Timings are measured between threshold voltage of 1.5 Volt unless otherwise specified.

PARAMETER	SYMBOL	MIN (nS)	MAX (nS)
Rising time from 0.5 to 3.5V	tR		10
Falling time from 3.5 to 0.5V	tF		10
Clock High Pulse Width	tCH	12	
Clock Low Pulse Width	tCL	12	
Clock Cycle duration	tCLK	25	
Data Setup Time from CLK ↓	tsdcl	5	
Data Hold Time from CLK ↓	thdcl	5	
Output Data Delay from CLK ↑	tdo		15
Enable Hold Time from CLK ↑	tCKEN	0	
Enable Setup Time from CLK ↓	tENCK	5	
Delay from OE ↑ to Output going to High Impedance State	tOFF		20
Delay from OE ↓ to Output going to High or Low State	tON		20
F/I or PR or S/D Setup Time from Beginning of Input Stream	tCO	100	

ANNEX : accuracy results

8*8 DCT CHIP ACCURACY - DOUBLE PRECISION MODE - 10,000 Blocks

IDCT CHARACTERISTICS
Pixel range : -256 to 255

Peak Mean Square Error : 0.0258000
Overall Mean Square Error : 0.0200031
Peak Mean Error : 0.0041000
Overall Mean Error : 0.0000062

Errors of -1 : 6399 (1.00%)
Errors of 0 : 627198 (98.00%)
Errors of 1 : 6403 (1.00%)

Mean Error Versus Pixel Location

-0.0013	0.0006	0.0002	-0.0022	0.0040	0.0006	0.0004	0.0000
0.0002	-0.0003	-0.0027	0.0009	-0.0026	-0.0014	0.0007	0.0004
0.0006	-0.0006	-0.0021	0.0010	0.0000	-0.0007	-0.0007	0.0012
0.0009	-0.0003	-0.0020	0.0041	-0.0001	-0.0022	-0.0030	0.0020
0.0004	-0.0020	0.0016	0.0031	0.0025	-0.0025	0.0002	-0.0001
0.0019	-0.0002	0.0002	0.0011	-0.0007	0.0016	0.0016	0.0000
-0.0024	0.0008	-0.0015	-0.0012	-0.0001	-0.0020	-0.0008	-0.0025
0.0006	-0.0012	0.0032	0.0015	0.0014	0.0003	-0.0001	0.0001

Mean Square Error Versus Pixel Location

0.0117	0.0156	0.0186	0.0188	0.0168	0.0180	0.0174	0.0136
0.0156	0.0167	0.0221	0.0233	0.0226	0.0210	0.0191	0.0150
0.0168	0.0226	0.0237	0.0254	0.0240	0.0229	0.0211	0.0200
0.0185	0.0225	0.0258	0.0255	0.0245	0.0244	0.0252	0.0204
0.0178	0.0212	0.0240	0.0249	0.0227	0.0219	0.0218	0.0195
0.0185	0.0220	0.0220	0.0225	0.0255	0.0232	0.0206	0.0172
0.0168	0.0198	0.0179	0.0220	0.0211	0.0208	0.0166	0.0153
0.0120	0.0164	0.0174	0.0217	0.0196	0.0189	0.0155	0.0139

8 DCT CHIP ACCURACY - DOUBLE PRECISION MODE - 10,000 Blocks

FDCT CHARACTERISTICS
Pixel range : -256 to 255

Peak Mean Square Error : 0.0607000
Overall Mean Square Error : 0.0314906
Peak Mean Error : 0.0572000
Overall Mean Error : -0.0044750

Errors of -1 : 11509 (1.80%)
Errors of 0 : 619846 (96.85%)
Errors of 1 : 8645 (1.35%)

Mean Error Versus Pixel Location

0.0000	0.0007	-0.0009	0.0010	0.0000	0.0011	0.0009	-0.0009
-0.0567	0.0054	-0.0016	0.0012	0.0076	0.0017	-0.0028	-0.0024
-0.0515	-0.0020	-0.0001	-0.0032	0.0086	-0.0015	-0.0003	0.0026
-0.0540	0.0007	0.0008	-0.0024	0.0082	-0.0033	-0.0016	0.0038
0.0000	0.0004	0.0012	-0.0008	0.0000	-0.0010	-0.0002	0.0000
-0.0533	0.0006	-0.0006	-0.0010	0.0072	-0.0016	0.0009	-0.0006
-0.0568	-0.0001	0.0026	-0.0005	0.0062	0.0009	-0.0001	0.0007
-0.0572	0.0013	-0.0010	0.0016	0.0076	-0.0035	0.0035	-0.0019

Mean Square Error Versus Pixel Location

0.0000	0.0297	0.0085	0.0152	0.0000	0.0197	0.0197	0.0287
0.0607	0.0504	0.0344	0.0412	0.0302	0.0437	0.0458	0.0554
0.0515	0.0350	0.0173	0.0198	0.0144	0.0213	0.0231	0.0308
0.0544	0.0403	0.0174	0.0274	0.0196	0.0301	0.0322	0.0402
0.0000	0.0294	0.0088	0.0160	0.0000	0.0154	0.0154	0.0270
0.0535	0.0448	0.0246	0.0284	0.0226	0.0314	0.0371	0.0448
0.0570	0.0431	0.0222	0.0313	0.0208	0.0343	0.0365	0.0455
0.0594	0.0527	0.0282	0.0404	0.0320	0.0477	0.0469	0.0601

8 DCT CHIP ACCURACY - DOUBLE PRECISION MODE - 10,000 Blocks

FDCT FOLLOWED BY AN IDCT CHARACTERISTICS
Pixel range : -256 to 255

Peak Mean Square Error : 0.1056000
Overall Mean Square Error : 0.0940828
Peak Mean Error : 0.0350000
Overall Mean Error : 0.0054078

Errors of -1 : 28375 (4.43%)
Errors of 0 : 579790 (90.59%)
Errors of 1 : 31834 (4.97%)
Errors of 2 : 1 (0.00%)

Mean Error Versus Pixel Location

0.0037	0.0035	-0.0075	0.0325	0.0292	0.0052	-0.0022	0.0263
-0.0317	0.0111	-0.0090	0.0172	0.0010	0.0013	-0.0014	0.0109
-0.0240	0.0135	-0.0100	0.0166	0.0044	-0.0011	-0.0004	0.0084
0.0024	0.0095	-0.0042	0.0350	0.0290	-0.0013	-0.0056	0.0240
0.0016	0.0023	-0.0077	0.0308	0.0250	-0.0012	-0.0044	0.0267
-0.0310	0.0090	-0.0033	0.0187	0.0102	-0.0017	-0.0012	0.0101
-0.0320	0.0079	-0.0068	0.0166	0.0016	0.0011	-0.0010	0.0067
0.0029	0.0008	-0.0102	0.0333	0.0258	0.0017	-0.0029	0.0304

Mean Square Error Versus Pixel Location

0.0905	0.0841	0.0887	0.0999	0.0960	0.0914	0.0920	0.0917
0.0925	0.0949	0.0962	0.0954	0.0892	0.0915	0.0940	0.0897
0.0964	0.0961	0.0986	0.0944	0.0974	0.0897	0.0916	0.0968
0.0922	0.0897	0.0996	0.0980	0.0998	0.0939	0.0956	0.0960
0.0964	0.0975	0.0961	0.1056	0.1040	0.0890	0.0958	0.0935
0.0964	0.0972	0.0923	0.0935	0.0980	0.0939	0.0930	0.0943
0.0964	0.0917	0.0900	0.0962	0.0900	0.0941	0.0892	0.0931
0.0889	0.0910	0.0904	0.0975	0.0960	0.0933	0.0891	0.0944

*8 DCT CHIP ACCURACY - DOUBLE PRECISION MODE - 10,000 Blocks

IDCT CHARACTERISTICS
Pixel range : -128 to 127

Peak Mean Square Error : 0.0007000
Overall Mean Square Error : 0.0002500
Peak Mean Error : 0.0006000
Overall Mean Error : 0.0000406

Errors of -1 : 67 (0.01%)
Errors of 0 : 639840 (99.97%)
Errors of 1 : 93 (0.01%)

Mean Error Versus Pixel Location

0.0003	-0.0002	0.0000	0.0001	0.0001	-0.0002	0.0001	0.0000
-0.0001	-0.0002	0.0002	-0.0003	0.0001	0.0003	-0.0004	0.0002
0.0001	0.0001	-0.0006	0.0001	0.0001	0.0001	0.0002	-0.0002
0.0003	-0.0001	0.0001	-0.0003	0.0004	-0.0002	-0.0002	0.0002
0.0005	0.0000	0.0000	0.0001	0.0002	0.0002	0.0000	0.0005
-0.0001	-0.0002	0.0002	-0.0001	0.0001	0.0000	0.0003	0.0002
-0.0002	0.0002	-0.0002	0.0002	-0.0001	-0.0002	0.0000	0.0000
0.0003	0.0002	-0.0001	0.0001	0.0000	-0.0001	0.0002	0.0003

Mean Square Error Versus Pixel Location

0.0003	0.0002	0.0000	0.0003	0.0003	0.0002	0.0001	0.0000
0.0003	0.0002	0.0002	0.0005	0.0003	0.0005	0.0004	0.0002
0.0001	0.0003	0.0006	0.0001	0.0001	0.0001	0.0002	0.0002
0.0003	0.0003	0.0001	0.0003	0.0006	0.0002	0.0002	0.0004
0.0005	0.0002	0.0000	0.0007	0.0002	0.0002	0.0002	0.0005
0.0003	0.0004	0.0004	0.0001	0.0001	0.0000	0.0005	0.0002
0.0002	0.0004	0.0004	0.0002	0.0003	0.0002	0.0000	0.0000
0.0005	0.0002	0.0001	0.0001	0.0002	0.0001	0.0002	0.0003

8 DCT CHIP ACCURACY - DOUBLE PRECISION MODE - 10,000 Blocks

FDCT CHARACTERISTICS
Pixel range : -128 to 127

Peak Mean Square Error : 0.0590000
Overall Mean Square Error : 0.0316172
Peak Mean Error : 0.0567000
Overall Mean Error : -0.0041859

Errors of -1 : 11457 (1.79%)
Errors of 0 : 619765 (96.84%)
Errors of 1 : 8778 (1.37%)

Mean Error Versus Pixel Location

0.0000	0.0023	-0.0014	0.0012	0.0000	0.0015	0.0003	-0.0012
-0.0528	0.0018	0.0017	-0.0028	0.0104	0.0008	-0.0015	0.0007
-0.0519	0.0030	0.0000	-0.0022	0.0067	0.0009	-0.0027	-0.0003
-0.0531	-0.0031	-0.0001	0.0011	0.0085	0.0006	0.0030	0.0028
0.0000	-0.0003	0.0016	-0.0002	0.0000	0.0003	0.0019	0.0013
-0.0567	-0.0040	0.0020	-0.0015	0.0102	0.0008	-0.0012	0.0004
-0.0554	-0.0009	0.0003	-0.0019	0.0078	-0.0013	-0.0017	-0.0031
-0.0532	-0.0017	-0.0002	0.0017	0.0108	0.0025	0.0024	-0.0028

Mean Square Error Versus Pixel Location

0.0000	0.0317	0.0094	0.0132	0.0000	0.0201	0.0183	0.0280
0.0550	0.0526	0.0327	0.0390	0.0318	0.0478	0.0461	0.0565
0.0519	0.0342	0.0170	0.0200	0.0155	0.0187	0.0225	0.0291
0.0531	0.0405	0.0193	0.0281	0.0187	0.0330	0.0336	0.0392
0.0000	0.0309	0.0094	0.0146	0.0000	0.0191	0.0191	0.0265
0.0573	0.0424	0.0218	0.0297	0.0212	0.0370	0.0342	0.0442
0.0558	0.0417	0.0249	0.0325	0.0182	0.0351	0.0395	0.0477
0.0574	0.0533	0.0320	0.0353	0.0326	0.0479	0.0466	0.0590

8 DCT CHIP ACCURACY - DOUBLE PRECISION MODE - 10,000 Blocks

FDCT FOLLOWED BY AN IDCT CHARACTERISTICS
Pixel range : -128 to 127

Peak Mean Square Error	: 0.0016000
Overall Mean Square Error	: 0.0007969
Peak Mean Error	: 0.0012000
Overall Mean Error	: 0.0002000

Errors of -1	:	191 (0.03%)
Errors of 0	:	639490 (99.92%)
Errors of 1	:	319 (0.05%)

Mean Error Versus Pixel Location

0.0003	0.0002	0.0000	0.0010	0.0006	-0.0001	0.0004	0.0004
-0.0002	-0.0003	0.0000	0.0004	0.0004	0.0001	0.0005	-0.0002
-0.0002	0.0002	-0.0003	0.0003	0.0005	-0.0003	-0.0001	0.0007
0.0001	-0.0002	0.0003	0.0010	0.0010	-0.0001	0.0001	0.0010
0.0009	0.0004	-0.0002	0.0008	0.0012	-0.0002	0.0003	0.0010
-0.0004	-0.0001	-0.0001	0.0001	-0.0001	0.0000	0.0002	0.0000
-0.0003	-0.0002	-0.0002	0.0006	0.0000	0.0001	-0.0002	0.0001
0.0002	-0.0003	0.0000	0.0008	0.0005	-0.0003	0.0000	0.0007

Mean Square Error Versus Pixel Location

0.0009	0.0010	0.0008	0.0014	0.0010	0.0009	0.0008	0.0010
0.0006	0.0007	0.0006	0.0010	0.0008	0.0013	0.0007	0.0002
0.0004	0.0008	0.0007	0.0007	0.0007	0.0007	0.0009	0.0009
0.0005	0.0006	0.0005	0.0012	0.0014	0.0007	0.0007	0.0012
0.0011	0.0006	0.0008	0.0010	0.0014	0.0004	0.0007	0.0012
0.0008	0.0005	0.0003	0.0003	0.0009	0.0006	0.0008	0.0006
0.0007	0.0008	0.0006	0.0008	0.0006	0.0005	0.0008	0.0007
0.0010	0.0007	0.0006	0.0016	0.0009	0.0007	0.0008	0.0009

8 DCT CHIP ACCURACY - SINGLE PRECISION MODE - 10,000 Blocks

IDCT CHARACTERISTICS
Pixel range : -256 to 255

Peak Mean Square Error : 0.0403000
Overall Mean Square Error : 0.0286562
Peak Mean Error : 0.0125000
Overall Mean Error : 0.0050188

Errors of -1 : 7564 (1.18%)
Errors of 0 : 621660 (97.13%)
Errors of 1 : 10776 (1.68%)

Mean Error Versus Pixel Location

0.0077	0.0024	0.0046	0.0045	0.0065	0.0031	0.0058	0.0061
0.0034	0.0044	0.0019	0.0067	0.0042	0.0047	0.0064	0.0044
0.0102	0.0048	0.0049	0.0078	0.0048	0.0020	0.0033	0.0062
0.0125	0.0041	0.0049	0.0078	0.0059	0.0045	0.0073	0.0060
0.0004	0.0054	0.0052	0.0117	0.0061	0.0040	0.0056	0.0036
0.0038	0.0066	0.0058	0.0048	0.0027	0.0039	0.0048	0.0026
0.0015	0.0035	0.0035	0.0039	0.0055	0.0038	0.0043	0.0043
0.0055	0.0042	0.0055	0.0051	0.0069	0.0052	0.0023	0.0054

Mean Square Error Versus Pixel Location

0.0195	0.0198	0.0218	0.0229	0.0215	0.0209	0.0204	0.0195
0.0234	0.0234	0.0291	0.0297	0.0288	0.0281	0.0268	0.0236
0.0280	0.0306	0.0315	0.0368	0.0326	0.0306	0.0299	0.0266
0.0347	0.0373	0.0403	0.0380	0.0373	0.0367	0.0397	0.0352
0.0334	0.0356	0.0366	0.0377	0.0377	0.0376	0.0338	0.0360
0.0268	0.0304	0.0318	0.0296	0.0359	0.0295	0.0300	0.0268
0.0231	0.0265	0.0259	0.0307	0.0279	0.0268	0.0251	0.0249
0.0169	0.0210	0.0221	0.0249	0.0239	0.0240	0.0175	0.0186

*3 DCT CHIP ACCURACY - SINGLE PRECISION MODE - 10,000 Blocks

FDCT CHARACTERISTICS
Pixel range : -256 to 255

Peak Mean Square Error : 0.1037000
Overall Mean Square Error : 0.0644250
Peak Mean Error : 0.0472000
Overall Mean Error : 0.0109906

Errors of -1 : 17099 (2.67%)
Errors of 0 : 598768 (93.56%)
Errors of 1 : 24133 (3.77%)

Mean Error Versus Pixel Location

0.0000	-0.0355	0.0106	-0.0221	0.0000	0.0151	0.0405	0.0394
-0.0339	0.0011	0.0120	-0.0309	0.0324	0.0170	0.0392	0.0287
0.0105	-0.0103	0.0151	-0.0328	0.0285	0.0178	0.0420	0.0331
-0.0200	-0.0121	0.0094	-0.0343	0.0332	0.0094	0.0378	0.0334
0.0000	-0.0339	0.0083	-0.0227	0.0000	0.0125	0.0373	0.0405
0.0112	-0.0021	0.0118	-0.0325	0.0349	0.0155	0.0432	0.0310
0.0367	-0.0087	0.0078	-0.0343	0.0263	0.0168	0.0472	0.0326
0.0462	-0.0031	0.0077	-0.0295	0.0317	0.0170	0.0454	0.0343

Mean Square Error Versus Pixel Location

0.0000	0.0965	0.0556	0.0755	0.0000	0.0577	0.0455	0.0578
0.0581	0.1037	0.0782	0.0951	0.0566	0.0850	0.0722	0.0827
0.0339	0.0929	0.0613	0.0840	0.0411	0.0682	0.0592	0.0633
0.0434	0.0977	0.0664	0.0879	0.0474	0.0714	0.0606	0.0682
0.0000	0.0941	0.0495	0.0739	0.0000	0.0557	0.0417	0.0557
0.0360	0.1037	0.0650	0.0853	0.0475	0.0729	0.0652	0.0698
0.0441	0.1015	0.0630	0.0863	0.0389	0.0686	0.0672	0.0708
0.0584	0.1011	0.0689	0.0861	0.0479	0.0804	0.0724	0.0845

8 DCT CHIP ACCURACY - SINGLE PRECISION MODE - 10,000 Blocks

FDCT FOLLOWED BY AN IDCT CHARACTERISTICS
Pixel range : -256 to 255

Peak Mean Square Error : 0.1493000
Overall Mean Square Error : 0.1074141
Peak Mean Error : 0.0983000
Overall Mean Error : 0.0110547

Errors of -1 : 30833 (4.82%)
Errors of 0 : 571261 (89.26%)
Errors of 1 : 37904 (5.92%)
Errors of 2 : 2 (0.00%)

Mean Error Versus Pixel Location

0.0519	-0.0194	0.0164	0.0134	0.0200	0.0066	0.0067	0.0258
-0.0852	0.0164	-0.0053	0.0079	-0.0072	0.0053	0.0045	0.0067
0.0746	-0.0231	0.0310	-0.0053	0.0132	0.0010	0.0194	0.0147
0.0222	-0.0103	0.0202	0.0154	0.0256	0.0039	0.0072	0.0206
0.0178	-0.0111	0.0126	0.0260	0.0301	0.0056	0.0057	0.0309
-0.0293	0.0044	0.0066	0.0126	0.0144	0.0044	0.0061	0.0206
-0.0149	-0.0030	0.0078	0.0136	0.0117	0.0040	0.0078	0.0240
0.0983	-0.0286	0.0233	0.0175	0.0331	0.0118	0.0109	0.0380

Mean Square Error Versus Pixel Location

0.1219	0.1134	0.1168	0.1190	0.1182	0.1090	0.1189	0.1194
0.1226	0.0960	0.0987	0.1025	0.1018	0.0989	0.0997	0.0985
0.1220	0.1033	0.1052	0.0989	0.1046	0.1054	0.1010	0.1001
0.1108	0.1001	0.1088	0.1048	0.1060	0.1029	0.1050	0.1098
0.1106	0.1065	0.1020	0.1102	0.1135	0.1022	0.1003	0.1127
0.0971	0.0986	0.0980	0.1010	0.1030	0.1022	0.0975	0.0984
0.0993	0.1000	0.0994	0.1006	0.1037	0.1010	0.0970	0.0996
0.1493	0.1152	0.1169	0.1197	0.1213	0.1170	0.1165	0.1202

8*8 DCT CHIP ACCURACY - SINGLE PRECISION MODE - 10,000 Blocks

IDCT CHARACTERISTICS
Pixel range : -128 to 127

Peak Mean Square Error : 0.0011000
Overall Mean Square Error : 0.0003719
Peak Mean Error : 0.0010000
Overall Mean Error : 0.0001250

Errors of -1 : 79 (0.01%)
Errors of 0 : 639762 (99.96%)
Errors of 1 : 159 (0.02%)

Mean Error Versus Pixel Location

0.0003	0.0001	0.0001	0.0000	0.0001	-0.0001	0.0001	0.0000
-0.0001	-0.0002	0.0001	0.0000	0.0002	0.0006	0.0000	0.0003
0.0001	0.0002	-0.0005	-0.0001	0.0001	0.0005	0.0002	0.0000
0.0006	-0.0001	0.0002	0.0000	0.0007	-0.0003	-0.0002	0.0010
0.0005	0.0000	0.0002	0.0004	0.0002	-0.0001	0.0002	0.0006
0.0001	-0.0002	0.0002	-0.0001	0.0003	-0.0002	0.0003	0.0003
0.0000	0.0002	0.0001	0.0003	-0.0001	-0.0001	0.0000	0.0000
0.0004	0.0001	-0.0002	0.0000	0.0001	0.0000	0.0003	0.0003

Mean Square Error Versus Pixel Location

0.0003	0.0001	0.0003	0.0004	0.0003	0.0003	0.0001	0.0000
0.0003	0.0002	0.0001	0.0004	0.0002	0.0006	0.0004	0.0003
0.0001	0.0002	0.0005	0.0005	0.0001	0.0007	0.0002	0.0004
0.0010	0.0003	0.0004	0.0004	0.0007	0.0005	0.0002	0.0010
0.0011	0.0002	0.0006	0.0010	0.0004	0.0003	0.0004	0.0008
0.0003	0.0004	0.0004	0.0003	0.0003	0.0004	0.0005	0.0003
0.0002	0.0004	0.0003	0.0003	0.0003	0.0001	0.0004	0.0002
0.0004	0.0003	0.0002	0.0002	0.0003	0.0002	0.0003	0.0005

8 DCT CHIP ACCURACY - SINGLE PRECISION MODE - 10,000 Blocks

FDCT CHARACTERISTICS
Pixel range : -128 to 127

Peak Mean Square Error : 0.1105000
Overall Mean Square Error : 0.0643234
Peak Mean Error : 0.0435000
Overall Mean Error : 0.0110203

Errors of -1 : 17057 (2.67%)
Errors of 0 : 598833 (93.57%)
Errors of 1 : 24110 (3.77%)

Mean Error Versus Pixel Location

0.0000	-0.0310	0.0082	-0.0245	0.0000	0.0169	0.0391	0.0352
-0.0274	-0.0025	0.0068	-0.0316	0.0361	0.0192	0.0414	0.0331
0.0113	-0.0053	0.0122	-0.0299	0.0301	0.0215	0.0383	0.0332
-0.0152	-0.0078	0.0084	-0.0312	0.0321	0.0183	0.0398	0.0313
0.0000	-0.0312	0.0109	-0.0245	0.0000	0.0107	0.0410	0.0372
0.0113	-0.0091	0.0062	-0.0382	0.0315	0.0188	0.0420	0.0363
0.0353	-0.0061	0.0112	-0.0326	0.0299	0.0132	0.0390	0.0281
0.0435	-0.0089	0.0113	-0.0320	0.0308	0.0194	0.0431	0.0311

Mean Square Error Versus Pixel Location

0.0000	0.0894	0.0574	0.0753	0.0000	0.0591	0.0429	0.0528
0.0524	0.1105	0.0794	0.0948	0.0567	0.0848	0.0748	0.0817
0.0345	0.0975	0.0660	0.0845	0.0437	0.0665	0.0583	0.0616
0.0392	0.0984	0.0702	0.0894	0.0437	0.0767	0.0616	0.0637
0.0000	0.0940	0.0541	0.0715	0.0000	0.0567	0.0470	0.0518
0.0355	0.0999	0.0628	0.0892	0.0445	0.0758	0.0634	0.0745
0.0435	0.0989	0.0648	0.0868	0.0403	0.0686	0.0624	0.0681
0.0575	0.1019	0.0721	0.0890	0.0494	0.0790	0.0691	0.0771

SGS-THOMSON MICROELECTRONICS

STV3200

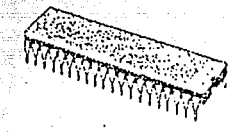
DISCRETE COSINE TRANSFORM (DCT)

ADVANCED DATA

- 0 TO 13.5 MHz OPERATING FREQUENCY EQUAL TO PIXEL RATE
- FORWARD OR INVERSE TRANSFORM
- 7 BLOCK SIZE POSSIBILITIES :

16 x 16	8 x 8	4 x 4
16 x 8	8 x 4	
8 x 16	4 x 8	

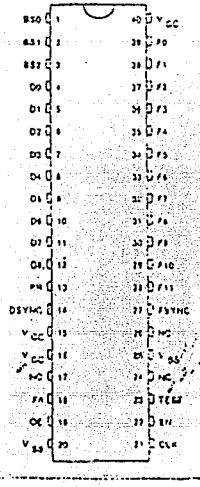
- 9-BIT TWO'S COMPLEMENT PIXEL FORMAT CORRESPONDING TO 3 POSSIBLE MAGNITUDES DEPENDING ON THE PIXEL RANGE PIN (PR) STATE :
  - 8-BIT UNSIGNED MAGNITUDE
  - 8-BIT 2's COMPLEMENT MAGNITUDE
  - 9-BIT 2's COMPLEMENT MAGNITUDE
- 12-BIT TWO'S COMPLEMENT COEFFICIENT FORMAT
- FULLY TTL AND CMOS COMPATIBLE
- CMOS TECHNOLOGY
- SINGLE + 5 VOLTS POWER SUPPLY
- POWER DISSIPATION : 500 mW AT 13.5 MHz



C
DIP40
(Ceramic Package)

(see the ordering information at the end of the datasheet)

PIN CONNECTIONS



DESCRIPTION

The STV3200 is a dedicated circuit for the discrete cosine transform (DCT) computation. The two-dimensional forward DCT (FDCT) or inverse DCT (IDCT) is performed for various block sizes and a pixel rate up to 13.5 MHz. The circuit architecture is fully bidirectional with a 9-bit magnitude pixel data bus and a 12-bit magnitude coefficient data bus programmed as input or output depending on the selection of FDCT or IDCT.

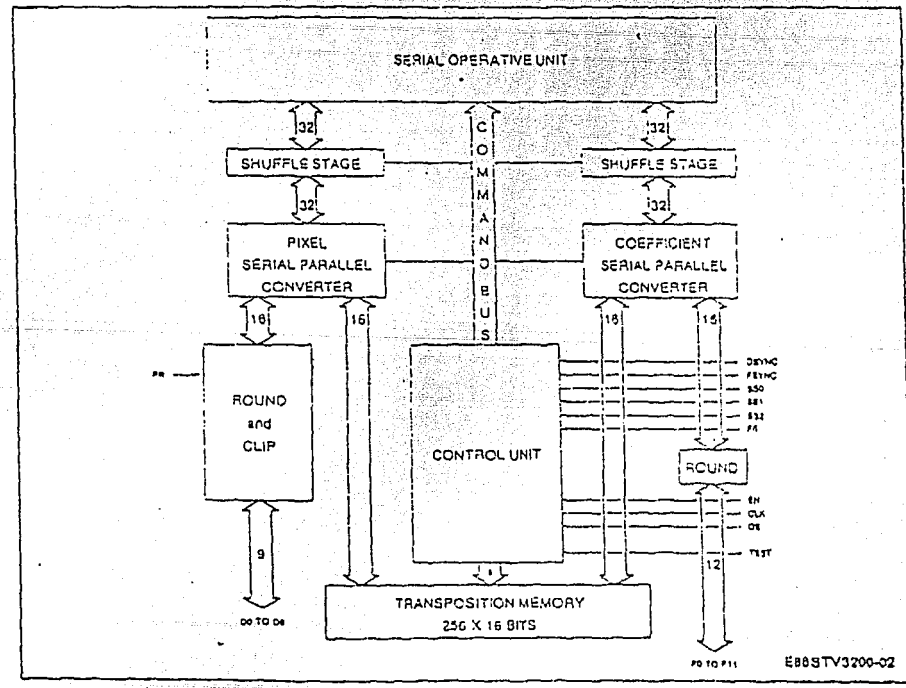
	FDCT	IDCT	Data Format
Pixel Bus	Input	Output	9-bit 2's Complement
Coefficient Bus	Output	Input	12-bit 2's Complement

For the forward transform, the input pixels are coded in 9-bit 2's complement and the output coefficients are coded in 12-bit 2's complement. For the inverse transform, the data format is identical with the coefficients used as input and the pixels used as output.

PIN NAME

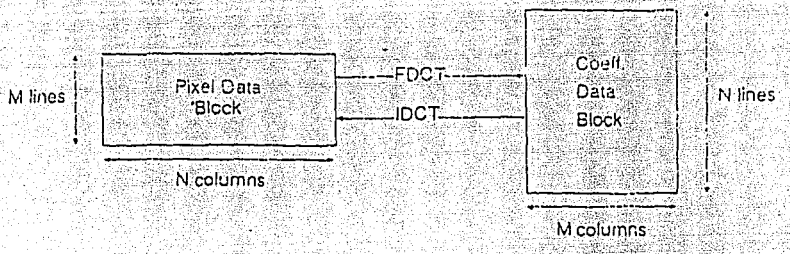
N°	Symbol	Direction	Function
1:3	BS0 to BS2	IN	Block Size Selection
4:12	D0 to D8	IN/OUT	Pixel Data Bus
13	PR	IN	Pixel Range Selection
14	DSYNC	IN/OUT	Pixel Block Synchronization
18	F/I	IN	Forward or Inverse Transform Selection
19	OE	IN	Tristate Output Control
20-25	VSS		Ground
21	CLK	IN	Clock Input
22	EN	IN	Clock Enable
23	TEST	IN	Test Mode
27	FSYNC	IN/OUT	Coefficient Block Synchronization
28:39	F0 to 11	IN/OUT	Coefficient Data Bus
40-15-16	VCC		+ 5 V ± 10 %
17-24-26	NC		Not Connected

FUNCTIONAL BLOCK DIAGRAM



FUNCTIONAL DESCRIPTION

1. EQUATIONS



The STV3200 performs a Two dimensional Discrete Cosine Transform according to the following equations, where the block size is defined by M lines and N columns of pixels :

FORWARD TRANSFORM EQUATION :

$$F(u,v) = \text{Round}\left[\frac{32}{N \cdot M} C^2(u)\, C^2(v) \sum_{i=0}^{M-1} \sum_{j=0}^{N-1} D(i,j) \cos\frac{(2i+1)\,u\,\pi}{2 \cdot M} \cos\frac{(2j+1)\,v\,\pi}{2 \cdot N}\right]$$

where $c^2(u) = 1/2$ if $u = 0$
$= 1$ otherwise

INVERSE TRANSFORM EQUATION :

$$D(i,j) = \text{Round}\left[\frac{1}{8} \sum_{v=0}^{N-1} \sum_{u=0}^{M-1} F(u,v) \cos\frac{(2i+1)\,u\,\pi}{2 \cdot M} \cos\frac{(2j+1)\,v\,\pi}{2 \cdot N}\right]$$

2. BLOCK FORMAT

M·N is the block size. This means that pixel blocks contain N columns of M pixels. The STV3200 performs a block transposition, and therefore the coefficient blocks contain M columns of N pixels.

The seven different possible block sizes are : 16 x 16, 8 x 16, 16 x 8, 8 x 8, 4 x 8, 8 x 4 and 4 x 4.

3. BLOCK SCANNING

Many possible arrangements for pixel block scanning are possible. These different arrangements are :

A - the block is entered line by line from the top line to the bottom line. Each line is entered from the left pixel to the right pixel.

B - the block is entered line by line from the top line to the bottom line. Each line is entered from the right pixel to the left pixel.

C - the block is entered line by line from the bottom line to the top line. Each line is entered from the left pixel to the right pixel.

D - the block is entered line by line from the bottom line to the top line. Each line is entered from the right pixel to the left pixel.

E - the block is entered column by column from the left column to the right column. Each column is entered from the top pixel to the bottom pixel.

F - the block is entered column by column from the left column to the right column. Each column is entered from the bottom pixel to the top pixel.

G - the block is entered column by column from the right column to the left column. Each column is entered from the top pixel to the bottom pixel.

H - the block is entered column by column from the right column to the left column. Each column is entered from the bottom pixel to the top pixel.

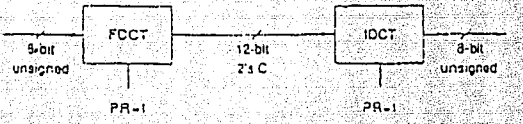
4. DATA FORMAT

The coefficient format is 12-bit 2's Complement, corresponding to the range - 2048 to 2047.

There are 3 possible ranges for pixel data :

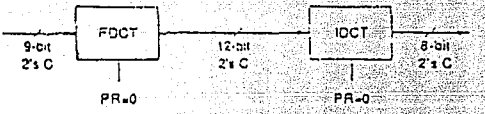
8-BIT UNSIGNED PIXEL MAGNITUDE

The pixel data range is 0 to 255. In this case DB is always equal to 0 and the PR pin must be set to 1 for FDCT and IDCT. A clipping to the range 0 to 255 is performed before outputting the reconstructed pixels after an IDCT.



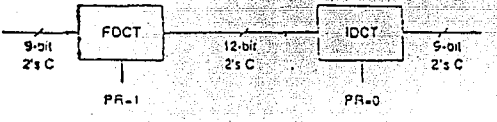
8-BIT TWO'S COMPLEMENT MAGNITUDE

The input pixel data range is – 256 to 255 and a clipping to the range – 128 to 127 is performed internally just before the FDCT. In this case, the PR pin must be set to 0 for FDCT and IDCT.



9-BIT TWO'S COMPLEMENT MAGNITUDE

The input pixel data range is – 256 to 255 and no clipping is performed. In this case, the PR pin must be set to 1 for FDCT and 0 for IDCT. Internal overflows may occur leading to aberrant errors on reconstructed values.



5. BLOCK FLOW

Depending on the application, blocks may be entered in different ways.

Latent period : the latent period between input data and the corresponding output results in 130 + M·N cycles. This means that the first data item of a resulting block is provided 130 + M·N clock cycles after the first data item of the corresponding input block.

Synchronization signals : an input block synchronization signal must be provided. The input pin for this signal is DSYNC if FDCT is selected and FSYNC if IDCT is selected. This signal must be active with the first data item of each input block or group of blocks.

An output block synchronization signal is provided. The output pin for this signal is FSYNC if FDCT is selected and DSYNC if IDCT is selected. This signal is active with the first data item of each output block or group of blocks.

The output synchronization signal is equal to the input synchronization signal delayed from 130 + M·N clock cycles.

CONTINUOUS BLOCK FLOW

Input data is entered continuously with one new item data at each clock cycle and output data is provided continuously with one new result data item at each clock cycle.

The input synchronization pulse can be provided for each input block. In this case the output synchronization pulse is provided for each output block (figure 3). Another way is to provide a synchronization pulse only for the first block. In this case, only one synchronization pulse is provided for the first output block (figure 4).

Figure 3 : Continuous Block Flow-1.

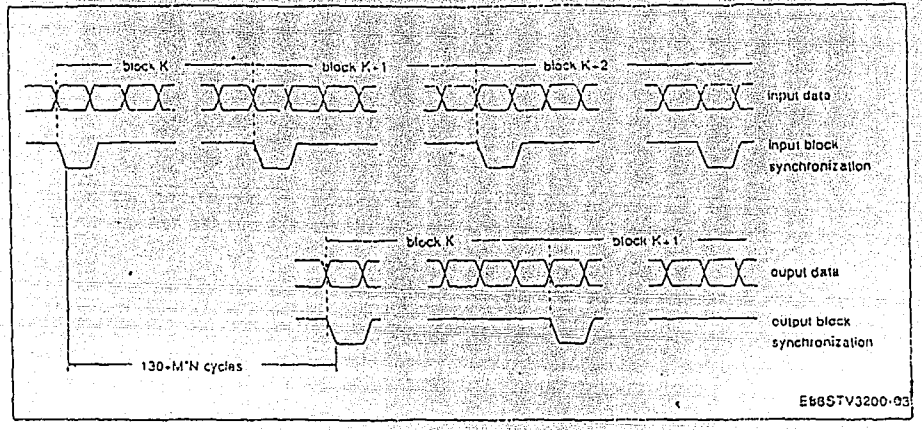
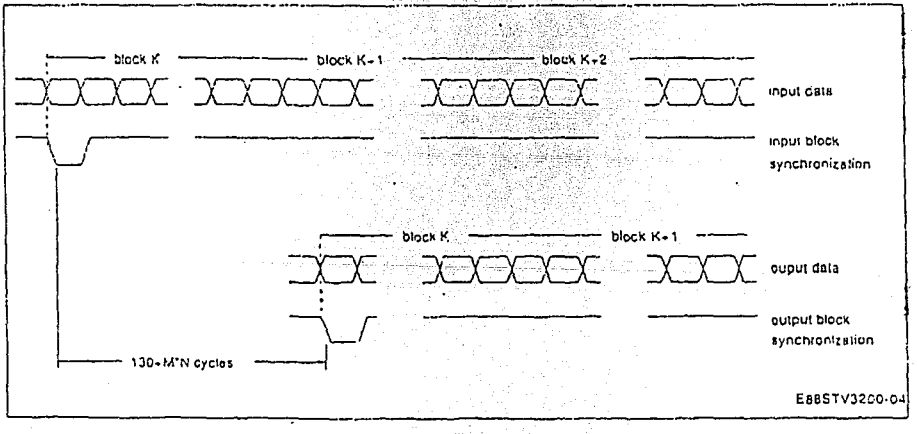


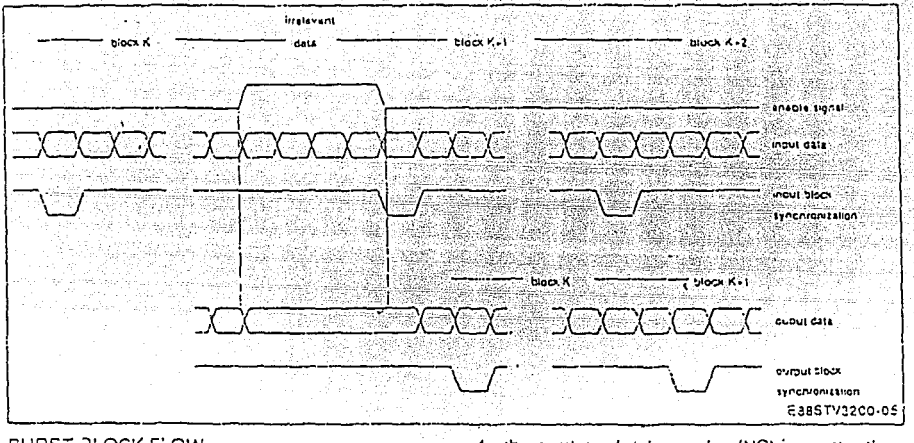
Figure 4 : Continuous Block Flow-2.



CONTINUOUS BLOCK FLOW WITH BYPASS OF IRRELEVANT DATA

It is possible to process a block flow including irrelevant data (corresponding to line suppresssion for example) as if it was a continuous block flow. One way is to stop the clock signal during the irrelevant data occurrence. Another way is to use the Clock Enable Signal (EN) to inhibit the chip internal clock during irrelevant data occurrence (figure 5).

Figure 5 : Continuous Block Flow with Irrelevant Data.



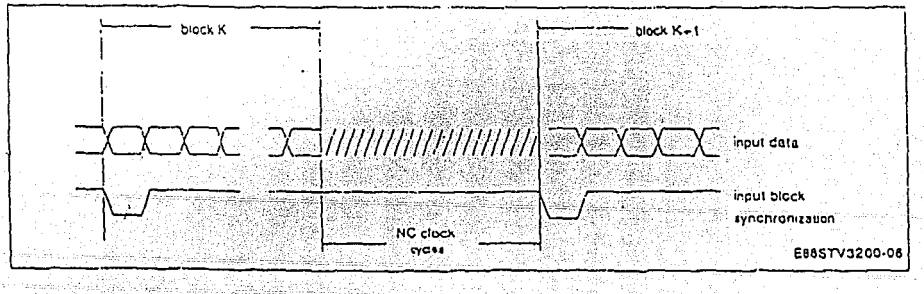
BURST BLOCK FLOW

Single blocks (or groups of blocks) may not be contiguous. In other words, delay cycles between two blocks (or groups of blocks) may exist. During these delay cycles, the clock is still running and the chip continues to perform computations. The constraint is that the internal pipe line must not be broken when a new block occurs. To take this constraint into account, the number of delay cycles (NC) must respect one of the following conditions :

1 - the number of delay cycles (NC) is greater than or equal to 130 + M-N. In this case the pipe line is empty (all the relevant data has been output ted) when a new input block processing starts.

2 - the number of delay cycles (NC) is a multiple of the number of cycles required to enter a block (M-N). In this case, the input data always remains synchronous with the internal pipe line.

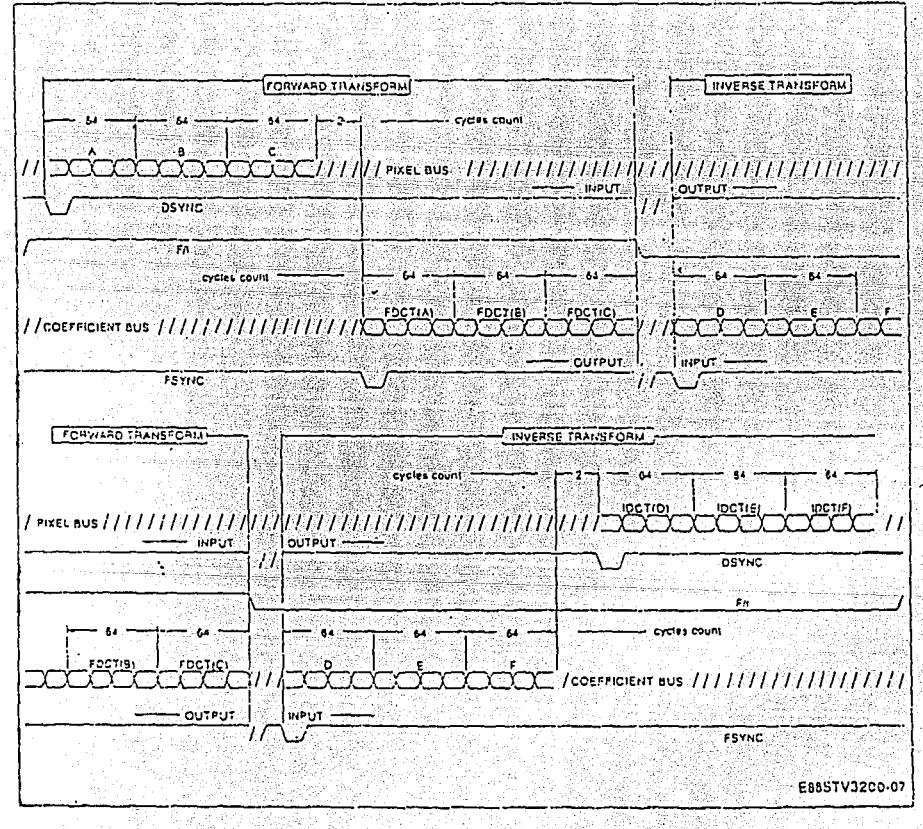
Figure 6 : Burst Block Flow.



MIXED FDCT/IDCT

In some low frequency applications, it could be useful to use only one chip to compute all the FDCT and IDCT required by the coding scheme. Blocks must be entered in a burst fashion with at least 130 + M·N delay cycles between the last item of the set of input pixels for FDCT and the first item of input coefficients for IDCT. The same delay must be respected between the last item of input coefficients for IDCT and the first pixel of input pixels for FDCT.

Figure 7 : Mixed 8x8 FDCT/IDCT Example Waveforms.



6. PINS DESCRIPTION

CLK : Clock-signal

DATA PINS

D0 TO D8 : 9-bit bidirectional Pixel data bus pins. Direction is programmed by the F/I pin :

F/I State	D0 to D8 Direction
High	Input
Low	Output

Data is loaded (when input) on the falling edge of CLK or settled (when output) on the rising edge of CLK. D0 is the least significant bit and D8 the most significant one.

MSB LSB

D8	D7	D6	D5	D4	D3	D2	D1	D0	Pin
- 256	128	64	32	16	8	4	2	1	Weight

DSYNC : Pixel data block synchronization signal. This pin is bidirectional with the direction programmed by the F/I pin (like D0 to D8). DSYNC is active (low level) with the first pixel data of a block (or group of blocks).

F0 TO F11 : 12-bit bidirectional Coefficient data bus pins. Direction is programmed by the F/I pin :

F/I State	F0 to F11 Direction
High	Output
Low	Input

Data is loaded (when input) on the falling edge of CLK or settled (when output) on the rising edge of CLK. F0 is the least significant bit and F11 the most significant one.

MSB LSB

F11	F10	F9	F8	F7	F6	F5	F4	F3	F2	F1	F0	Pin
- 2048	1024	512	256	128	64	32	16	8	4	2	1	Weight

FSYNC : Coefficient data block synchronization signal. This pin is bidirectional with the direction programmed by the F/I pin like F0 to F11. FSYNC is active (low level) with the first coefficient data of block (or group of blocks).

CONTROL PINS

F/I : Forward or inverse selection. When F/I is high, forward DCT is performed. When F/I is low, inverse DCT is performed.

BS0 to BS2 : Block size selection. The block size is programmed through these three pins according to the following table :

BS0	BS1	BS2	Pixel Block Size	Coefficient Block Size
0	0	0	16*16	16*16
0	0	1	8*16	16*8
0	1	0	16*8	8*16
0	1	1	8*8	8*8
1	0	0	4*8	8*4
1	0	1	8*4	4*8
1	1	0	4*4	4*4
1	1	1	Reserved	

PR : Pixel range selection. This pin controls the clipping of the pixel data. If PR is high, output pixels of an IDCT are clipped to the range 0 to 255. If PR is low input pixels of an FDCT are clipped to the range – 128 to 127.

OE : Output enable. This signal is active low. When OE is high, all outputs (defined by the F/I pin state) are forced to the high impedance state.

EN : Enable. This signal is active low. When EN is high, internal states of the chip are frozen. When EN becomes high, execution restarts.

POWER SUPPLY AND GROUND PINS

VCC : + 5 Volt power supply

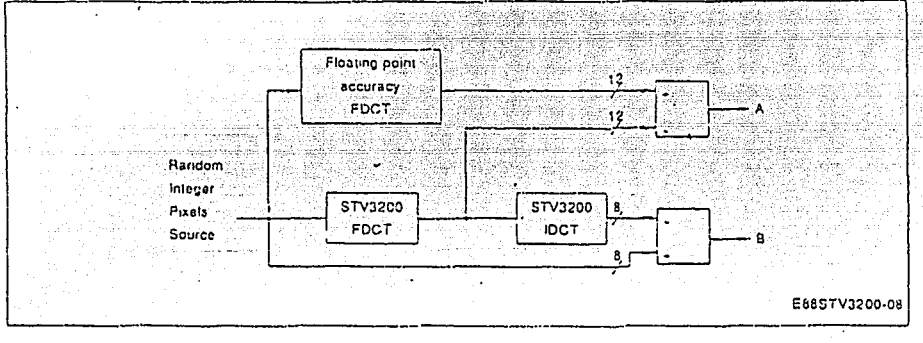
VSS : ground

OTHERS

TEST : test control. This pin is reserved and must be low in normal mode.

7. ACCURACY CHARACTERISTICS

The accuracy characteristics have been measured according to the following scheme :



A : characteristics of FDCT for 8-bit magnitude random pixel data. Error between the FDCT computed with floating point accuracy and the FDCT computed by the STV3200 is measured.

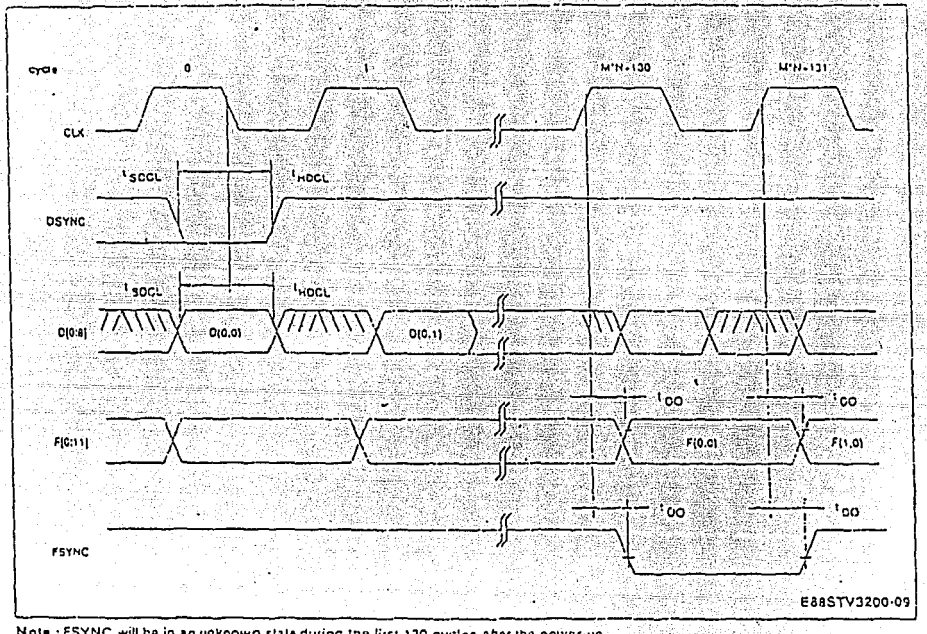
Block Size	16*16	8*16	16*8	8*8	4*8	8*4	4*4
Exact Value	86.8 %	88.4 %	86.9 %	88.4 %	89.4 %	88.8 %	91.0 %
Error of ± 1 LSB	13.0 %	11.4 %	13.0 %	11.5 %	10.5 %	11.1 %	8.9 %
Error of ± 2 LSB	0.23 %	0.19 %	0.14 %	0.11 %	0.11 %	0.07 %	0.11 %

B : characteristics of FDCT followed by an IDCT for 8-bit magnitude random pixel data. Error between the source picture and the FDCT computed by the STV3200 followed by an IDCT computed by the STV3200 is measured.

Block Size	16*16	8*16	16*8	8*8	4*8	8*4	4*4
Exact Value	82.2 %	93.9 %	93.0 %	99.0 %	99.9 %	99.9 %	100 %
Error of ± 1 LSB	17.8 %	6.1 %	7.0 %	1.0 %	0.06 %	0.06 %	0 %
Error of ± 2 LSB	0.02 %	0 %	0 %	0 %	0 %	0 %	0 %

TIMING WAVEFORMS

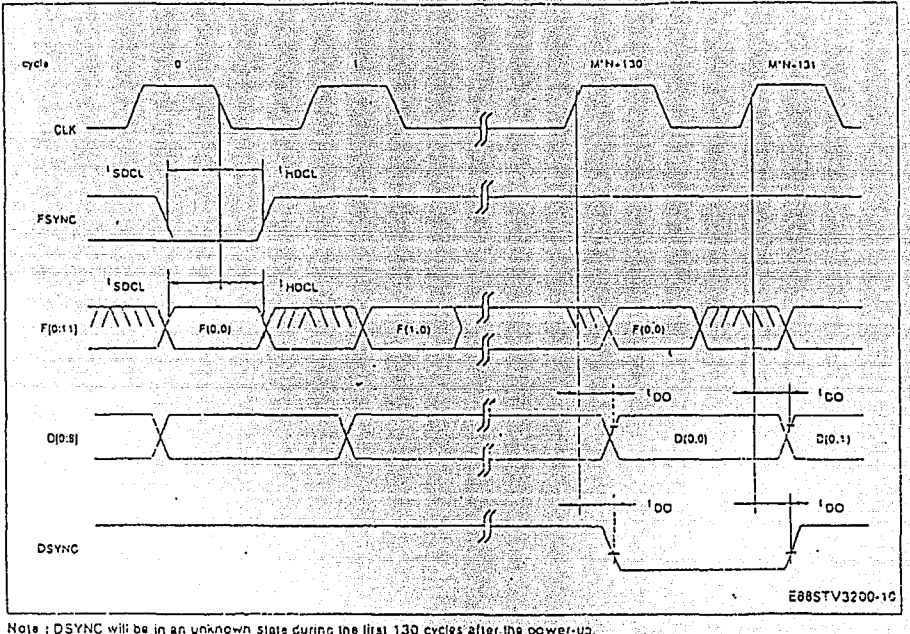
Sync Signals Timing Diagram for a Forward Transform on a M*N Block.



Note : FSYNC will be in an unknown state during the first 130 cycles after the power-up.

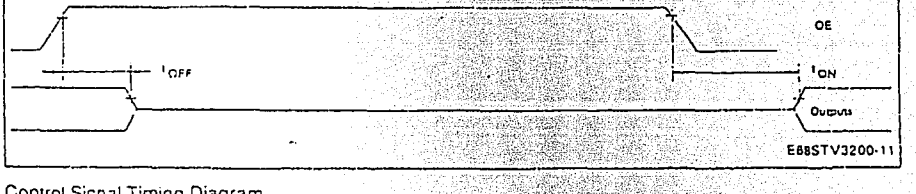
TIMING WAVEFORMS (continued)

Sync Signals Timing Diagram for a Inverse Transform on a M*N Block.

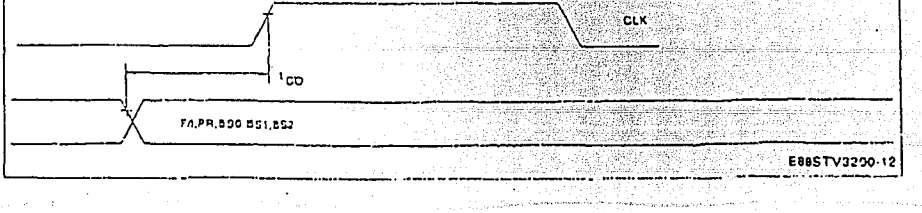


Note : DSYNC will be in an unknown state during the first 130 cycles after the power-up.

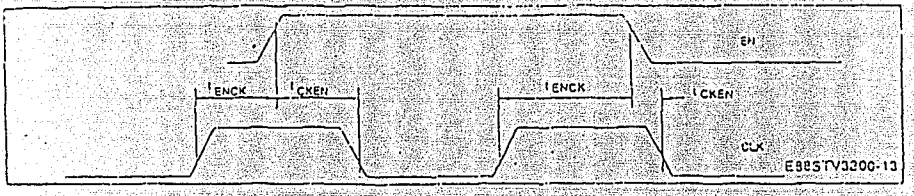
Outputs Enable Signal Timing Diagram.



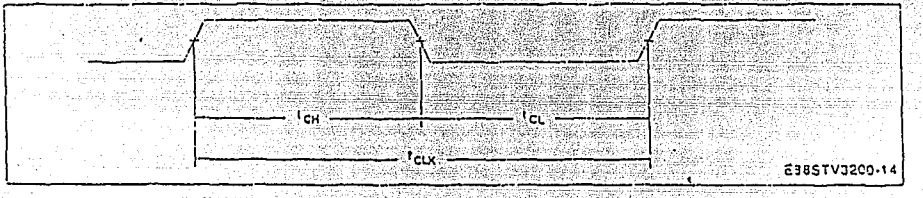
Control Signal Timing Diagram.



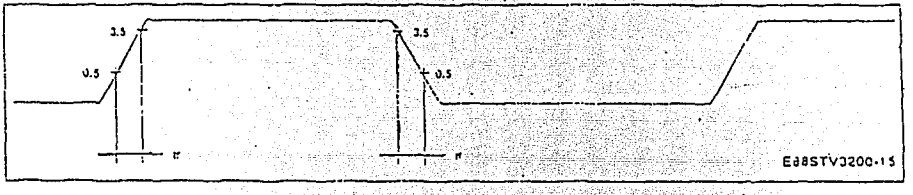
Enable Signal Timing Diagram.



Clock Timing Diagram.



Output Timing Diagram.



ELECTRICAL CHARACTERISTICS

ABSOLUTE MAXIMUM RATINGS

Supply voltage (V_{CC}) : 6 Volts
Operating Temperature Range : 0 to 70 °C
Voltage on Any Pin Relative to V_{SS} : − 0.5 to V_{CC} + 0.5 Volts

DC ELECTRICAL CHARACTERISTICS

Operating conditions :

V_{SS} = 0 Volt
T_A = 0 to 70 °C
V_{CC} = 5V ± 10 % unless otherwise noted

Symbol	Parameter	Test Conditions	Min.	Max.	Unit
V_{CC}	Operating Voltage		4.5	5.5	V
	Power Supply Ripple			0.5	V
I_{CC}	Supply Current	Cload = 50 pF on all Output.			
	Fclk = 13.5 Mhz	all Inputs		100	mA
	Fclk = 0 Mhz	at V_{CC} or V_{SS}		1	mA
	Input Voltage Level (all Inputs)	V_{CC} = 5 ± 0.5			
V_{IL}	Logic Low			0.8	V
V_{IH}	Logic High		2		V
	Hi - Z Input Leakage In/out Buffers	V_{IN} = V_{SS} to	− 5	+ 5	µA
	Input Buffers	V_{CC}	− 1	+ 1	µA
	Output Voltage Level (all outputs)	V_{CC} = 4.5 V			
V_{OL}	Logic Low Iload = 6.4 mA			0.4	V
V_{OH}	Logic High Iload = − 1 mA		2.7		V
	Output Current Level (all outputs)	V_{CC} = 5 V			
I_{OL}	Source V_{OUT} = 0 V		− 25		mA
I_{OH}	Sink V_{OUT} = 5 V		45		mA
C_{in}	Input Capacitance (DIL PACKAGE)				
	D [0 : 8] F [0 : 11] Dsync Fsync	V offset = 2.5 V		12	pF
	all Others	F = 1 Mhz		10	

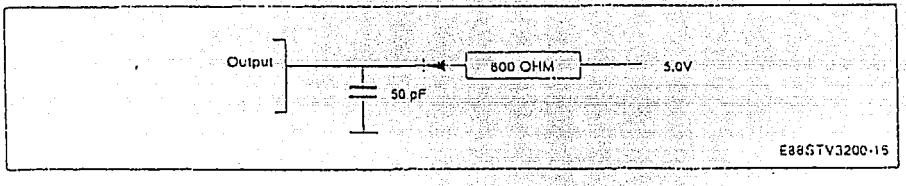
AC ELECTRICAL CHARACTERISTICS

Operating Conditions :

V_{SS} = 0 Volt
T_A = 0 to 70 °C
V_{CC} = 5 V ± 10 % unless otherwise noted

Outputs Load : capacitance = 50 pF
current logic low = 6.4 mA

Test Load on All Outputs :

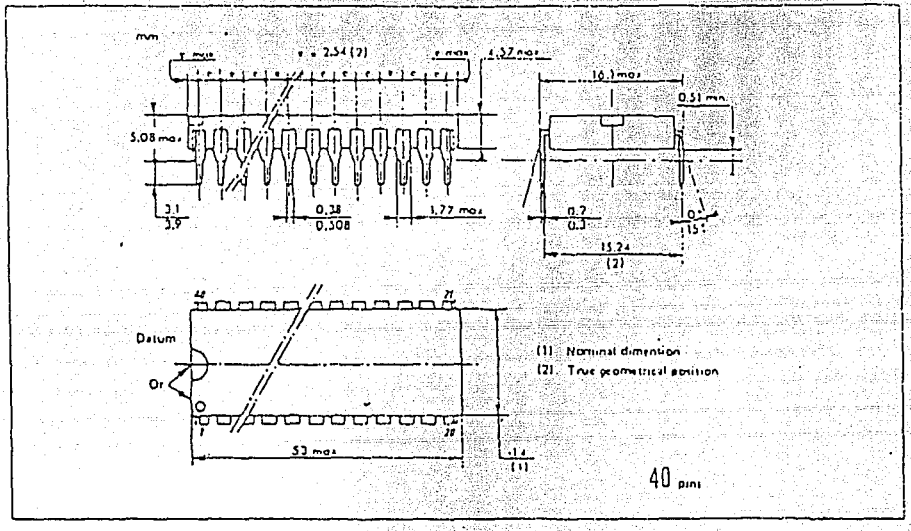


Timings are measured between threshold voltage of 1.5 V unless otherwise specified.

Symbol	Parameter	Min.	Max.	Unit
t_R	Rise Time from 0.5 to 3.5 V		10	ns
t_F	Fall Time from 3.5 to 0.5 V		10	ns
t_{CH}	Clock High Pulse Width	30		ns
t_{CL}	Clock Low Pulse Width	30		ns
t_{CLK}	Clock Cycle	60		ns
t_{SDCL}	Data Setup Time from CLK ↓	5		ns
t_{HDCL}	Data Hold Time from CLK ↓	20		ns
t_{DO}	Output Data Delay from CLK ↑		33	ns
t_{ENCK}	Enable Hold from CLK ↑	0		ns
t_{CKEN}	Enable Setup from CLK ↓	5		ns
t_{OFF}	Delay from OE ↑ to Output going to High Impedance State		20	ns
t_{ON}	Delay from OE ↓ to Output going to High or Low State		20	ns
t_{CO}	Control Signal Setup from beginning of Input Stream	100		ns

PACKAGE MECHANICAL DATA

40 PINS - CERAMIC PACKAGE



ORDER CODES

Part Number	Temperature Range	Package
STV3200 CC	0 to 70 °C	DIL Ceramic 40 Pins

A P E N D I C E

I I

MEMORIAS RAM ESTATICAS FABRICADAS POR INMOS

Y QUE SON UTILIZADAS PARA EL STV 3200

ZEROPOWERS

ORGANISATION DESCRIPTION	PART NUMBER	SPEEDns	ICCma ACTIVE mA@ns	ICCma TTL STBY	ICCma CMOS STBY	VCC	TEMP RANGE	PACKAGE
- 2KX8	MK48Z02	120, 150, 200, 250	90	3	1	5V - 10 - 5%	0 to + 70°C	P DIP 24
- 2KX8 UL CERTIFIED	MK48Z02BU	120, 150, 200, 250	90	3	1	5V - 10 - 5%	0 to + 70°C	P DIP 24
- 2K 8	MK48Z12	120, 150, 200, 250	90	3	1	5V - 10 - 10%	0 to + 70°C	P DIP 24
- 2KX8 UL CERTIFIED	MK48Z12BU	120, 150, 200, 250	90	3	1	5V - 10 - 10%	0 to + 70°C	P DIP 24
- 2KX8	MKI48Z02	120, 150, 200, 250	90	3	1	5V - 10 - 5%	- 40 to + 85°C	P DIP 24
- 2KX8 UL CERTIFIED	MKI48Z02BU	120, 150, 200, 250	90	3	1	5V - 10 - 5%	- 40 to + 85°C	P DIP 24
- 2KX8	MKI48Z12	120, 150, 200, 250	90	3	1	5V - 10 - 10%	- 40 to + 85°C	P DIP 24
- 2KX8 UL CERTIFIED	MKI48Z12BU	120, 150, 200, 250	90	3	1	5V - 10 - 10%	- 40 to + 85°C	P DIP 24
- 2KX8 W/O BATTERY	MK48C02AN	150, 200, 250	90	3	1	5V - 10 - 5%	0 to + 70°C	P DIP 24
- 2KX8 W/O BATTERY	MK48C02AK	150, 200, 250	90	3	1	5V - 10 - 5%	0 to + 70°C	PLCC 32
- 8KX8 UL CERTIFIED	MK48Z08	55,70 100, 150, 200	125@70 80@100	3	3	5V - 10 - 10%	0 to + 70°C	P DIP 28
- 8KX8 UL CERTIFIED	MK48Z08BU	55,70 100, 150, 200	125@70 80@100	3	3	5V - 10 - 10%	0 to + 70°C	P DIP 28
- 8KX8	MK48Z18	55,70 100, 150, 200	125@70 80@100	3	3	5V - 10 - 10%	0 to + 70°C	P DIP 28
- 8KX8 UL CERTIFIED	MK48Z18BU	55,70 100, 150, 200	125@70 80@100	3	3	5V - 10 - 10%	0 to + 70°C	P DIP 28
- 8KX8 PWR FAIL INT	MK48Z09	55,70 100, 150, 200	125@70 80@100	3	3	5V - 10 - 10%	0 to + 70°C	P DIP 28
- 8KX8 UL CERTIFIED PWR FAIL INT	MK48Z09BU	55,70 100, 150, 200	125@70 80@100	3	3	5V - 10 - 10%	0 to + 70°C	P DIP 28
- 8KX8 PWR FAIL INT	MK48Z19	55,70 100, 150, 200	125@70 80@100	3	3	5V - 10 - 10%	0 to + 70°C	P DIP 28
- 8KX8 UL CERTIFIED, PWR FAIL INT	MK48Z19BU	55,70 100, 150, 200	125@70 80@100	3	3	5V - 10 - 10%	0 to + 70°C	P DIP 28
- 32KX8 10 YEARS -25°C	MK48Z30	100, 120, 150	90	5	2	5V - 10 - 5%	0 to + 70°C	P DIP 28
- 32KX8 10 YEARS -25°C	MK48Z30A	100, 120, 150	90	5	2	5V - 10 - 5%	0 to + 70°C	P DIP 28
- 32KX8 10 YEARS -70°C	MK48Z32	100, 120, 150	90	5	2	5V - 10 - 5%	0 to + 70°C	P DIP 28

TIMEKEEPERS

ORGANISATION DESCRIPTION	PART NUMBER	SPEEDns	ICCma ACTIVE mA@ns	ICCma TTL STBY	ICCma CMOS STBY	VCC	TEMP RANGE	PACKAGE
- 2KX8	MK48T02	120, 150, 200, 250	80	5	3	5V + 10 - 5%	0 to + 70°C	P DIP 24
- 2KX8 UL CERTIFIED	MK48T02BU	120, 150, 200, 250	80	5	3	5V + 10 - 5%	0 to + 70°C	P DIP 24
- 2KX8	MK48T12	120, 150, 200, 250	80	5	3	5V + 10 - 10%	0 to + 70°C	P DIP 24
- 2KX8 UL CERTIFIED	MK48T12BU	150, 200, 250	80	5	3	5V + 10 - 10%	0 to + 70°C	P DIP 24
- 2KX8	MK48T02	120, 150, 200, 250	80	5	3	5V + 10 - 5%	- 40 to + 85°C	P DIP 24
- 2KX8 UL CERTIFIED	MK48T02BU	120, 150, 200, 250	80	5	3	5V + 10 - 10%	- 40 to + 85°C	P DIP 24
- 8KX8	MK48T08	100, 150, 200	80	3	3	5V + 10 - 5%	0 to + 70°C	P DIP 24
- 8KX8	MK48T18	100, 150, 200	80	3	3	5V + 10 - 10%	0 to + 70°C	P DIP 24
- PC REAL TIME CLOCK	MK48T18-10	100	15	NA	NA	5V + 10 - 10%	0 to + 70°C	P DIP 24
- PC REAL TIME CLOCK W/O BATTERY	MK48T85O	100	15	0.5	NA	5V + 10 - 10%	0 to + 70°C	PLCC 28

CACHETAG MEMORIES

ORGANISATION DESCRIPTION	PART NUMBER	SPEEDns	ICCma ACTIVE mA@ns	ICCma TTL STBY	ICCma CMOS STBY	VCC	TEMP RANGE	PACKAGE
- 4KX4	MK41H80	20,22, 25, 35	120	NA	NA	5V + 10 - 10%	0 to + 70°C	P DIP 24
- 4KX4	MK41S80	12, 15	120	NA	NA	5V + 10 - 10%	0 to + 70°C	P DIP 24 SOJ 24
- 2KX20	MK4202	20, 22, 25	250	50	NA	5V + 10 - 10%	0 to + 70°C	PLCC 28
- 8KX8	MK48S74	20, 22, 25	150	NA	NA	5V + 10 - 10%	0 to + 70°C	P DIP 24 SOJ 24
8KX8 Totem Pole Output	MK48S80	20, 22, 25	150	NA	NA	5V + 10 - 10%	0 to + 70°C	P DIP 24 SOJ 24
16KX4	MK44S80	15, 17, 20	150	NA	NA	5V + 10 - 10%	0 to + 70°C	P DIP 24 300 (1)

FIFO's

ORGANISATION DESCRIPTION	PART NUMBER	SPEEDns	ICCma ACTIVE mA@ns	ICCma TTL STBY	ICCma CMOS STBY	VCC	TEMP RANGE	PACKAGE
- 512X9	MK4501	65, 80, 100, 120 150 & 200	80	8	NA	5V + 10 - 10%	0 to + 70°C	P DIP 28 PLCC 32
- 512X9 FAST	MK45H01	25, 35, 50, 65, 120	120	12	NA	5V + 10 - 10%	0 to + 70°C	P DIP 28 PLCC 32
- 1KX9 FAST	MK45H02	25, 35, 50, 65, 120	120	12	NA	5V + 10 - 10%	0 to + 70°C	P DIP 28 PLCC 32
- 2KX9	MK4503	65, 80, 100, 120 150 & 200	120	12	NA	5V + 10 - 10%	0 to + 70°C	P DIP 28 PLCC 32
- 2KX9 FAST	MK45H03	25, 35, 50, 65, 120	120	12	NA	5V + 10 - 10%	0 to + 70°C	P DIP 28 PLCC 32
- 4KX9 FAST	MK45H04	25, 35, 50, 65, 120	120	12	NA	5V + 10 - 10%	0 to + 70°C	P DIP 28
- 8KX9 FAST	MK45H08	25, 35, 50, 65, 120	120	12	NA	5V + 10 - 10%	0 to + 70°C	P DIP 28
- 1KX9 clk 40MHz	MK4505 M	25, 33, 50	100	NA	NA	5V + 10 - 10%	0 to + 70°C	P DIP 28
MASTER & SLAVE	MK4505 S	25, 33, 50	100	NA	NA	5V + 10 - 10%	0 to + 70°C	P DIP 28
- 64X5X2	MK45264	55, 70	60	NA	NA	5V + 10 - 10%	0 to + 70°C	P DIP 24
BIDIRECTIONAL	MK45265	55, 70	60	NA	NA	5V + 10 - 10%	0 to + 70°C	P DIP 24

(1) 600 mil and 300 mil PDIP

STATIC RAM

ORGANISATION DESCRIPTION	PART NUMBER	SPEEDns	ICC ma ACTIVE @ ns	ICC ma TTL STBY	ICC ma CMOS STBY	ICC_{DR} µA	VCC	TEMP RANGE	PACKAGE
- 4KX1	IMS1203	25, 35, 45	90	15	10		5V + 10-10%	0 °C to 70 °C	P DIP 18
- 1KX4	IMS1223	25, 35, 45	100	15	8		5V + 10-10%	0 °C to 70 °C	P DIP 18
- 2KX8	MK6116	150, 200, 250	70	3	1	1	5V + 10-10%	0 °C to + 70 °C	P DIP 24 /SOIC 28
- 2KX8 LOW POWER	MK6116L	150, 200, 250	70	3	0.001	1	5V + 10-10%	0 °C to + 70 °C	P DIP 24 SOIC 28
- 2KX8	MK6116	150, 200, 250	55	3	1	1	5V + 10-10%	-40 °C to + 85 °C	P DIP 24 SOIC 28
- 2KX8 LOW POWER	MK6116L	150, 200, 250	55	3	0.01	1	5V + 10-10%	-40 °C to + 85 °C	P DIP 24 SOIC 28
- 16KX1 FAST CS	MK41H66	20, 25, 35	120	10			5V + 10-10%	0 °C to + 70 °C	P DIP 20
- 16KX1	IMS1403	25, 35, 45, 55	75	15	10		5V + 10-10%	0 °C to 70 °C	P DIP/LCC 20
- 16KX1	MK41H67	20, 25, 35	120	10	0.05	50	5V + 10-10%	0 °C to + 70 °C	P DIP 20
- 4KX4	MK41H68	20, 25, 35	120	8	0.05	50	5V + 10-10%	0 °C to + 70 °C	P DIP 20
- 4KX4	IMS1423	25, 35, 45, 55	105@25 100@35	15	0.01		5V + 10-10%	0 °C to 70 °C	P DIP/LCC 20 SOJ 20
- 4KX4 FAST CS	MK41H69	20, 25, 35	120	8			5V + 10-10%	0 °C to + 70 °C	P DIP 20
- 4KX4 OE	MK41H78	20, 25, 35	120	10	0.05	50	5V + 10-10%	0 °C to + 70 °C	P DIP 22
- 4KX4 OE + FLASH CLR	MK41H79	20, 25, 35	120	16	0.05	50	5V + 10-10%	0 °C to + 70 °C	P DIP 22
- 64KX1	IMS1600	25, 35, 45, 55	77@25 70@35	25	15		5V + 10-10%	0 °C to + 70 °C	P DIP/LCC 22 SOJ 24
- 64KX1	IMS1601	35, 45, 55	70	15	5	100	5V + 10-10%	0 °C to + 70 °C	P DIP/LCC 22 SOJ 24
- 64KX1	IMS1605	15, 20, 25	100	25	10	350	5V + 10-10%	0 °C to + 70 °C	P DIP/LCC 22 SOJ 24
- 16KX4	IMS1620	25, 35, 45, 55	110@25 100@35	25	17		5V + 10-10%	0 °C to + 70 °C	P DIP/LCC 22 SOJ 24
- 16KX4	IMS1625	15, 20, 25	100	25	10	350	5V + 10-10%	0 °C to + 70 °C	P DIP/LCC 22 SOJ 24
- 16KX4 OE	IMS1624	25, 35, 45, 55	110@25 100@35	25	17		5V + 10-10%	0 °C to + 70 °C	P DIP/SOJ 24 LCC 28
- 16KX4 OE	IMS1629	15, 20, 25	100	25	10	350	5V + 10-10%	0 °C to + 70 °C	P DIP/SOJ 24 LCC 28
- 16KX4 OE Sep I/O	IMS1626/7	15, 20, 25	100	25	10	350	5V + 10-10%	0 °C to + 70 °C	P DIP/SOJ 28 LCC 28
- 8KX8 OE	IMS1630L	45,55,70, 100, 120	90	20	10	350	5V + 10-10%	0 °C to + 70 °C	P DIP /SOIC 28

STATIC RAM (Continued)

ORGANISATION DESCRIPTION	PART NUMBER	SPEEDns	ICC ma ACTIVE @ ns	ICC ma TTL STBY	ICC ma CMOS STBY	ICC_{DR} µA	VCC	TEMP RANGE	PACKAGE
- 8KX8 OE	MK48H64	70, 120	100@70 90@120	5	500	500	5V + 10-10%	0 °C to + 70 °C	P DIP 28 /SOIC 28
- 8KX8 OE	MK48H64L	70, 120	100@70 90@120	5	0.05	25	5V + 10-10%	0 °C to + 70 °C	P DIP 28 /SOIC 28
- 8KX8 OE	IMS1635	15, 20, 25	100	25	10	350	5V + 10-10%	0 °C to + 70 °C	P DIP 28
- 8KX9 OE	MK48H89	20, 25, 35	120	25	1	500	5V + 10-10%	0 °C to + 70 °C	P DIP 28
- 8KX9 OE	IMS1695	15, 20, 25	100	25	10	350	5V + 10-10%	0 °C to + 70 °C	P DIP/SOJ 28 LCC 32
- 8KX8/9 PARITY	MK48H99/99	20, 30, 40	120	25	1	500	5V + 10-10%	0 °C to + 70 °C	P DIP 28
- 256KX1	IMS1600	25, 30, 35, 45	120	30	15		5V + 10-10%	0 °C to + 70 °C	P DIP/SOJ 24 LCC 28
- 64KX4	IMS1820	25, 30, 35, 45	120	30	15		5V + 10-10%	0 °C to + 70 °C	P DIP/SOJ 24 LCC 28
- 32KX8 OE	MK4832	70, 120	70	3	1	500	5V + 10-10%	0 °C to + 70 °C	P DIP 28
- 32KX8 OE	MK4832L	70, 120	70	3	0.05	20	5V + 10-10%	0 °C to + 70 °C	P DIP 28
- 128KX8	(1) MK48127	55, 70, 85	80	3	0.2	150	5V + 10-10%	0 °C to + 70 °C	P DIP 32 /SOJ 32
- 128KX8 CHIP ENABLE 2	(1) MK48128	55, 70, 85	80	3	0.2	150	5V + 10-10%	0 °C to + 70 °C	P DIP 32 /SOJ 32

(1) Product preview

SGS-THOMSON MICROELECTRONICS

MILITARY PRODUCTS

ORGANISATION DESCRIPTION	PART NUMBER	Speed ns	Active current (mA @ ns)	ICC max Stdby 1	Icc max Stdby 2	ICC_{DR} @ 3V	Vcc	Temp Range	Packages
4KX1	IMS1203M	25, 35, 45	80 mA	15 mA	10 mA		5 ± 10 %	- 55 to 125	DIP, F-PACK
1KX4	IMS1223M	25, 35, 45	110 mA	15 mA	10 mA		5 ± 10 %	- 55 to 125	DIP, F-PACK
16KX1	IMS1400M	45, 55, 70	120 mA	30 mA	(3)		5 ± 10 %	- 55 to 125	DIP, LCC
16KX1	IMS1403M	35, 45, 55	75 mA	15 mA	10 mA		5 ± 10 %	- 55 to 125	DIP, LCC
16KX1	IMS1403LM	35, 45, 55	75 mA	15 mA	10 mA	400 µA	5 ± 10 %	- 55 to 125	DIP, LCC
16KX1	MKB41H67	25, 35, 45	115 mA	10 mA	0,05 mA		5 ± 10 %	- 55 to 125	DIP 20
2KX8	MKB6116	150, 200, 250	70 mA	10 mA	0,1 mA		5 ± 10 %	- 55 to 125	DIP 24
512X9 (FIFO)	MKB4501	100, 120, 140, 175, 235	90 mA	8 mA	0,9 mA		5 ± 10 %	- 55 to 125	DIP 24
4KX4	IMS1420M	55, 70	140 mA	30 mA	(3)		5 ± 10 %	- 55 to 125	DIP, LCC
4KX4	IMS1425M	35, 45, 55	130 mA, 110 mA	20 mA	15 mA		5 ± 10 %	- 55 to 125	DIP, LCC, F-PACK
64KX1	IMS1600M	45, 55, 70	70 mA	25 mA	19 mA		5 ± 10 %	- 55 to 125	DIP, LCC
64KX1	IMS1603M	45, 55, 70	70 mA	20 mA	15 mA	1000 µA	5 ± 10 %	- 55 to 125	DIP, LCC
16KX4	IMS1620M	45, 55, 70	100 mA	30 mA	20 mA		5 ± 10 %	- 55 to 125	DIP, LCC
16KX4	IMS1620LM	45, 55, 70	100 mA	20 mA	8 mA	1200 µA	5 ± 10 %	- 55 to 125	DIP, LCC
16KX4	IMS1624M (1)	45, 55, 70	100 mA	30 mA	20 mA		5 ± 10 %	- 55 to 125	DIP, LCC
16KX4	IMS1624LM (1)	45, 55, 70	100 mA	20 mA	8 mA	1200 µA	5 ± 10 %	- 55 to 125	DIP, LCC
8KX8	IMS1630M	45, 55, 70	85 mA	30 mA	20 mA		5 ± 10 %	- 55 to 125	DIP, LCC
8KX8	IMS1630LM	45, 55, 70	85 mA	30 mA	20 mA	1200 µA	5 ± 10 %	- 55 to 125	DIP, LCC
64KX1	IMS1605M	20, 25, 35					5 ± 10 %	- 55 to 125	DIP, LCC
16KX4	IMS1625M	20, 25, 35					5 ± 10 %	- 55 to 125	DIP, LCC
16KX4	IMS1629M (1)	20, 25, 35					5 ± 10 %	- 55 to 125	DIP, LCC
16KX4	IMS1626M (1, 2)	20, 25, 35					5 ± 10 %	- 55 to 125	DIP, LCC
16KX4	IMS1627M (1, 2)	20, 25, 35					5 ± 10 %	- 55 to 125	DIP, LCC
8KX8	IMS1635M (1)	20, 25, 35					5 ± 10 %	- 55 to 125	DIP, LCC
8KX9	IMS1695M (1)	20, 25, 35					5 ± 10 %	- 55 to 125	DIP, LCC
256KX1	IMS1800M	30, 35, 45					5 ± 10 %	- 55 to 125	DIP, LCC
64KX4	IMS1820M	30, 35, 45					5 ± 10 %	- 55 to 125	DIP, LCC

(1) Output Enable

(2) Separate I/O

Stdby 1 : Stable Input, TTL Levels

Stdby 2 : Cycling Input, CMOS Levels

AMD	SGS-THOMSON Group
AM2147	IMS1203
AM2148/9	IMS1223
AM2167	IMS1403
AM2168	IMS1423 or MK41H68
AM9126	MK6116
AM99C68	IMS1423 or MK41H68
AM99C88	MK48H64
AM99C89	MK48H89
AM67C4501	MK4501/H01
AM67C4502	MK45H02
AM67C4503	MK4503/H03

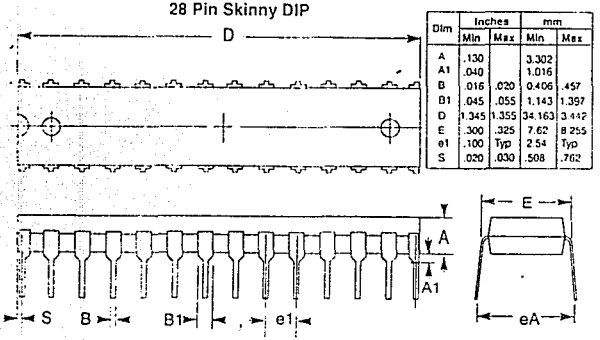
CYPRESS	SGS-THOMSON Group
CY2147	IMS1203
CY2148/9	IMS1223
CY7C147	IMS1203
CY7C148	IMS1223
CY7C164	IMS1620
CY7C166	IMS1624
CY7C167	IMS1403 or MK41H67
CY7C168	MK41H68
CY7C185	IMS1630L
CY7C187	IMS1600
CY7194	IMS1820
CY7C197	IMS1800
CYC412	MK4501/H01
CY7C424	MK45H02
CYC429	MK4503/H03

DALLAS Semiconductor	SGS-THOMSON Group
DS2009	MK4501/H01
DS2010	MK45H02
DS2011	MK4503/H03
DS2012	MK45H04
DS1210	MK48Z02
DS1225	MK48Z08
DS1243	MK48T08
DS1287	MK48T87
DS1235	MK48Z30
DS1230	MK48Z30A

FUJITSU	SGS-THOMSON Group
MB81C67	IMS1403 orMK41H67
MB81C68	IMS1423 or MK41H68
MB81C71	IMS1600
MB81C74	IMS1620
MB81C75	IMS1624
MB81C78	MK48H64

HARRIS	SGS-THOMSON Group
HM65747	IMS1203
HM65748	IMS1223
HM65767	IMS1403
HM65261	IMS1403
HM65768	IMS1423 or MK41H68
HM65787	IMS1600
HM65764	IMS1630L
HM6116L	MK6116/L
HM20256	MK4832

28 Pin Skinny DIP



Dim	Inches		mm	
	Min	Max	Min	Max
A	.130		3.302	
A1	.040		1.016	
B	.016	.020	0.406	.457
B1	.045	.055	1.143	1.397
D	1.345	1.355	34.163	3.442
E	.300	.325	7.62	8.255
e1	.100	Typ	2.54	Typ
S	.020	.030	.508	.762

inmos

IMS 1605: 64K x 1
IMS 1625: 16K x 4
IMS 1629: 16K x 4 with Output Enable
IMS 1626/7: 16K x 4 with Separate I/Os
IMS 1635: 8K x 8
IMS 1695: 8K x 9

IMS 16X5 Series High Performance Memory Products

Advance Information

FEATURES

- INMOS' Very High Speed Double Metal CMOS
- Advanced Process-1.2 Micron Design Rules
- 64K Bit Devices
- 15, 20 and 25 ns Address Access Times
- 15, 20 and 25 ns Chip Enable Access Times
- Fully TTL Compatible
- Single +5V ± 10% Operation
- Battery Backup Operation - 2V Data Retention, 10µA typical at 25°C
- Packages include: DIP, LCC and SOJ
- Military Versions Available

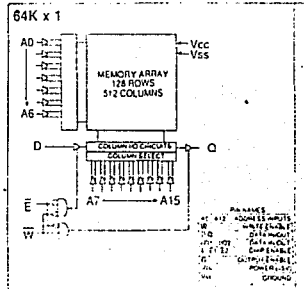
DESCRIPTION

The INMOS IMS16X5 series are high speed advanced 64K double layer metal CMOS Static RAMs.

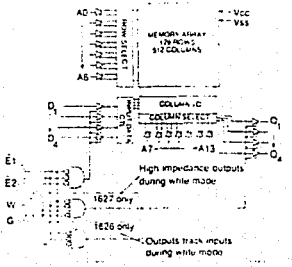
The range features fully static operation requiring no external clocks or timing strobes, with equal access and cycle times. A chip enable function (E) that can be used to place the device into a low-power standby mode is available on all organisations. The 8K x 8 organisations provide an additional Chip Enable for reduced low-power standby mode. Output Enable (G) is an enhancement on organisations requiring fast access to data and enhanced bus contention control.

Military versions of the 16X5 are also available.

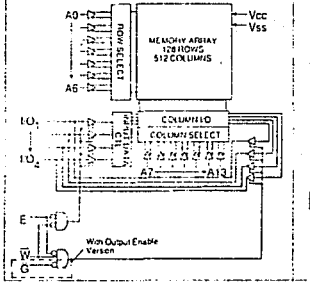
64K x 1



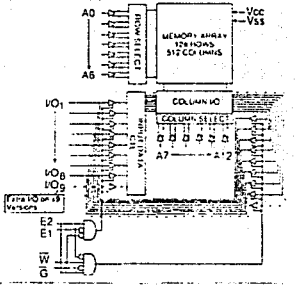
16K x 4 (Separate Inputs and Outputs)



16K x 4 (Without and with Output Enable)



8K x 8 / 8K x 9



2.1 Electrical specifications

2.1.1 Absolute maximum ratings

Symbol	Parameter	Min	Max	Unit
V_{SS}	Value on relative pin	-2.0	7.0	V
	Voltage on I/O	-1.0	V_{CC} +0.5	V
T_A	Temperature under bias	-55	125	°C
	Storage temperature	-65	150	°C
	Power dissipation		1	W
	DC output current		25	mA

(One output at a time, one second duration)

2.1.2 DC operating conditions

Symbol	Parameter	Min	Typ	Max	Units	Notes
V_{CC}	Supply Voltage	4.5	5.0	5.5	V	
V_{SS}	Supply Voltage	0	0	0	V	
V_{IH}	Input Logic '1' Voltage	2.0		V_{CC} + 0.5	V	All inputs
V_{IL}	Input Logic '0' Voltage	-0.5*		0.8	V	All inputs
T_A	Ambient Operating Temperature	0		70	°C	400 linear ft/min air flow

*V_{IL} min. = -3.0V for pulse width < 10ns. Note B

2.1.3 DC electrical characteristics ($0°C \leq T_A \leq 70°C$)($V_{CC} = 5.0V \pm 10\%$)[a]

For suffixes refer to section 2.1.7.

Symbol	Parameter	Min	Max	Units	Notes
I_{CC1}	Average V_{CC} Power Supply Current		100	mA	$t_{AVAV} = t_{AVAV}$(min).
I_{CC2}	V_{CC} Power Supply Current (Standby, Stable TTL Input Levels)		40	mA	$\overline{E}_1 \geq V_{IH}$ or $E_2 \leq V_{IL}$. All other inputs at $V_{IH} \leq V_{IN}$ or $\geq V_{IH}$.
I_{CC3}	V_{CC} Power Supply Current (Standby, Stable CMOS Input Levels)		2	mA	$\overline{E}_1 \geq (V_{CC} - 0.2V)$ or $E_2 \leq 0.2V$. All other inputs at $V_{IN} \leq 0.2$ or $\geq (V_{CC} - 0.2V$.
I_{CC4}	V_{CC} Power Supply Current (Standby, Cycling CMOS Input Levels)		25	mA	$\overline{E}_1 \geq (V_{CC} - 0.2V)$ or $E_2 \leq 0.2V$. Inputs cycling at $V_{IN} \leq 0.2$ or $\geq (V_{CC} - 0.2V$.
I_{LIX}	Input Leakage Current (any input)		±1	µA	V_{CC} = max. $V_{IN} = V_{SS}$ to V_{CC}.
I_{OLX}	Off State Output Leakage Current		±5	µA	V_{CC} = max. $V_{IN} = V_{SS}$ to V_{CC}.
V_{OH}	Output Logic '1' Voltage	2.4		V	$I_{OH} = -4mA$.
V_{OL}	Output Logic '0' Voltage		0.4	V	$I_{OL} = 8mA$.

[a] Stresses greater than those listed may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other conditions above those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect reliability.

2.1.4 AC test condition

Input pulse levels	V_{SS} to 3.0V
Input rise and fall times	5ns
Input and output timing reference levels	1.5V
Output load	see figure 2.1

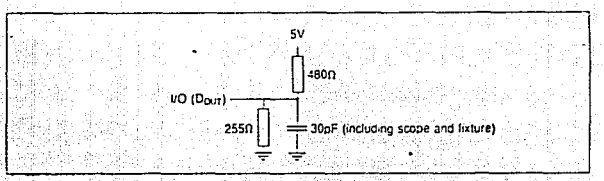


Figure 2.1 Output load

2.1.5 Capacitance[a]

Symbol	Parameter	Min	Max	Units	Conditions
C_{IN}	Input capacitance		5	pF	ΔV = 0 to 3.0V
C_{OUT}	Output capacitance		7	pF	ΔV = 0 to 3.0V

2.1.6 Recommended AC operating conditions ($0°C \leq T_A \leq 70°C$)($V_{CC} = 5.0V \pm 10\%$)

Read cycle[a]

No.	Symbol Stand.	Symbol Alt.	Parameter	16X5-15 min	16X5-15 max	16X5-20 min	16X5-20 max	16X5-25 min	16X5-25 max	Units	Notes*
1	t_{E1LQV}	t_{ACS}	Chip Enable Access Time		15		20		25	ns	
2	t_{E2HQV}	t_{ACS}	Chip Enable Access Time		15		20		25	ns	
3	t_{AVAV}	t_{RC}	Read Cycle Time	15		20		25		ns	c
4	t_{AVQV}	t_{AA}	Address Access Time		15		20		25	ns	d
5	t_{GLQV}	t_{OE}	O/P Enable to Data Valid		8		10		10	ns	
6	t_{AXQX}	t_{OH}	O/P Hold After Add's Chge	3		3		3		ns	
7	t_{E1LQX}[b]	t_{LZ}	Chip Enable to O/P Active	8		8		10		ns	
8	t_{E1HQZ}	t_{HZ}	Chip Disable to O/P Inactive		8		10		10	ns	f,j
9	t_{E2HQZ}	t_{LZ}	Chip Enable to O/P Active	8		10		10		ns	
10	t_{E2LQZ}	t_{HZ}	Chip Disable to O/P Inactive		8		10		10	ns	f,j
11	t_{GLQX}	t_{LZ}	O/P Enable to O/P Active	3		3		3		ns	
12	t_{GHQZ}	t_{HZ}	O/P Disable to O/P Inactive		8		10		10	ns	f,j
13	$t_{E1LICCH}$	t_{PU}	Chip Enable to Power Up	0		0		0		ns	j
14	$t_{E1HICCL}$	t_{PD}	Chip Disable to Power Down		15		20		25	ns	j
15	$t_{E2HICCH}$	t_{PU}	Chip Enable to Power Up	0		0		0		ns	j
16	$t_{E2LICCL}$	t_{PD}	Chip Disable to Power down		15		20		25	ns	j
17		t_T	I/P Rise and Fall Times		50		50		50	ns	e,j

* Refer to section 2.1.7.

[b] T_{ELQX} is always greater than T_{EHQZ}

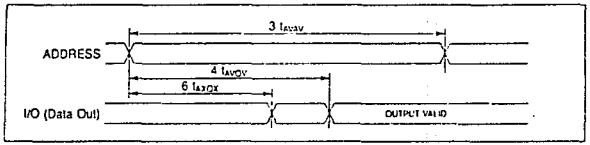


Figure 2.2 Read cycle 1[c,d]

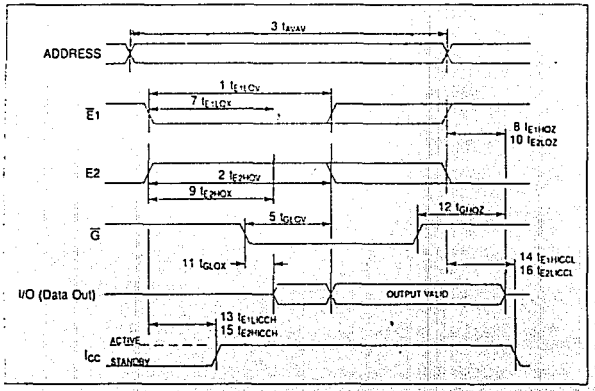


Figure 2.3 Read cycle 2[c]

Write cycle 1: $\overline{W}$ controlled[g,h]

No.	Symbol Stand.	Symbol Alt.	Parameter	16X5-15 min	16X5-15 max	16X5-20 min	16X5-20 max	16X5-25 min	16X5-25 max	Units	Notes*
18	t_{AVAV}	t_{WC}	Write Cycle Time	15		20		25		ns	
19	t_{WLWH}	t_{WP}	Write Pulse Width	12		15		20		ns	
20	t_{E1LWH}	t_{CW}	Chip Enable 1 to End of Write	12		15		20		ns	
21	t_{E2HWH}	t_{CW}	Chip Enable 2 to End of Write	12		15		20		ns	
22	t_{DVWH}	t_{DW}	Data Setup to End of Write	10		12		15		ns	
23	t_{WHDX}	t_{DH}	Data Hold after End of Write	0		0		0		ns	
24	t_{AVWH}	t_{AW}	Address Setup to End of Write	15		15		20		ns	
25	t_{AVWL}	t_{AS}	Address Setup to Start of Write	0		0		0		ns	
26	t_{WHAX}	t_{WR}	Add's Hold After End of Write	0		0		0		ns	
27	t_{WLQZ}	t_{WZ}	Write Enable to Output Disable	0	8	0	10	0	10	ns	f,j
28	t_{WHQX}	t_{OW}	Output Active after End of Write	0		0		0		ns	f,j

Write cycle 2: $\overline{E1}$ or E2 controlled[g,h]

No.	Symbol Stand.	Symbol Alt.	Parameter	16X5-15 min	16X5-15 max	16X5-20 min	16X5-20 max	16X5-25 min	16X5-25 max	Units	Notes*
29	t_{AVAV}	t_{WC}	Write Cycle Time	15		20		25		ns	
30	t_{WLWEH}	t_{WP}	Write Pulse Width	12		15		20		ns	
31	t_{E1LEH}	t_{CW}	Chip Enable 1 to End of Write	12		15		20		ns	
32	t_{E2HEL}	t_{CW}	Chip Enable 2 to End of Write	12		15		20		ns	
33	t_{DVEH}	t_{DW}	Data Setup to End of Write	10		12		15		ns	
34	t_{EHDX}	t_{DH}	Data Hold after End of Write	0		0		0		ns	
35	t_{AVEH}	t_{AW}	Address Setup to End of Write	15		15		20		ns	
36	t_{EHAX}	t_{WR}	Address Hold after End of Write	0		0		0		ns	
37	t_{AVEL}	t_{AS}	Address Setup to Start of Write	0		0		0		ns	
38	t_{WLQZ}	t_{WZ}	Write Enable to Output Disable	0	8	0	10	0	10	ns	f,j

* Refer to section 2.1.7.

2.1.7 Notes

Note a I_{CC} is dependent on output loading and cycle rate. the specified values are obtained with the output unloaded.
Note b This parameter is sampled and not 100% tested.
Note c For Read Cycle 1 and 2, $\overline{W}$ is high for entire cycle.
Note d Device is continually selected. $\overline{E1}$ low $\overline{G}$ low and E2 high.
Note e Measured between V_A max and V_{ss} min.
Note f Measured ±200mV from steady state output voltage. Load capacitance is 5pF.
Note g $\overline{E1}$, E2, $\overline{G}$ and $\overline{W}$ must transition between V_{ss} to V_A or V_A to V_{ss} in a monotonic fashion.
Note h $\overline{E1}$ or $\overline{W}$ must be $\geq V_{ss}$ or E2 must be $\leq V_A$ during address transitions.
Note i If $\overline{W}$ is low when the later of $\overline{E1}$ goes low or E2 goes high, the outputs remain in the high impedance state.
Note j Parameter guaranteed but not tested.
Note k Supply recovery rate should not exceed 100mV per 10µs from V_{DR} to V_{CC} min.

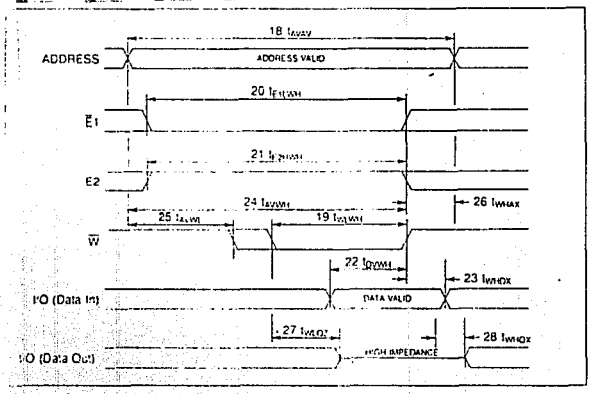


Figure 2.4 WRITE CYCLE 1

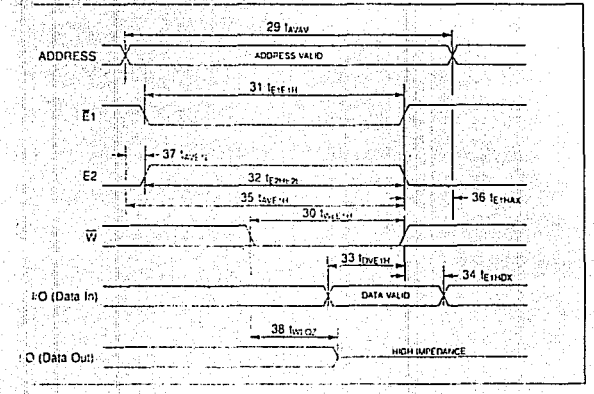


Figure 2.5 WRITE CYCLE 2

2.1.8 Power distribution

Recommended power distribution schemes combine proper power trace layout and placement of decoupling capacitors to maintain the wide operating margins of the IMS 16X5 series. The impedance in the decoupling path from the V_{CC} power pin through the decoupling capacitor to the ground pin should be kept to a minimum. The impedance of this path is determined by the series impedance of the power line inductance and the inductance/reactance of the decoupling capacitor.

Current transients associated with the operation of high speed memories have very high frequency components, so line inductance is the dominating factor. To reduce the line inductance, the power trace and ground trace should be gridded or provided by separate power planes. The decoupling capacitor supplies energy for high frequency current transients and should be located as near the memory as possible, with the shortest lead lengths practical. The high frequency decoupling capacitor should have a minimum value of 0.1µF and be placed between the rows of memory devices in the array. A larger tantalum capacitor for low frequency current transients should be placed near the memory board edge connection where the power traces meet the backplane power distribution system. These larger capacitors provide bulk energy storage to prevent voltage drop due to the main supply being located off the memory board and at the end of a long inductive path. The ground grid of the memory array should extend to the TTL driver periphery circuit. This will provide a solid ground reference for the drivers and prevent loss of operating margin due to differential ground noise.

2.1.9 Termination

Trace lines on a memory board in the array look to TTL driver signals like low impedance, unterminated transmission lines. In order to reduce or eliminate the reflections of the TTL signals propagating down the line, especially low going TTL signals, line termination is recommended. The termination may be either series or parallel.

The recommended series termination technique uses no DC current and a minimum number of components. This is accomplished by placing a series resistor in the signal line at the output of the TTL driver to dampen the reflection on the line. The termination resistor should be placed as close to the driver package as possible. The line should be kept short by placing the driver–termination combination close to the memory array.

Some experimentation will have to be done to find the proper value to use for the series termination to minimise reflections, but generally a series resistor in the 10 to 33Ω range will be required. Because the characteristic impedance of each layout will be different, it is necessary to select the proper value of this resistor by trial and error. A resistor of predetermined value may not properly terminate the transmission line.

Proper power distribution techniques, including adequate use of decoupling capacitors, and proper termination of TTL drive outputs are some of the most important yet basic guidelines that need to be followed when designing and building a memory board. The guidelines are intended to maintain the operating margins of all devices on the memory board by providing a quiet environment free of noise spikes, undershoot, and excessive ringing. It is wise to verify signal fidelity by observation using a wideband oscilloscope and probe

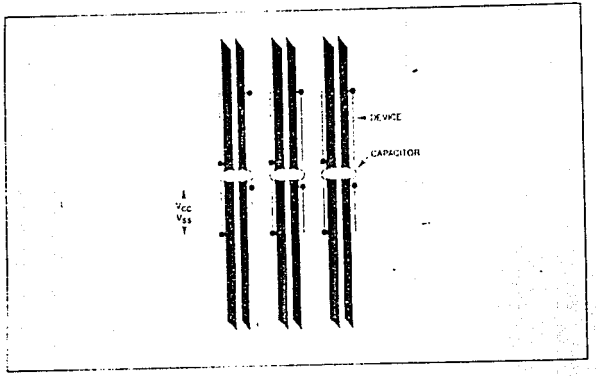


Figure 2.6 Grid showing decoupling capacitors

2.1.10 Data retention (low power versions only)(0°C ≤ T_A ≤ 70°C)

Symbol	Parameter	Min	Typ(25°C)	Max	Units	Notes*
V_{DR}	Data Retention Voltage	2.0			V	$V_{IN} \leq 0.2V$ or $\geq (V_{CC}-0.2V)$ $\overline{E} \geq (V_{CC}-0.2V)$
I_{CCDR1}	Data Retention Current		15	100	µA	V_{CC} = 3.0V
I_{CCDR2}	Data Retention Current		10	70	µA	V_{CC} = 2.0V
t_{EHVCCL}	Deselect Time (t_{CDR})	0			ns	
t_{VCCHEL}	Recovery Time (t_R)	t_{RC}			ns	t_{RC} = Read Cycle Time

* Refer to section 2.1.7

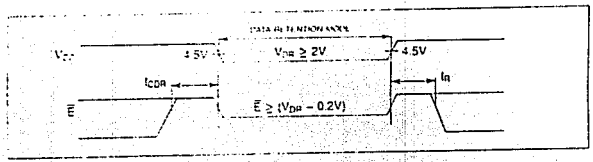


Figure 2.7 Data retention

2.2 Packaging information

2.2.1 Pin-outs and packages

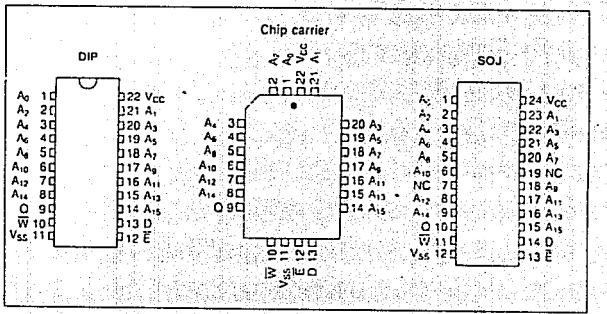


Figure 2.8 64K x 1 pin configuration

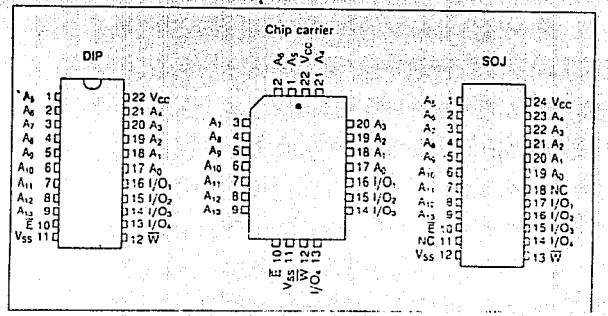


Figure 2.9 16K x 4 pin configuration

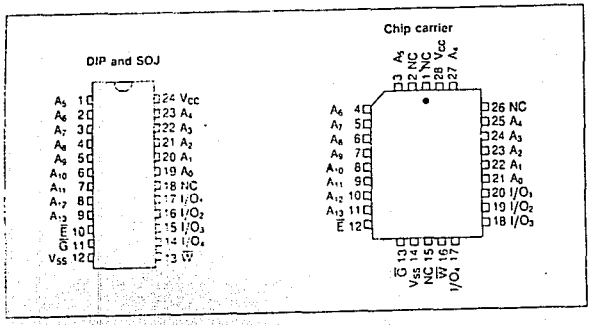


Figure 2.10 16K x 4 (with output enable) pin configuration

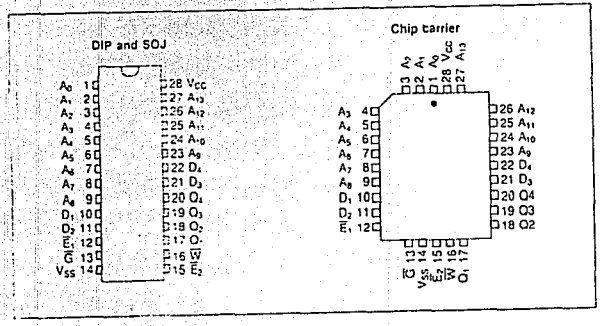


Figure 2.11 16K x 4 (with separate Inputs and Outputs) pin configuration

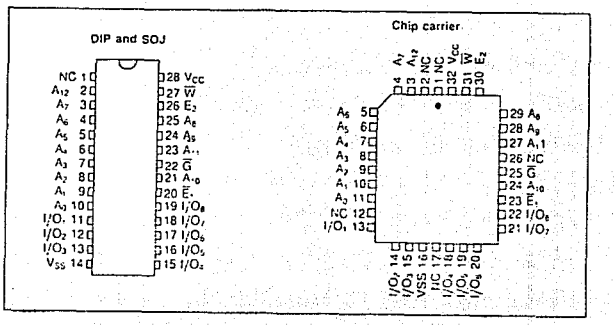


Figure 2.12 8K x 8 pin configuration

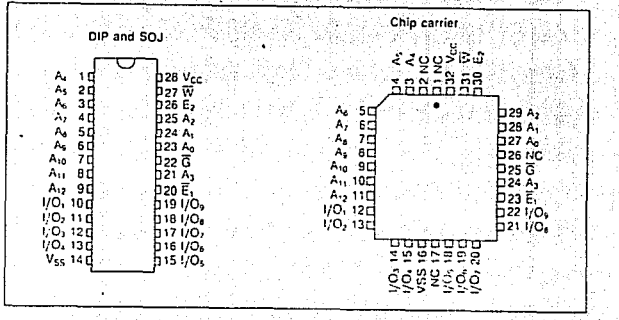


Figure 2.13 8K x 9 pin configuration

2.3 Ordering information

Device	Speed	Part Number
IMS1605	15ns	IMS1605z-15
	20ns	IMS1605z-20
	25ns	IMS1605z-25
IMS1625	15ns	IMS1625z-15
	20ns	IMS1625z-20
	25ns	IMS1625z-25
IMS1629	15ns	IMS1629z-15
	20ns	IMS1629z-20
	25ns	IMS1629z-25
IMS1626	15ns	IMS1626z-15
	20ns	IMS1626z-20
	25ns	IMS1626z-25
IMS1627	15ns	IMS1627z-15
	20ns	IMS1627z-20
	25ns	IMS1627z-25
IMS1635	15ns	IMS1635z-15
	20ns	IMS1635z-20
	25ns	IMS1635z-25
IMS1695	15ns	IMS1695z-15
	20ns	IMS1695z-20
	25ns	IMS1695z-25

Where z refers to packages P, S, E, or W. See also Appendix D.

inmos®

IMS18(
CMO
High Performanc
256K x 1 Static RA

FEATURES

- INMOS' Very High Speed CMOS
- Advanced Process - 1.2 Micron Design Rules
- 256K x 1 Bit Organization
- 25, 30, 35 and 45 ns Address Access Times
- 25, 30, 35 and 45 ns Chip Enable Access Times
- Fully TTL Compatible
- Separate Data Input and Outputs
- Three-state Output
- 24 Pin 300-mil DIP, SOJ and 28 Pin LCC
- Single +5V ± 10% Operation
- Power Down Function

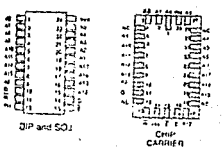
DESCRIPTION

The INMOS IMS1800 is a high performance 25... CMOS Static RAM. The IMS1800 provides max... density and speed enhancements with the addi... benefits of lower power and superior reliability.

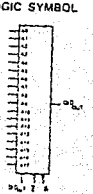
The IMS1800 features fully static operation req... no external clocks or timing strobes, with equal acc... and cycle times. Additionally, the IMS1800 provi... Chip Enable function (Ē) that can be used to plac... device into a low power standby mode.

The IMS1800E is an extended temperature ... pending military qualification of the IMS1800M.

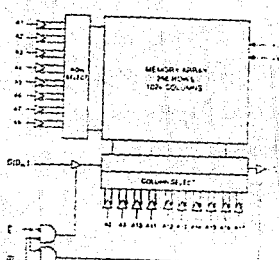
PIN CONFIGURATION



LOGIC SYMBOL



BLOCK DIAGRAM



PIN NAMES

A_0 - A_{17}	ADDRESS INPUTS	Q	DATA OUT
W	WRITE ENABLE	V_{CC}	POWER (+5V)
Ē	CHIP ENABLE	V_{SS}	GROUND
D	DATA INPUT		

A P E N D I C E

I I I

TARJETA PROTOTIPO PARA COMPUTADORAS PERSONALES TIPO IBM - AT

Personal Computer Hardware Reference Library

Prototype Adapter

6361674

Contents

Description

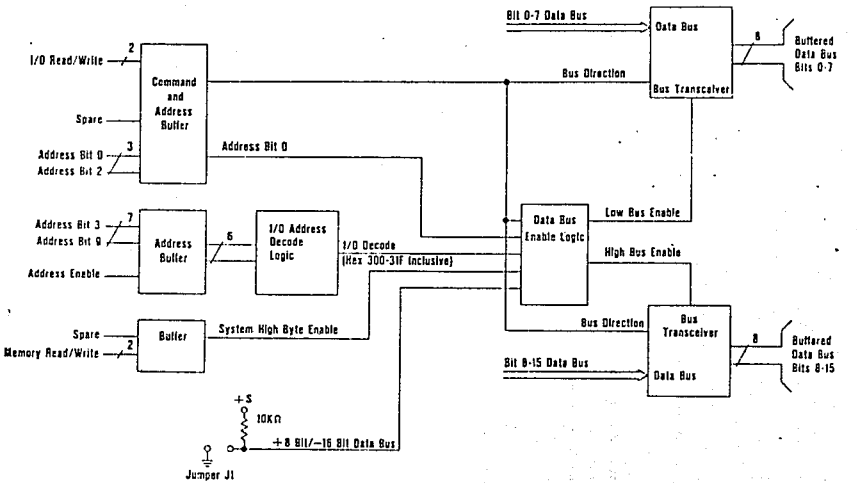
The IBM Personal Computer AT Prototype Adapter is 121.9 millimeters (4.8 inches) high by 333.25 millimeters (13.12 inches) long and plugs into any system-unit expansion slot except number 1 or 7. Two card-edge tabs, one 2-by 31- position and one 2-by 18-position, provide all system control signals and voltages.

The adapter has a voltage bus (+5 Vdc) and a ground bus (0 Vdc). Each bus borders the adapter, with the ground bus on the component side and the voltage bus on the pin side. A system interface is also provided on the adapter with a jumper to specify whether the device has an 8- or a 16-bit data bus.

This adapter also accommodates a D-shell connector from 9 to 37 positions.

> Note: All components must be installed on the component side of the adapter. The total width of the adapter, including components, may not exceed 12.7 millimeters (0.5 inch). If these specifications are not met, components on the IBM Personal Computer AT Prototype Adapter may touch other adapters plugged into adjacent expansion slots.

The following is a block diagram of the IBM Personal Computer AT Prototype Adapter.



Prototype Adapter Block Diagram

Adapter Design

The following information is provided to assist in designing an adapter using the IBM Personal Computer AT Prototype Adapter.

Designing an Input/Output Adapter

The following information may be used to design an input/output type of adapter.

Programming

Insert a Jump instruction after all I/O read (IOR) or I/O write (IOW) assembler language instructions to avoid a potential timing problem caused by slow I/O devices. The following figure shows a typical programming sequence.

Before	After
Your code	Your code
IOR	IOR
Your code	JMP NEXT
NEXT:	Your code

Program Sequence

Jumper Wire (J1)

Your design can use either 8 bits of the data bus (jumper off) or the full 16 bits of the data bus (jumper on). Most devices have 8-bit data buses.

Wait-State Generator Circuits

If your device runs too slow, you must add a wait-state generator to make the I/O read and write signals longer. First, determine the time needed by your device from the start of an IOR signal until it can put data on the system's data bus. Next, compare that

ime with the time given by the system's microprocessor. The system microprocessor gives 750 nanoseconds for 8-bit devices and 250 nanoseconds for 16-bit devices.

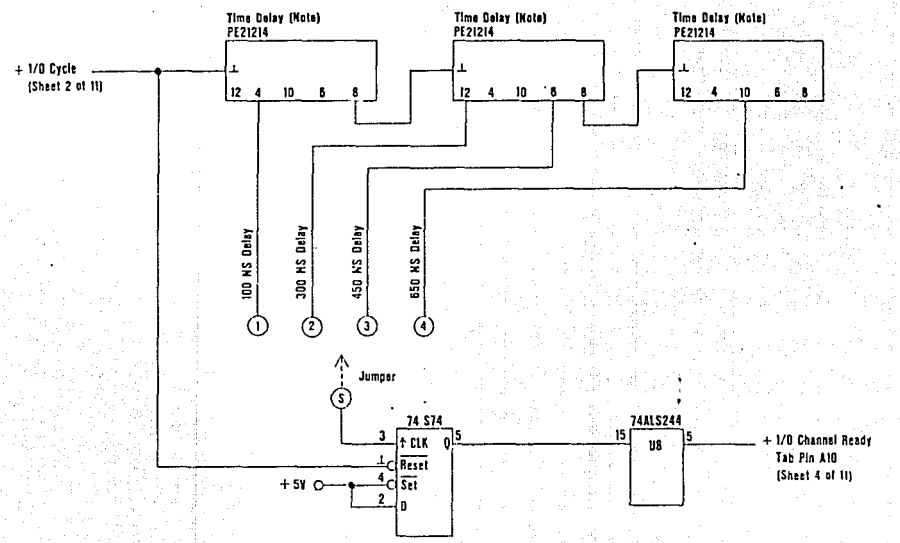
A similar problem may exist for an IOW signal. Determine the 'write data' setup time, which is the time required by your lesign from the time it is given valid data until it is told to take his data by the IOW signal. The time given by the system microprocessor from when data is first valid to the device until he IOW signal goes active and then inactive is shown in the ollowing figure. Your design can take the data when IOW goes active (less setup time) or when IOW goes inactive (more setup time).

8-Bit Device	16-Bit Device	
100 ns	100 ns	Data valid until IOW is active.
850 ns	350 ns	Data valid until IOW is inactive.

IOW Timing

If the time given by the system microprocessor is not enough, you must add a wait-state generator circuit that will provide longer IOR and IOW signals. A recommended wait-state generator circuit is shown in the following figure.

Note: Pulse Engineering Inc. PE21214 is the delay module used.



Wait-State Generator Circuit

Note: To add wait states and increase the time given by the microprocessor for I/O Read and Write commands, install one of the following jumpers.

16-Bit Design	
1 wait state	250 nanoseconds--No jumper
2 wait states	416 nanoseconds--Jumper 1 to 5
3 wait states	583 nanoseconds--Jumper 2 to 5
4 wait states	750 nanoseconds--Jumper 3 to 5
5 wait states	916 nanoseconds--Jumper 4 to 5
8-Bit Design	
4 wait states	750 nanoseconds--No Jumper
5 wait states	916 nanoseconds--Jumper 4 to 5

Designing a Memory Adapter

The following information may be used to design a memory adapter.

Control Lines

There are two sets of memory control lines. SMEMR for system-memory read, and SMEMW for system-memory write. They are active when accessing memory in the first megabyte (address bits 20 through 23 are all off). If you use these lines, you can avoid an address decode circuit that checks for address bits 20 through 23 being off.

The other set of control lines is MEMR and MEMW. These are active when addressing all memory locations. If you wish to design memory that will answer to addresses above the first

Prototype Adapter

megabyte, you must use these lines and decode address bits 20 through 23 to select the particular address range your memory occupies.

System Address Lines (SA)

The 20 lowest-order address lines are SA0 through SA19. SA address bits are active a minimum of 30 nanoseconds before a control line goes active, and they stay active until a minimum of 66 nanoseconds *after* the control line goes inactive. Timings are at the adapter socket.

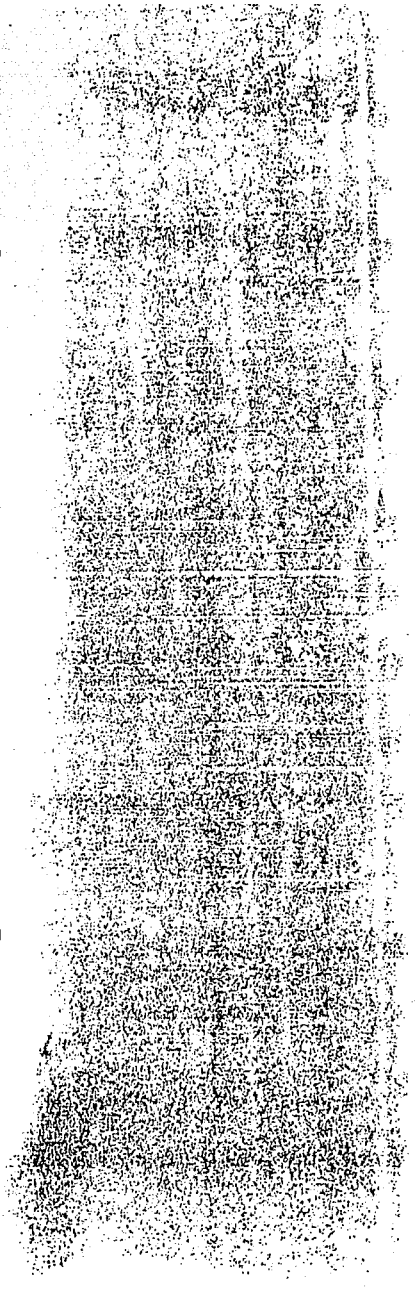
Local Address Lines (LA)

There are seven high-order address lines called LA17 through LA23. LA address bits are active a minimum of 159 nanoseconds before a control line goes active, and they stay active until typically 83 nanoseconds *before* the control line goes inactive. LA bits should be decoded to select the particular address range your memory occupies. Because this decode will go inactive 83 nanoseconds before the control line goes inactive, it may be necessary to latch the decode. The output of this decoder circuit should be connected to the input of a transparent latch, such as a 74ALS573 (+BALE should be connected to the clock pin on the latch). If this is done, the output of the 74LS573 will be active approximately 30 nanoseconds before a control line goes active, and will stay active until approximately 66 nanoseconds *after* the control line goes inactive. Timings are at the adapter socket.

IBM Personal Computer AT Prototype Adapter Layout

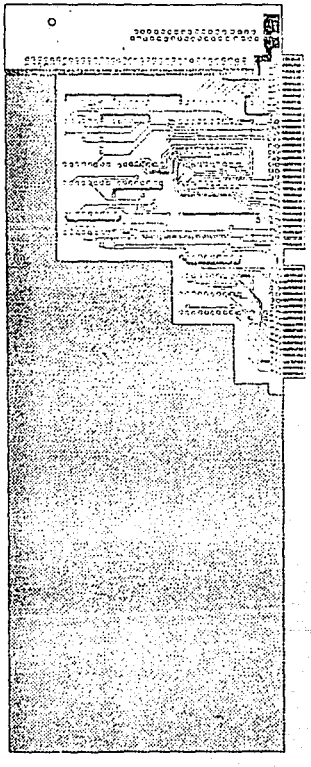
The IBM Personal Computer AT Prototype Adapter has two layers screened onto it: one on the front and one on the back. It also has 4,311 plated through-holes that are 10.1 millimeters (0.04 inch) wide and have a 1.52-millimeter (0.06-inch) pad. These holes are arranged in a 2.54-millimeter (0.1-inch) grid. There are 37 plated through-holes, 1.22 millimeters (0.048 inch) wide, on the rear of the adapter that are used for a 9- to 37-position D-shell connector. The adapter also has 5 holes that

re 3.18 millimeters (0.125 inch) wide. One of these is just above
ie two rows of D-shell connector holes, and each of the other
our is in a corner of the adapter.



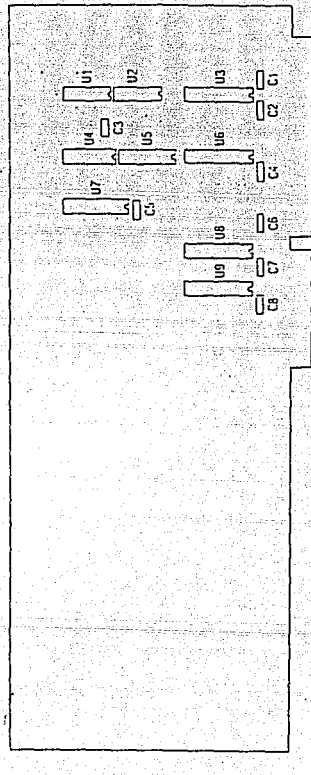
Component Side

The component side of the adapter has a ground bus, 1.27 millimeters (0.05 inch) wide screened onto it and two card-edge tabs labeled A1 through A31 and C1 through C31. .The following figure shows the ground bus and card edge-tabs.



Prototype Adapter 9

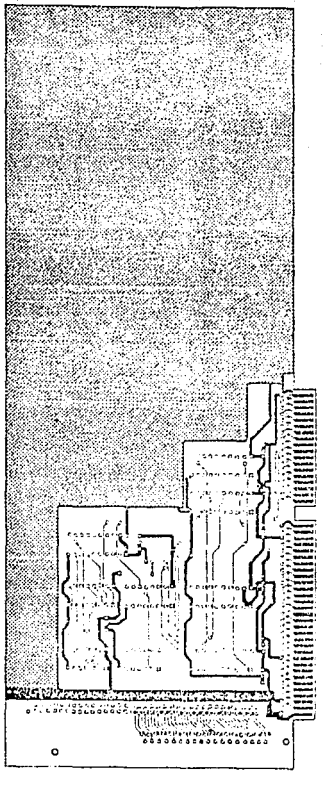
The component side of the adapter also has a silk screen printed on it that may be used as a component guide for the I/O interface. The following figure shows this silk screen.



10 Prototype Adapter

Pin Side

The pin side of the adapter has a 5-Vdc bus, 1.27 millimeters (0.05 inch) wide, screened onto it, and two card-edge tabs: labeled B1 through B31 and D1 through D18. The following figure shows the 5-Vdc bus and card edge-tabs.



Prototype Adapter 11

Card-Edge Tabs

Each card-edge tab is connected to a plated through-hole by a 0.3-millimeter (0.012-inch) land. Four ground tabs are connected to the ground bus by four 0.3-millimeter (0.012-inch) lands, and three 5 Vdc tabs are connected to the 5-Vdc bus by three 0.3-millimeter (0.012 inch) lands.

Additional Information

Additional information regarding the I/O interface may be found under "I/O Channel" in Section 1 of IBM Personal Computer AT Technical Reference manual. Logic diagrams of the IBM Personal Computer AT Prototype Adapter may be found later in this section. If the recommended interface logic is to be used, the following figure shows the recommended components and their TTL numbers.

Component	TTL #	Description
U1	74SC0	Quad 2 input NAND
U2	74S10	Triple 3 input NAND
U3, U9	74LS245	Octal bus transceiver
U4	74S139	Dual 1 of 4 decoder
U5	74S138	1 of 8 decoder
U6, U7, U8	74ALS244	Octal buffers
C1, C6		10-microfarad tantalum capacitor
C2, C3, C4, C5, C7, C8		0.047-microfarad ceramic capacitor
R1		10 Kohm, .25-watt, 10% resistor (axial leads)
J1		Jumper wire

Recommended Components

Note: J1, U8, and U9 are not required for a design using only the low-order 8 bits of the data bus. Designs using all 16 bits of the data bus require these components.

Interfaces

Internal Interface

Because of the number of adapters that may be installed in the system, I/O bus loading should be limited to 1 Schottky TTL load. If the recommended interface logic is used, this requirement is met. Power limitations may be found under "Power Supply" in the IBM Personal Computer AT Technical Reference Manual.

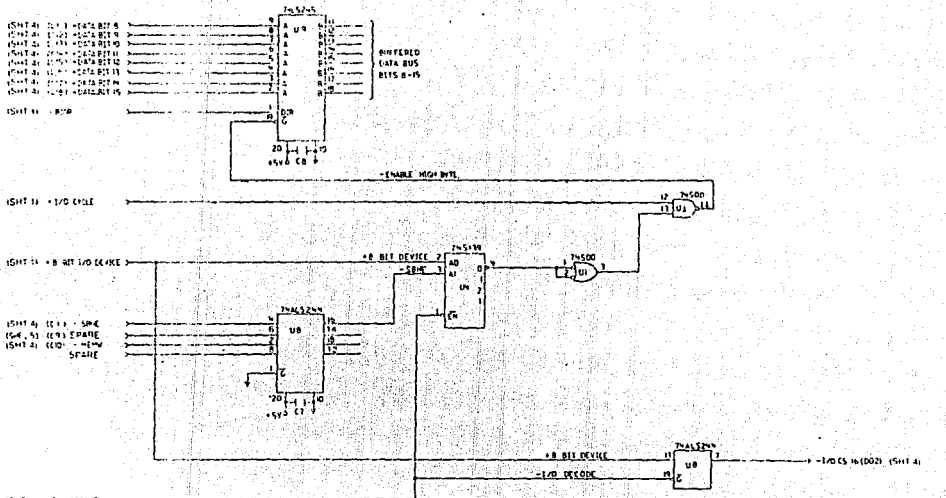
External Interface

The following figure lists the recommended connectors for the rear of the adapter.

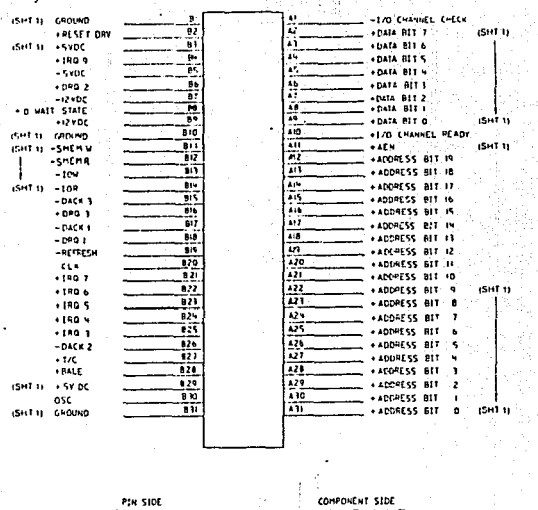
Connector	Part no. (Amp) or equivalent
9-pin D-shell (male)	205865-1
9-pin D-shell (female)	205866-1
15-pin D-shell (male)	205867-1
15-pin D-shell (female)	205868-1
25-pin D-shell (male)	205857-1
25-pin D-shell (female)	205858-1
37-pin D-shell (male)	205859-1
37-pin D-shell (female)	205860-1

Recommended Connectors

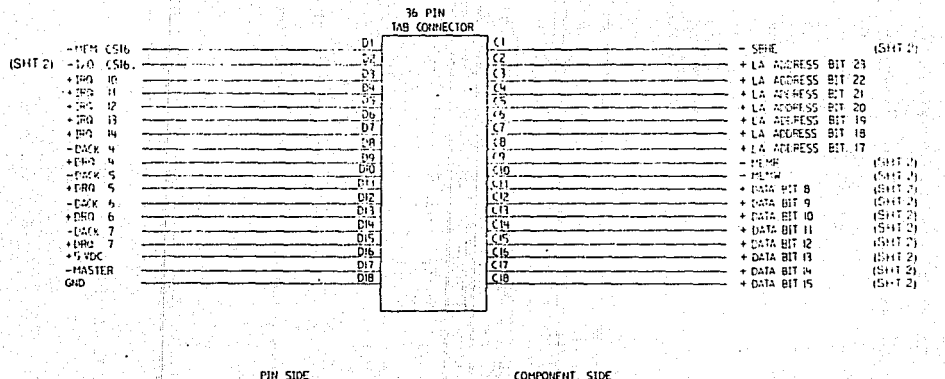
Prototype Adapter (Sheet 1 of 4)



Prototype Adapter (Sheet 2 of 4)



Prototype Adapter (Sheet 3 of 4)



Prototype Adapter (Sheet 4 of 4)

A P E N D I C E

I V

ACCESO DIRECTO DE MEMORIA

Y

SEÑALES DE LA TARJETA MADRE DE UNA COMPUTADOA PERSONAL TIPO IBM - AT

IBM

Personal Computer
Hardware Reference
Library

System Board

6361674

I/O Channel Signal Description

The following is a description of the system board's I/O channel signals. All signal lines are TTL-compatible. I/O adapters should be designed with a maximum of two low-power Shottky (LS) loads per line.

SA0 through SA19 (I/O)

Address bits 0 through 19 are used to address memory and I/O devices within the system. These 20 address lines, in addition to LA17 through LA23, allow access of up to 16Mb of memory. SA0 through SA19 are gated on the system bus when 'BALE' is high and are latched on the falling edge of 'BALE.' These signals are generated by the microprocessor or DMA Controller. They also may be driven by other microprocessors or DMA controllers that reside on the I/O channel.

LA17 through LA23 (I/O)

These signals (unlatched) are used to address memory and I/O devices within the system. They give the system up to 16Mb of addressability. These signals are valid when 'BALE' is high. LA17 through LA23 are not latched during microprocessor cycles and therefore do not stay valid for the whole cycle. Their purpose is to generate memory decodes for 1 wait-state memory cycles. These decodes should be latched by I/O adapters on the falling edge of 'BALE.' These signals also may be driven by other microprocessors or DMA controllers that reside on the I/O channel.

CLK (0)

This is the 6-MHz system clock. It is a synchronous microprocessor cycle clock with a cycle time of 167 nanoseconds. The clock has a 50% duty cycle. This signal should only be used for synchronization. It is not intended for uses requiring a fixed frequency.

Level		Function
MicroProcessor NMI		Parity or I/O Channel Check
Interrupt Controllers		
CTLR 1	CTLR 2	
IRQ 0		Timer Output 0
IRQ 1		Keyboard (Output Buffer Full)
IRQ 2		Interrupt from CTLR 2
	IRQ 8	Realtime Clock Interrupt
	IRQ 9	Software Redirected to INT ØAH (IRQ 2)
	IRQ 10	Reserved
	IRQ 11	Reserved
	IRQ 12	Reserved
	IRQ 13	Coprocessor
	IRQ 14	Fixed Disk Controller
	IRQ 15	Reserved
IRQ 3		Serial Port 2
IRQ 4		Serial Port 1
IRQ 5		Parallel Port 2
IRQ 6		Diskette Controller
IRQ 7		Parallel Port 1

ROM Subsystem

The system board's ROM subsystem consists of two 32K by 8-bit ROM/EPROM modules or four 16K by 8-bit ROM/EPROM modules in a 32K by 16-bit arrangement. The code for odd and even addresses resides in separate modules. ROM is assigned at the top of the first and last 1M address space (hex 0F0000 and hex FF0000). ROM is not parity-checked. Its access time is 150 nanoseconds and its cycle time is 230 nanoseconds.

RAM Subsystem

The system board's RAM subsystem starts at address hex 000000 of the 16M address space. It consists of either 256Kb or 512Kb of 128K by 1-bit RAM modules. Memory access time is 150 nanoseconds and the cycle time is 275 nanoseconds.

Memory-refresh requests one memory cycle every 15 microseconds through the timer/counter (channel 1). The RAM initialization program performs the following functions:

- Initializes channel 1 of the timer/counter to the rate generation mode, with a period of 15 microseconds.
- Performs a memory write operation to any memory location

  Note: The memory must be accessed or refreshed eight times before it can be used.

Direct Memory Access (DMA)

The system supports seven DMA channels. Two Intel 8237A-5 DMA Controller Chips are used, with four channels for each chip. The DMA channels are assigned as follows:

Ctlr 1	Ctlr 2
Ch 0 - Spare	Ch 4 - Cascade for Ctlr 1
Ch 1 - SDLC	Ch 5 - Spare
Ch 2 - Diskette (IBM Personal Computer)	Ch 6 - Spare
Ch 3 - Spare	Ch 7 - Spare

DMA Channels

DMA controller 1 contains channels 0 through 3. These channels support 8-bit data transfers between 8-bit I/O adapters and 8- or

16-bit system memory. Each channel can transfer data throughout the 16-megabyte system-address space in 64Kb blocks.

DMA controller 2 contains channels 4 through 7. Channel 4 is used to cascade channels 0 through 3 to the microprocessor. Channels 5, 6, and 7 support 16-bit data transfers between 16-bit I/O adapters and 16-bit system memory. These DMA channels can transfer data throughout the 16-megabyte system-address space in 128Kb blocks. Channels 5, 6, and 7 cannot transfer data on odd byte boundaries.

The following figure shows the addresses for the page register.

Page Register	I/O Hex Address
DMA Channel 0	0087
DMA Channel 1	0083
DMA Channel 2	0081
DMA Channel 3	0082
DMA Channel 5	008B
DMA Channel 6	0089
DMA Channel 7	008A
Refresh	008F

Page Register Addresses

The following figures show address generation for the DMA channels.

Source	DMA Page Registers	8237A-5
Address	A23<----------->A16	A15<----------->A0

Address Generation for DMA Channels 3 through 0

Note: The addressing signal, 'byte high enable' (BHE), is generated by inverting address line A0.

Source	DMA Page Registers	8237A-5
Address	A23<----------->A17	A16<----------->A1

Address Generation for DMA Channels 7 through 5

Note: The addressing signals, 'BHE' and 'A0', are forced to a logic 0.

Addresses for all DMA channels do not increase or decrease through page boundaries (64Kb for channels 0 through 3 and 128Kb for channels 5 through 7).

Programming the 16-Bit DMA Channels

DMA channels 5 through 7 perform 16-bit data transfers. Access can be gained only to 16 bit devices (I/O or memory) during the DMA cycles of channels 5 through 7. Access to the DMA controller (8237A-5), which controls these channels, is through I/O addresses 0C0 through 0DF. The command codes for the DMA controller are as follows:

Hex Address	Command Codes
0C0	CH0 base and current address
0C2	CH0 base and current word count
0C4	CH1 base and current address
0C6	CH1 base and current word count
0C8	CH2 base and current address
0CA	CH2 base and current word count
0CC	CH3 base and current address
0CE	CH3 base and current word count
0D0	Read Status Register/Write Command Register
0D2	Write Request Register
0D4	Write Single Mask Register Bit
0D6	Write Mode Register
0D8	Clear Byte Pointer Flip-Flop
0DA	Read Temporary Register/Write Master Clear
0DC	Clear Mask Register
0DE	Write All Mask Register Bits

DMA Controller Registers

All DMA memory transfers made with channels 5 through 7 must occur on even-byte boundaries. When the base address for these channels is programmed, the real address divided by 2 is the data that is written to the base address register. Also, when the base word count for channels 5 through 7 is programmed, the count is the number of 16-bit words to be transferred. Therefore, DMA channels 5 through 7 can transfer 65,536 words or 128Kb maximum for any selected page of memory. These DMA channels divide the 16Mb memory space into 128Kb pages. When the DMA page registers for channels 5 through 7 are

programmed, data bits D7 through D1 should contain the high-order seven address bits (A23 through A17) of the desired memory space. Data bit D0 of the page registers for channels 5 through 7 is not used in the generation of the DMA memory address.

After power-up time, all internal locations, especially the mode registers, should be loaded with some valid value. This should be done even if some channels are unused.

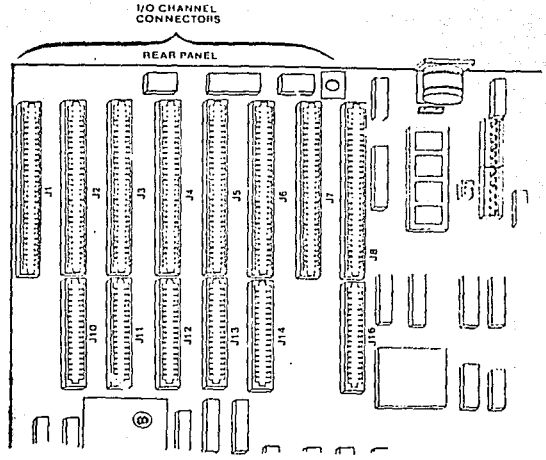
I/O Channel

The I/O channel supports:

- I/O address space hex 100 to hex 3FF
- 24-bit memory addresses (16Mb)
- Selection of data accesses (either 8- or 16-bit)
- Interrupts
- DMA channels
- I/O wait-state generation
- Open-bus structure (allowing multiple microprocessors to share the system's resources, including memory)
- Refresh of system memory from channel microprocessors.

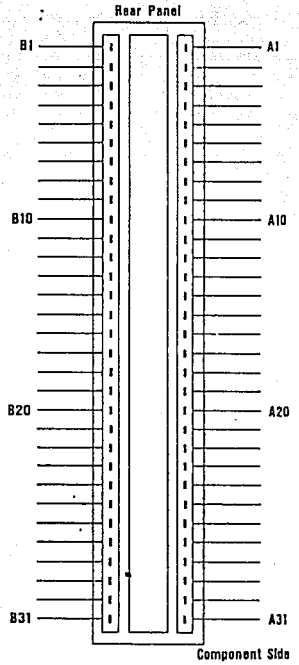
The following figure shows the location and the numbering of the I/O channel connectors. These connectors consist of eight 62-pin and six 36-pin edge connector sockets.

Note: In two positions on the I/O channel, the 36-pin connector is not present. These positions can support only 62-pin I/O bus adapters.



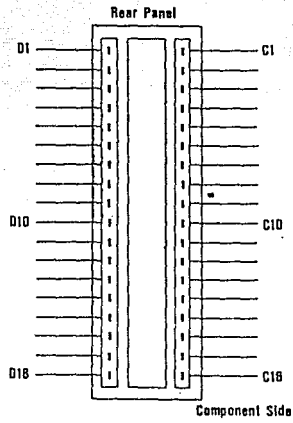
1-16 System Board

The following figure shows the pin numbering for I/O channel connectors J1 through J8.



I/O Channel Pin Numbering
(J1-J8)

The following figure shows the pin numbering for I/O channel connectors J12 through J16 and J18.



I/O Channel Pin Numbering (J10-J14 and J16)

The following figures summarize pin assignments for the I/O channel connectors.

I/O Pin	Signal Name	I/O
A 1	-I/O CH CK	I
A 2	SD7	I/O
A 3	SD6	I/O
A 4	SD5	I/O
A 5	SD4	I/O
A 6	SD3	I/O
A 7	SD2	I/O
A 8	SD1	I/O
A 9	SD0	I/O
A 10	-I/O CH RDY	I
A 11	AEN	O
A 12	SA19	I/O
A 13	SA18	I/O
A 14	SA17	I/O
A 15	SA16	I/O
A 16	SA15	I/O
A 17	SA14	I/O
A 18	SA13	I/O
A 19	SA12	I/O
A 20	SA11	I/O
A 21	SA10	I/O
A 22	SA9	I/O
A 23	SA8	I/O
A 24	SA7	I/O
A 25	SA6	I/O
A 26	SA5	I/O
A 27	SA4	I/O
A 28	SA3	I/O
A 29	SA2	I/O
A 30	SA1	I/O
A 31	SA0	I/O

I/O Channel (A-Side, J1 through J8)

I/O Pin	Signal Name	I/O
B 1	GND	Ground
B 2	RESET DRV	O
B 3	+5 Vdc	Power
B 4	IRQ 9	I
B 5	-5 Vdc	Power
B 6	DRQ2	I
B 7	-12 Vdc	Power
B 8	OWS	I
B 9	+12 Vdc	Power
B 10	GND	Ground
B 11	-SMEMW	O
B 12	-SMEMR	O
B 13	-IOW	I/O
B 14	-IOR	I/O
B 15	-DACK3	O
B 16	DRQ3	I
B 17	-DACK1	O
B 18	DRQ1	I
B 19	-Refresh	I/O
B 20	CLK	O
B 21	IRQ7	I
B 22	IRQ6	I
B 23	IRQ5	I
B 24	IRQ4	I
B 25	IRQ3	I
B 26	-DACK2	O
B 27	T/C	O
B 28	BALE	O
B 29	+5 Vdc	Power
B 30	OSC	O
B 31	GND	Ground

I/O Channel (B-Side J1, through J8)

I/O Pin	Signal Name	I/O
C 1	SBHE	I/O
C 2	LA23	I/O
C 3	LA22	I/O
C 4	LA21	I/O
C 5	LA20	I/O
C 6	LA19	I/O
C 7	LA18	I/O
C 8	LA17	I/O
C 9	-MEMR	I/O
C 10	-MEMW	I/O
C 11	SD08	I/O
C 12	SD09	I/O
C 13	SD10	I/O
C 14	SD11	I/O
C 15	SD12	I/O
C 16	SD13	I/O
C 17	SD14	I/O
C 18	SD15	I/O

I/O Channel (C-Side J10 through J14 and J16)

I/O Pin	Signal Name	I/O
D 1	-MEM CS16	I
D 2	-I/O CS16	I
D 3	IRQ10	I
D 4	IRQ11	I
D 5	IRQ12	I
D 6	IRQ15	I
D 7	IRQ14	I
D 8	-DACK0	O
D 9	DRQ0	I
D 10	-DACK5	O
D 11	DRQ5	I
D 12	-DACK6	O
D 13	DRQ6	I
D 14	-DACK7	O
D 15	DRQ7	I
D 16	+5 Vdc	Power
D 17	-MASTER	I
D 18	GND	Ground

I/O Channel (D-Side, J10 through J14 and J16)

RESET DRV (0)

'Reset drive' is used to reset or initialize system logic at power-up time or during a low line-voltage outage. This signal is active high.

SD0 through SD15 (I/O)

These signals provide bus bits 0 through 15 for the microprocessor, memory, and I/O devices. D0 is the least-significant bit and D15 is the most-significant bit. All 8-bit devices on the I/O channel should use D0 through D7 for communications to the microprocessor. The 16-bit devices will use D0 through D15. To support 8-bit devices, the data on D8 through D15 will be gated to D0 through D7 during 8-bit transfers to these devices; 16-bit microprocessor transfers to 8-bit devices will be converted to two 8-bit transfers.

BALE (0) (buffered)

'Address latch enable' is provided by the 82288 Bus Controller and is used on the system board to latch valid addresses and memory decodes from the microprocessor. It is available to the I/O channel as an indicator of a valid microprocessor or DMA address (when used with 'AEN'). Microprocessor addresses SA0 through SA19 are latched with the falling edge of 'BALE.' 'BALE' is forced high during DMA cycles.

-I/O CH CK (I)

'-I/O channel check' provides the system board with parity (error) information about memory or devices on the I/O channel. When this signal is active, it indicates an uncorrectable system error.

I/O CH RDY (I)

'I/O channel ready' is pulled low (not ready) by a memory or I/O device to lengthen I/O or memory cycles. Any slow device using this line should drive it low immediately upon detecting its valid address and a Read or Write command. Machine cycles are extended by an integral number of clock cycles (167 nanoseconds). This signal should be held low for no more than 2.5 microseconds.

IRQ3-IRQ7, IRQ9-IRQ12 and IRQ 14 through 15 (I)

Interrupt Requests 3 through 7, 9 through 12, and 14 through 15 are used to signal the microprocessor that an I/O device needs attention. The interrupt requests are prioritized, with IRQ9 through IRQ12 and IRQ14 through IRQ15 having the highest priority (IRQ9 is the highest) and IRQ3 through IRQ7 having the lowest priority (IRQ7 is the lowest). An interrupt request is generated when an IRQ line is raised from low to high. The line must be held high until the microprocessor acknowledges the interrupt request (Interrupt Service routine). Interrupt 13 is used on the system board and is not available on the I/O channel. Interrupt 8 is used for the real-time clock.

-IOR (I/O)

'-I/O Read' instructs an I/O device to drive its data onto the data bus. It may be driven by the system microprocessor or DMA controller, or by a microprocessor or DMA controller resident on the I/O channel. This signal is active low.

-IOW (I/O)

'-I/O Write' instructs an I/O device to read the data on the data bus. It may be driven by any microprocessor or DMA controller in the system. This signal is active low.

-SMEMR (O) -MEMR (I/O)

These signals instruct the memory devices to drive data onto the data bus. '-SMEMR' is active only when the memory decode is within the low 1Mb of memory space. '-MEMR' is active on all memory read cycles. '-MEMR' may be driven by any microprocessor or DMA controller in the system. '-SMEMR' is derived from '-MEMR' and the decode of the low 1Mb of memory. When a microprocessor on the I/O channel wishes to drive '-MEMR', it must have the address lines valid on the bus for one system clock period before driving '-MEMR' active. Both signals are active LOW.

-SMEMW (O) -MEMW (I/O)

These signals instruct the memory devices to store the data present on the data bus. '-SMEMW' is active only when the memory decode is within the low 1Mb of the memory space. '-MEMW' is active on all memory read cycles. '-MEMW' may be driven by any microprocessor or DMA controller in the system. '-SMEMW' is derived from '-MEMW' and the decode of the low 1Mb of memory. When a microprocessor on the I/O channel wishes to drive '-MEMW', it must have the address lines valid on the bus for one system clock period before driving '-MEMW' active. Both signals are active low.

DRQ0-DRQ3 and DRQ5-DRQ7 (I)

DMA Requests 0 through 3 and 5 through 7 are asynchronous channel requests used by peripheral devices and the I/O channel microprocessors to gain DMA service (or control of the system). They are prioritized, with 'DRQ0' having the highest priority and 'DRQ7' having the lowest. A request is generated by bringing a DRQ line to an active level. A DRQ line must be held high until the corresponding 'DMA Request Acknowledge' (DACK) line goes active. 'DRQ0' through 'DRQ3' will perform 8-bit DMA transfers; 'DRQ5' through 'DRQ7' will perform 16-bit transfers. 'DRQ4' is used on the system board and is not available on the I/O channel.

-DACK0 to -DACK3 and -DACK5 to -DACK7 (O)

-DMA Acknowledge 0 to 3 and 5 to 7 are used to acknowledge DMA requests (DRQ0 through DRQ7). They are active low.

AEN (O)

'Address Enable' is used to degate the microprocessor and other devices from the I/O channel to allow DMA transfers to take place. When this line is active, the DMA controller has control of the address bus, the data-bus Read command lines (memory and I/O), and the Write command lines (memory and I/O).

-REFRESH (I/O)

This signal is used to indicate a refresh cycle and can be driven by a microprocessor on the I/O channel.

T/C (O)

'Terminal Count' provides a pulse when the terminal count for any DMA channel is reached.

SBHE (I/O)

'Bus High Enable' (system) indicates a transfer of data on the upper byte of the data bus, SD8 through SD15. Sixteen-bit devices use 'SBHE' to condition data bus buffers tied to SD8 through SD15.

-MASTER (I)

This signal is used with a DRQ line to gain control of the system. A processor or DMA controller on the I/O channel may issue a DRQ to a DMA channel in cascade mode and receive a '-DACK'. Upon receiving the '-DACK', an I/O microprocessor may pull '-MASTER' low, which will allow it to

control the system address, data, and control lines (a condition known as *tri-state*). After '-MASTER' is low, the I/O microprocessor must wait one system clock period before driving the address and data lines, and two clock periods before issuing a Read or Write command. If this signal is held low for more than 15 microseconds, system memory may be lost because of a lack of refresh.

-MEM CS16 (I)

'-MEM 16 Chip Select' signals the system board if the present data transfer is a 1 wait-state, 16-bit, memory cycle. It must be derived from the decode of LA17 through LA23. '-MEM CS16' should be driven with an open collector or tri-state driver capable of sinking 20 mA.

-I/O CS16 (I)

'-I/O 16 bit Chip Select' signals the system board that the present data transfer is a 16-bit, 1 wait-state, I/O cycle. It is derived from an address decode. '-I/O CS16' is active low and should be driven with an open collector or tri-state driver capable of sinking 20 mA.

OSC (O)

'Oscillator' (OSC) is a high-speed clock with a 70-nanosecond period (14.31818 MHz). This signal is not synchronous with the system clock. It has a 50% duty cycle.

0WS (I)

The 'Zero Wait State' (0WS) signal tells the microprocessor that it can complete the present bus cycle without inserting any additional wait cycles. In order to run a memory cycle to a 16-bit device without wait cycles, '0WS' is derived from an address decode gated with a Read or Write command. In order to run a memory cycle to an 8-bit device with a minimum of two wait states, '0WS' should be driven active one system clock after the

Read or Write command is active gated with the address decode for the device. Memory Read and Write commands to an 8-bit device are active on the falling edge of the system clock. '0WS' is active low and should be driven with an open collector or tri-state driver capable of sinking 20 mA.

The following figure is an I/O address map.

Hex Range	Device
000-01F	DMA controller 1, 8237A-5
020-03F	Interrupt controller 1, 8259A, Master
040-05F	Timer, 8254.2
060-06F	8042 (Keyboard)
070-07F	Real-time clock, NMI (non-maskable interrupt) mask
080-09F	DMA page register, 74LS612
0A0-0BF	Interrupt controller 2, 8259A
0C0-0DF	DMA controller 2, 8237A-5
0F0	Clear Math Coprocessor Busy
0F1	Reset Math Coprocessor
0F8-0FF	Math Coprocessor
1F0-1F8	Fixed Disk
200-207	Game I/O
278-27F	Parallel printer port 2
2F8-2FF	Serial port 2
300-31F	Prototype card
360-36F	Reserved
378-37F	Parallel printer port 1
380-38F	SDLC, bisynchronous 2
3A0-3AF	Bisynchronous 1
3B0-3BF	Monochrome Display and Printer Adapter
3C0-3CF	Reserved
3D0-3DF	Color/Graphics Monitor Adapter
3F0-3F7	Diskette controller
3F8-3FF	Serial port 1

I/O Address Map

Note: I/O addresses, hex 000 to 0FF, are reserved for the system board I/O. Hex 100 to 3FF are available on the I/O channel.

At power on time, the non-maskable interrupt (NMI) into the 80286 is masked off. The mask bit can be set and reset with system programs as follows:

Mask On — Write to I/O address hex 070, with data bit 7 equal to a logic 0

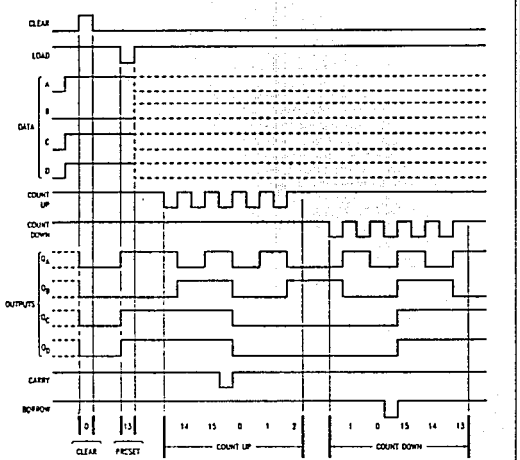
A P E N D I C E

V

CONTADOR 74LS193.

Timing Diagrams

Typical Clear, Load, and Count Sequences



Note A: Clear overrides load, data, and count inputs.

Note B: When counting up, count-down input must be high; when counting down, count-up input must be high.

National Semiconductor

54LS194A/DM74LS194A 4-Bit Bidirectional Universal Shift Register

General Description

This bidirectional shift register is designed to incorporate virtually all of the features a system designer may want in a shift register; they feature parallel inputs, parallel outputs, right-shift and left-shift serial inputs, operating-mode-control inputs, and a direct overriding clear line. The register has four distinct modes of operation, namely

Parallel (broadside) load
Shift right (in the direction Q_A toward Q_D)
Shift left (in the direction Q_D toward Q_A)
Inhibit clock (do nothing)

Synchronous parallel loading is accomplished by applying the four bits of data and taking both mode control inputs, S0 and S1, high. The data is loaded into the associated flip-flops and appear at the outputs after the positive transition of the clock input. During loading, serial data flow is inhibited.

Shift right is accomplished synchronously with the rising edge of the clock pulse when S0 is high and S1 is low. Serial data for this mode is entered at the shift-right input. When S0 is low and S1 is high, data shifts synchronously and new data is entered at the shift-left input.

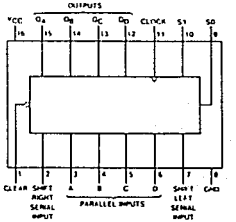
Clocking of the flip-flop is inhibited when both mode control inputs are low.

Features

- Parallel inputs and outputs
- Four operating modes:
  - Synchronous parallel load
  - Right shift
  - Left shift
  - Do nothing
- Positive edge-triggered clocking
- Direct overriding clear

Connection Diagram

Dual-In-Line Package



Order Number 54LS194ADMQB, 54LS194AFMQB, 54LS194ALMQB, DM74LS194AM or DM74LS194AN
See NS Package Number E20A, J16A, M16A, N16E or W16A

National Semiconductor

54LS193/DM54LS193/DM74LS193 Synchronous 4-Bit Up/Down Binary Counters with Dual Clock

General Description

This circuit is a synchronous up/down 4-bit binary counter. Synchronous operation is provided by having all flip-flops clocked simultaneously, so that the outputs change together when so instructed by the steering logic. This mode of operation eliminates the output counting spikes normally associated with asynchronous (ripple clock) counters.

The outputs of the four master-slave flip-flops are triggered by a low-to-high level transition of either count (clock) input. The direction of counting is determined by which count input is pulsed while the other count input is high.

The counter is fully programmable; that is, each output may be preset to either level by entering the desired data at the inputs while the load input is low. The output will change independently of the count pulses. This feature allows the counters to be used as modulo-N dividers by simply modifying the count length with the preset inputs.

A clear input has been provided which, when taken to a high level, forces all outputs to the low level, independent of the count and load inputs. The clear, count, and load inputs are buffered to lower the drive requirements of clock drivers, etc., required for long words.

These counters were designed to be cascaded without the need for external circuitry. Both borrow and carry outputs are available to cascade both the up and down counting functions. The borrow output produces a pulse equal in width to the count-down input when the counter underflows.

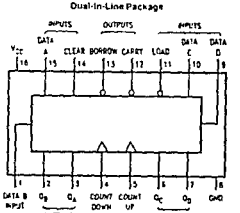
Similarly, the carry output produces a pulse equal in width to the count down input when an overflow condition exists. The counters can then be easily cascaded by feeding the borrow and carry outputs to the count down and count up inputs respectively of the succeeding counter.

Features

- Fully independent clear input
- Synchronous operation
- Cascading circuitry provided internally
- Individual preset each flip-flop
- Alternate Military/Aerospace device (54LS193) is available. Contact a National Semiconductor Sales Office/Distributor for specifications.

Connection Diagram

Dual-In-Line Package



TL/F/5404-1

Order Number 54LS193DMQB, 54LS193FMQB, 54LS193LMQB, DM54LS193J, DM54LS193W, DM74LS193M or DM74LS193N
See NS Package Number E20A, J16A, M16A, N16E or W16A

Absolute Maximum Ratings (Note)

If Military/Aerospace specified devices are required, please contact the National Semiconductor Sales Office/Distributors for availability and specifications.

Supply Voltage	7V
Input Voltage	7V
Operating Free Air Temperature Range	
DM54LS and 54LS	−55°C to +125°C
DM74LS	0°C to +70°C
Storage Temperature Range	−65°C to +150°C

Note: *The "Absolute Maximum Ratings" are those values beyond which the safety of the device cannot be guaranteed. The device should not be operated at these limits. The parametric values defined in the "Electrical Characteristics" table are not guaranteed at the absolute maximum ratings. The "Recommended Operating Conditions" table will define the conditions for actual device operation.*

Recommended Operating Conditions

Symbol	Parameter	DM54LS193			DM74LS193			Units
		Min	Nom	Max	Min	Nom	Max	
V_{CC}	Supply Voltage	4.5	5	5.5	4.75	5	5.25	V
V_{IH}	High Level Input Voltage	2			2			V
V_{IL}	Low Level Input Voltage			0.7			0.8	V
I_{OH}	High Level Output Current			−0.4			−0.4	mA
I_{OL}	Low Level Output Current			4			8	mA
f_{CLK}	Clock Frequency (Note 1)	0		25	0		25	MHz
	Clock Frequency (Note 2)	0		20	0		20	MHz
t_W	Pulse Width of Any Input (Note 6)	20			20			ns
t_{SU}	Data Setup Time (Note 6)	20			20			ns
t_H	Data Hold Time (Note 6)	0			0			ns
t_{REL}	Release Time (Note 6)	40			40			ns
T_A	Free Air Operating Temperature	−55		125	0		70	°C

Electrical Characteristics over recommended operating free air temperature range (unless otherwise noted)

Symbol	Parameter	Conditions		Min	Typ (Note 3)	Max	Units
V_I	Input Clamp Voltage	V_{CC} = Min, I_I = −18 mA				−1.5	V
V_{OH}	High Level Output Voltage	V_{CC} = Min, I_{OH} = Max, V_{IL} = Max, V_{IH} = Min	DM54	2.5	3.4		V
			DM74	2.7	3.4		
V_{OL}	Low Level Output Voltage	V_{CC} = Min, I_{OL} = Max, V_{IL} = Max, V_{IH} = Min	DM54		0.25	0.4	V
			DM74		0.35	0.5	
		I_{OL} = 4 mA, V_{CC} = Min	DM74		0.25	0.4	
I_I	Input Current @ Max Input Voltage	V_{CC} = Max, V_I = 7V				0.1	mA
I_{IH}	High Level Input Current	V_{CC} = Max, V_I = 2.7V				20	μA
I_{IL}	Low Level Input Current	V_{CC} = Max, V_I = 0.4V				−0.4	mA
I_{OS}	Short Circuit Output Current	V_{CC} = Max (Note 4)	DM54	−20		−100	mA
			DM74	−20		−100	
I_{CC}	Supply Current	V_{CC} = Max (Note 5)			19	34	mA

Note 1: C_L = 15 pF, R_L = 2 kΩ, I_A = 25°C and V_{CC} = 5V

Note 2: C_L = 50 pF, R_L = 2 kΩ, I_A = 25°C and V_{CC} = 5V

Note 3: All typicals are at V_{CC} = 5V, T_A = 25°C

Note 4: Not more than one output should be shorted at a time, and the duration should not exceed one second.

Note 5: I_{CC} is measured with all outputs open, CLEAR and LOAD inputs grounded, and all other inputs at 4.5V.

Note 6: T_A = 25°C and V_{CC} = 5V

Switching Characteristics at V_{CC} = 5V and T_A = 25°C (See Section 1 for Test Waveforms and Output Load)

Symbol	Parameter	From (Input) To (Output)	R_L = 2 kΩ				Units
			C_L = 15 pF		C_L = 50 pF		
			Min	Max	Min	Max	
f_{MAX}	Maximum Clock Frequency		25		20		MHz
t_{PLH}	Propagation Delay Time Low to High Level Output	Count Up to Carry		26		30	ns
t_{PHL}	Propagation Delay Time High to Low Level Output	Count Up to Carry		24		36	ns
t_{PLH}	Propagation Delay Time Low to High Level Output	Count Down to Borrow		24		29	ns
t_{PHL}	Propagation Delay Time High to Low Level Output	Count Down to Borrow		24		32	ns
t_{PLH}	Propagation Delay Time Low to High Level Output	Either Count to Any Q		38		45	ns
t_{PHL}	Propagation Delay Time High to Low Level Output	Either Count to Any Q		47		53	ns
t_{PLH}	Propagation Delay Time Low to High Level Output	Load to Any Q		40		41	ns
t_{PHL}	Propagation Delay Time High to Low Level Output	Load to Any Q		40		47	ns
t_{PHL}	Propagation Delay Time High to Low Level Output	Clear to Any Q		35		44	ns

Logic Diagram