DSP. Procesador de Señales Digitales

Practica libre de I.A.C.A.
Alumno: Juan Andrés Morales Cordovilla.

QUE HACE EL DSP

Permite escribir una tabla y(k) de elementos, a partir de otra dada x(k). Cada elemento de la tabla y(k) sale de hacer operaciones del estilo: **y(k) = 3*x(k) + 1*x(k-5) + 2*y(k-2)**. Tanto la tabla x(k) como la y(k) se dan por sus posiciones de inicio dx y dy previamente escritos en la RAM y sus elementos serán consecutivos.

Su principal ventaja reside en que va obteniendo cada elemento y(k) (que consume multiplicaciones, trasferencias entre registros…) a la vez que va leyendo y escribiendo los datos en la RAM (que se supone que es mas lenta que la unidad de procesamiento).

COMO USARLO:

1) Los limites de uso:

- Elementos x(k) e y(k): no mas de 8 bits y solo positivos (0, 255).
- Coeficientes: no mas de 8 bits y solo positivos (0, 255).
- Direcciones dx y dy de inicio: no mas de 8 bits (0, 255).
- Tamaño de la RAM: 2^8 palabras de 16 bits cada una.
- Nº de máximo de sumandos: 8
- Elementos mas recientes y antiguos que puedo usar: x(k) y x(k-7), y(k-1) e y(k-8).

2) Como se indica la operación que debe hacer el DSP:

Esto se indica escribiendo 'la zona de preparación' en las primeras posiciones de la RAM:

*¡¡M UY IMPORTANTE!!:

1. A partir de la posición 0 de la RAM empieza la zona de preparación.
2. La zona finaliza cuando la palabra escrita es 0000)h..
3. La ultima palabra de la zona de preparación debe ser la dirección dy donde se escribirá el primer dato de la tabla y(k) que será siempre el y(-2). Luego si me dicen que la tabla y(k) empiece en la posición dy (y(0) en dy), he de cargar contador DIRECCION_Y con dy-2
4. Si la tabla tiene ne elementos, debo de suponer que tiene 2 más y cargar el CONT_DESC_Nº_SUMANDOS con ne+2.

INSTRUCCIONES PARA CARGAR LOS REGISTROS Y CONTADORES:

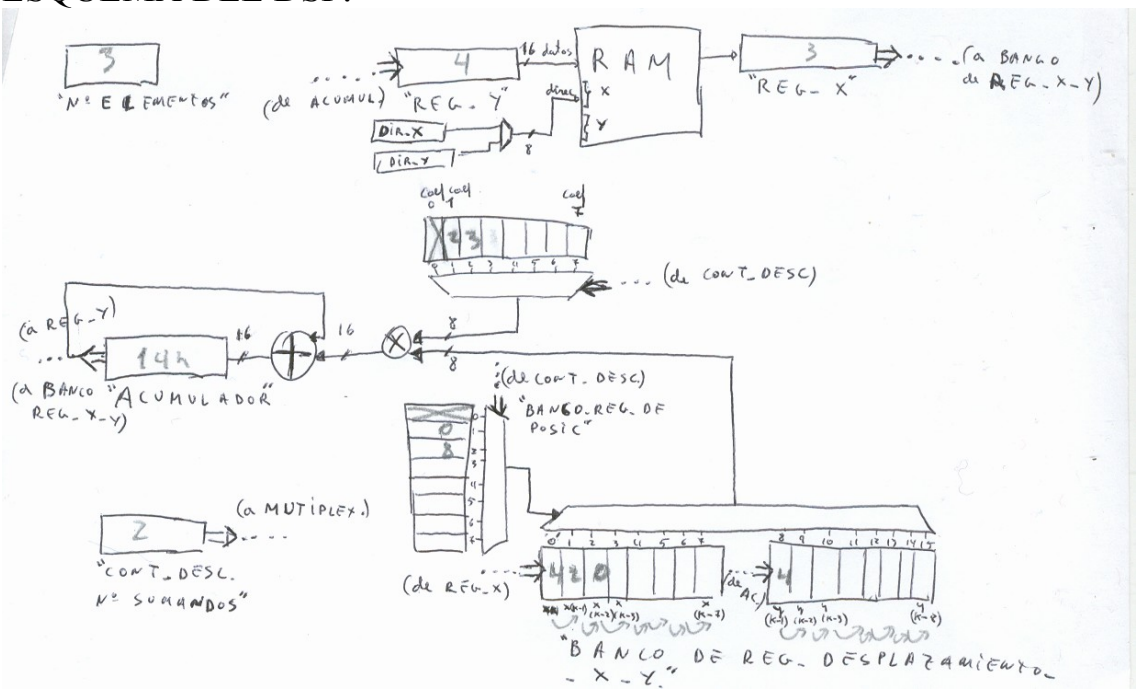
- 10xx)h: xx (valor con el que cargo DIRECCION_X).
- 20xx)h: xx (valor con el que cargo DIRECCION_Y)
- 300x)h: x (valor con el que cargo Nº_ELEMENTOS).
- 400x)h: x (valor con el que cargo CONT_DESC_Nº_SUMANDOS)

- 5yxx)h: y (nº de coeficiente [1,7]), xx (valor del coeficiente) en BANCO_REG_COEF
- 6yxx)h: y (nº de posición[1,7]), xx (valor de posición) en BANCO_REG_DESPL_X_Y

3) Como ponerlo en marcha:
Debemos fijarnos en el esquema de Max+plus del DSP.
1º PREPARACION: Escritos en la RAM los datos que indican como se ha de operar (tabla de x(k), coeficientes…), primero se establecen los bancos de registros, contadores…. con un pulso en **PREP** (su longitud no importa).
2º TABLA Y(K): Una vez finalizada la preparación (**busc**=0) se manda obtener la tabla y(k) con un pulso en **I** (su longitud no importa). Esta finaliza cuando **IOP**=0**.**
***CLK** es el reloj que mueve todo.

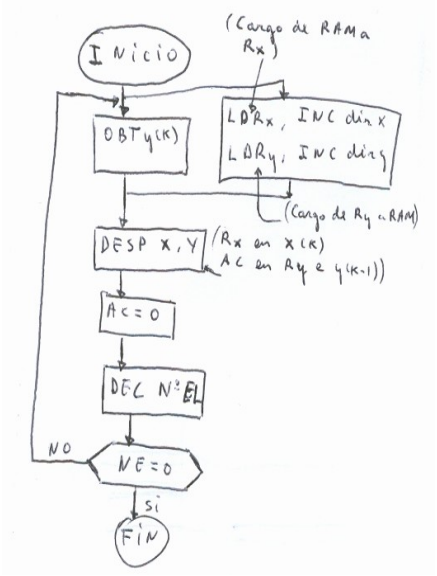
COMO FUNCIONA INTERNAMENTE:
Solo voy a explicar como se haría la obtención de los elementos de la tabla y(k). La preparación también tiene su complicación.
Este esquema simple del dsp servirá para aclarar las ideas:

ESQUEMA DEL DSP:



El siguiente organigrama muestra mejor que ninguna otra explicación como se va escribiendo la tabla y(k) en la RAM:

ORGANIGRAMA PARA OBTENER LA TABLA Y(K):



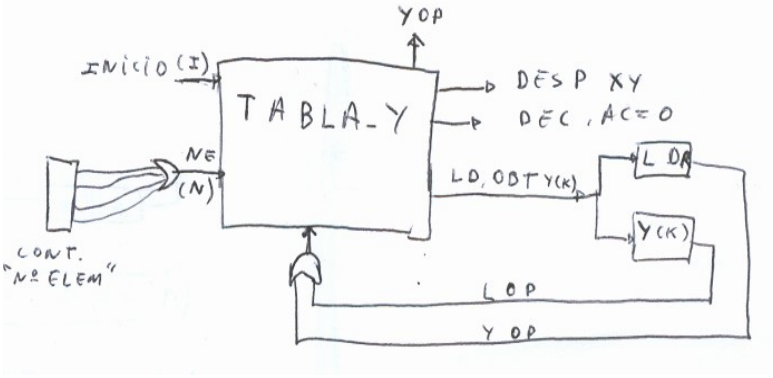
Cuando se inicia la obtención de la tabla y(k) (I=1) la unidad de control (UC) manda a otras unidades de control, OBTY(K) y LDR obtener el elemento y(k) (multiplicaciones, sumas…) y cargar el elemento apuntado por DIR_X en REG_X y escribir en RAM el REG_Y en la posición apuntada por DIR_Y, después se incrementan DIR_X y DIR_Y.

La obtención del elemento y(k) se hace así: el BANCO_DE _REG_DESPLAZAMIENTO_X_Y siempre tiene colocados en el instante k los elementos x(k), x(k-1).., y(k-1), y(k-2)… listos para ser multiplicados con los coefientes correspondientes del BANCO_REG_COEF. El CONT_DESC_Nº_SUMANDOS es cargado con el nº de sumandos y va decreciendo he indicando en cada instante que coef. con que elemento (x(k), x(k-1)….) se multiplica. Esto lo hace gracias a que va a la entrada de los dos multiplexores, uno que da el coef. directamente y el otro que da la posición de otro multiplexor que decide que elemento se multiplica. Estas posicones las contiene el BANCO_DE_REG_DE_POSIC. Tras cada multiplicación y suma (que es lógica combinacional) se recarga el ACUMULADOR con el valor que le este entrando del sumador. Llegado el CONT_DESC_Nº_SUMANDOS a 0 tenemos en el ACUMULADOR el elemnto y(k).

Obtenido y(k) se manda cargar en REG_Y e y(k-1) y el BANCO_DE _REG_DESPLAZAMINTO_X hace un desplazamiento al completo para tener los elementos listos para obtener el próximo y(k).
El ACUMULADOR se pone a 0.
Decremento el Nº_ELEMENTOS.
Compruebo si Nº_ELEMENTOS es 0 para terminar, si no, vuelvo a iterar.

Ahora muestro un simple esquema de cómo la UC que obtiene la tabla y(k), manda una señal (LD_OBTY(K)) a las otras dos unidades de control LDR e Y(K). Tambien vemos las entradas y salidas de control I, YOP, DESP_XY,….

ESQUEMA INTERNO DE LA UC_OBTENER_TABLA_Y(K):



En el esquema del DSP de Max+plus. Vemos la agrupación de la UC que obtiene la tabla y(k) (allí se llama UC_OBTENER_TABLA_Y(K)). La UC que prepara el DSP (allí se llama UC_PREPAR_REG/CONT). La unidad de procesamiento es el resto y la RAM es la memoria principal que se vé.

DOS EJEMPLOS DE SIMULACION:

Ejemplo1:

Los datos son :

- Ecuación: **y(k) = 2*x(k) + 3*y(k-1)**
- Tabla de x(k):
  - **x(0)=2** => y(0) = 2*2 + 3*0 = 4 (0004)h
  - **x(1)=4** => y(1) = 2*4 + 3*4 = 20 (0014)h
  - **x(2)=3** => y(2) = 2*3 + 3*20 = 66 (0042)h
  - **x(3)=1** => y(3) = 2*1 + 3*66 = 200 (00C8)h
- Posición de inicio en RAM de tabla x(k) dada: **000A)h**
- Posición de inicio en RAM de tabla a obtener y(k) en RAM: **0012)h**

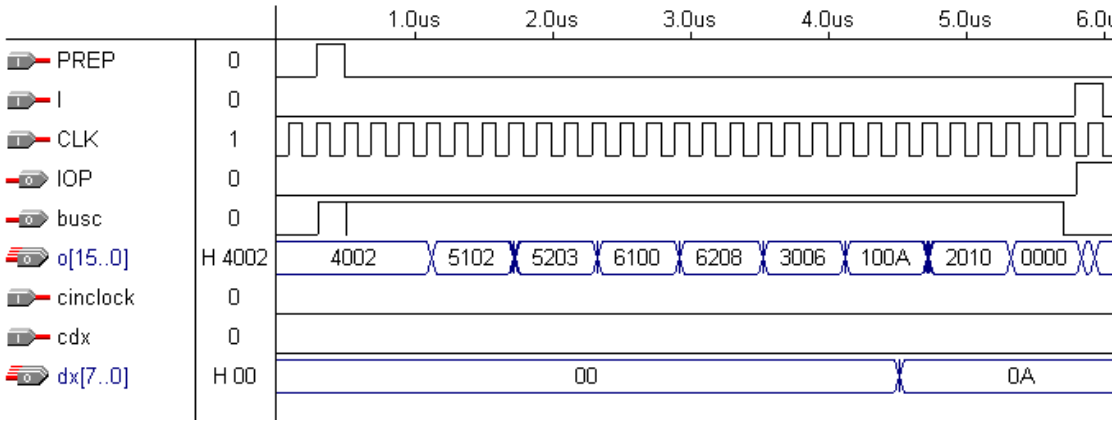
Según estos datos para preparar el dsp, el archivo ram.mif, de inicio de la RAM debe ser:

```
DEPTH = 256;    % Memory depth and width are required      %
WIDTH = 16;     % Enter a decimal number                   %

ADDRESS_RADIX = HEX; %Address and value radixes are optional%
DATA_RADIX = HEX;    %Enter BIN, DEC, HEX, or OCT; unless%

-- Specify values for addresses, which can be single address or range
```

```
CONTENT
        BEGIN
        0       :       4002;   %cont%
        1       :       5102;   %coef1%
        2       :       5203;   %coef2%
        3       :       6100;   %posic1%
        4       :       6208;   %posic2%
        5       :       3006;   %ne+2%
        6       :       100A;   %dx%
        7       :       2010;   %dy-2%
        8       :       0000;   %fin de prepara%

        A       :       0002;   %tabla de x%
        B       :       0004;
        C       :       0003;
        D       :       0001;
END ;
```

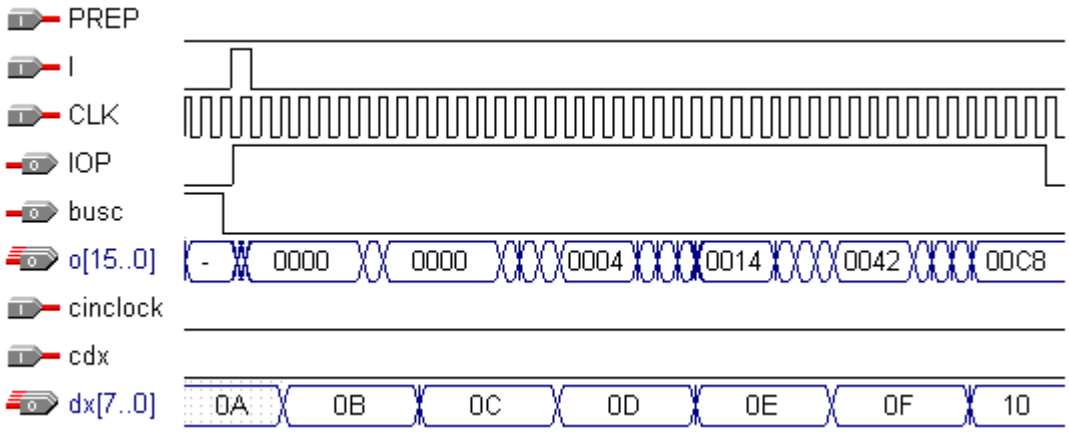
La simulación tiene tres partes:
1) Preparar:
Donde vemos como al poner a 1 la entrada PREP inicia la preparación y dura mientras busc está a 1.

PREPAR:



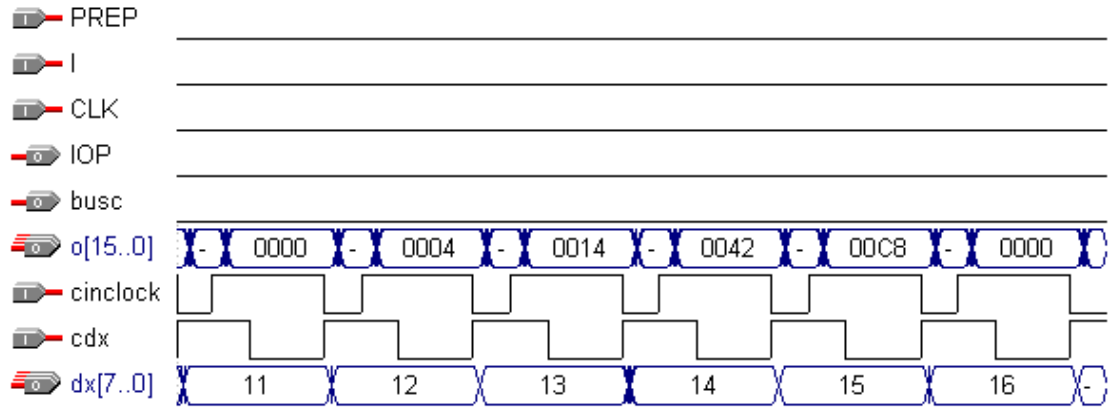
2) Obtener la tabla y(k):
Donde vemos como al poner la entrada I a 1 inicia la obtención de la tabla y dura mientras IOP esta a 1.

OBTENER:



3) Comprobar que la tabla y(k) esta en la RAM:
Donde al elevar cdx (ver esquema) se incrementa el contador DIRECCION_X cuya salida la da dx[7..0]. Este va cambiando la posición a leer en la RAM. Con cinclock se actualiza la salida de la RAM (o[15..0]). **Vemos que efectivamente la tabla y(k) se ha guardado en su lugar correcto (de la 12)h a la 15)h) y con los valores correctos (0004)h, 0014)h, 0042)h, 00C8)h)**.

COMPROBAR:



<u>Ejemplo2:</u>
Los datos son :
- Ecuación: **y(k) = 1*x(k-1) + 2*y(k-2)**
- Tabla de x(k):

x(0)=2	=>	y(0) = 1*0 + 2*0 = 0	(0000)h
x(1)=4	=>	y(1) = 1*2 + 2*0 = 2	(0002)h
x(2)=3	=>	y(2) = 1*4 + 2*0 = 4	(0004)h
x(3)=1	=>	y(3) = 1*3 + 2*2 = 7	(0007)h

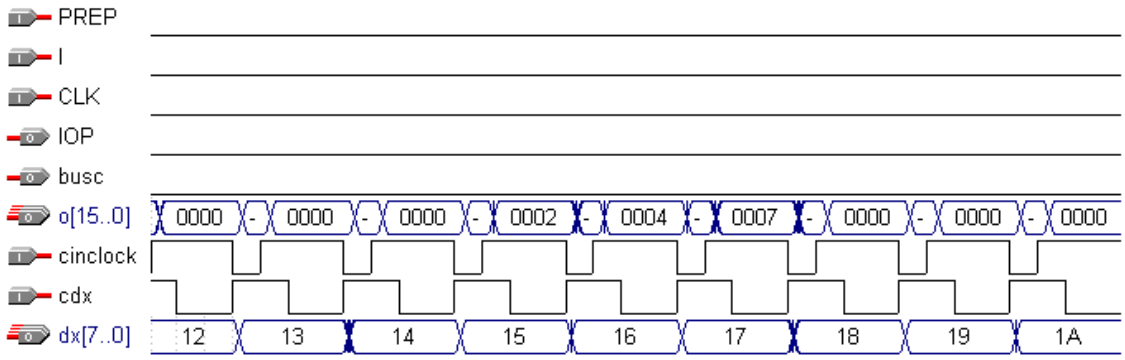
- Posición de inicio en RAM de tabla x(k) dada: **000B)h**

- Posición de inicio en RAM de tabla a obtener y(k) en RAM: **0014)h**
Según estos datos para preparar el dsp, el archivo ram.mif, de inicio de la RAM debe ser:

```
DEPTH = 256;    % Memory depth and width are required      %
WIDTH = 16;     % Enter a decimal number                   %

ADDRESS_RADIX = HEX;        % Address and value radixes are optional    %
DATA_RADIX = HEX;   % Enter BIN, DEC, HEX, or OCT; unless   %

CONTENT
        BEGIN
        0       :       4002;   %cont%
        1       :       5101;   %coef1%
        2       :       5202;   %coef2%
        3       :       6101;   %posic1%
        4       :       6209;   %posic2%
        5       :       3006;   %ne+2%
        6       :       100B;   %dx%
        7       :       2012;   %dy-2%
        8       :       0000;   %fin de prepara%

        B       :       0002;   %tabla de x%
        C       :       0004;
        D       :       0003;
        F       :       0001;
END ;
```

Al igual que antes, al final de la simulación se ve que la tabla y(k) ha quedado bien almacenada a partir de la posición 00014)h:

COMPROBAR:



TIEMPO QUE TARDA EN OBTENER UN ELEMENTO Y(K):

Los ejemplos y mas simulaciones muestran que para obtener un solo elemento de la tabla y(k) se tarda:
- Ecuación de 2 sumandos: 7 pulsos de reloj.
- Ecuación de 3 sumandos: 9 pulsos de reloj.
- Ecuación de 4 sumandos: 11 pulsos de reloj.

Que vemos que son tiempos muy buenos.

OBSERVACIONES FINALES Y POSIBLES MEJORAS:

- El multiplicador en este caso es combinacional pero podría ser secuencial ya que la UC_OBTENER_TABLA_Y(K) esta preparada para mandar una señal (m) cada vez que se debe ejecutar una multiplicación y está esperando hasta que finaliza (señal mop).

- Este DSP no hace ecuaciones en las que aparezcan restas. La memoria RAM es muy pequeña (256 palabras). Los valores máximos de los elementos x(k), y(k) y los coef. no deben superar los 255. El numero de sumandos no puede ser mas de 8. El numero de elementos maximo de la tabla es 16…. **Todo esto son restricciones muy importantes para un DSP real que necesita capacidad y precisión. Pero todo esto es facil de solucionar ya que lo mas importante es tener un modelo de base que sea rápido. El resto es ir perfeccionando algunos detalles.**