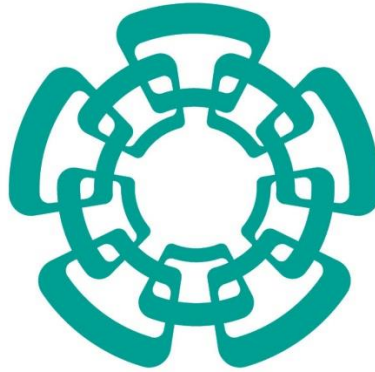


CENTRO DE INVESTIGACION Y DE ESTUDIOS
AVANZADOS DEL IPN

COPROCESADOR DE CONVOLUCIÓN



Cinvestav

Nombre del Estudiante

Eduardo Mateo Martínez Solís

Nombre del diplomado

Programa de talento altamente especializado

Nombre de la Materia

Metodología de diseño de System On Chips

Nombre del Profesor

Vidkar Anibal Delgado Gallardo

Fecha:09/05/24

CENTRO DE INVESTIGACION Y DE ESTUDIOS AVANZADOS DEL IPN

1) Descripción del problema

Implementar un coprocesador de convolución utilizando un enfoque top-down.

¿Cómo voy a recibir los datos?

Una de las señales a convolucionar será almacenada de manera interna en una memoria ROM, la segunda señal la tendremos como entrada en una memoria RAM externa, la cual nos proporcionará dos señales:

- dataY.
- sizeY.

Como señal de entrada de control al IP core tendremos:

- start.

¿Cómo voy a entregar los datos?

Los datos por entregar corresponden a la señal resultante de la convolución, estos datos se guardarán en una memoria RAM externa, por lo que el modulo tendrá las siguientes señales de salida para manipular dicha memoria:

- memZ_addr.
- dataZ.
- writeZ.

Como señales de salida también tendremos:

- busy.
- done.

¿Cuándo debe empezar a funcionar mi modulo?

En función de la señal de entrada start.

¿Tenemos limitación de tiempo de ejecución o de área?

No, ni de tiempo ni de área.

CENTRO DE INVESTIGACION Y DE ESTUDIOS AVANZADOS DEL IPN

2) Diagrama de caja negra y definición de señales de entrada y salida

En la *figura 1* se muestra el diagrama de caja negra, generado a partir de la descripción del problema y de los requerimientos solicitados en el mismo.

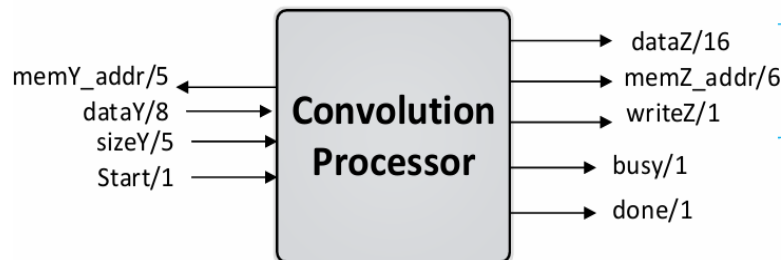


Figura 1.- Diagrama de caja negra del convolucionador.

Señales

Entradas:

Control

- start. - Señal de 1 bit que indicará cuando el módulo debe empezar a trabajar.

Dato

- dataY. – Señal de 8 bits que corresponde al dato almacenado en la dirección memY_addr de la memoria externa Y.
- sizeY. – Señal de 5 bits que indica la cantidad de muestras guardadas en la memoria Y.

Salidas:

Control

- writeZ. – Señal de 1 bit correspondiente a un enable para la memoria de salida.
- busy. – Señal en de 1 bit que indica cuando el IP core está trabajando y cuando no.
- done. - Señal de 1 bit “one shot” que indica cuando terminó la convolución.

CENTRO DE INVESTIGACION Y DE ESTUDIOS AVANZADOS DEL IPN

Dato

- memY_addr. – Señal de 5 bits que indica a la memoria externa Y, la posición del dato que debe proporcionar al módulo.
- memZ_addr. – Señal de 6 bits que direcciona a una posición de la memoria de salida, el dato resultante.
- dataZ. – Señal de 16 bits que corresponde al dato a almacenar en la dirección establecida por memZ_addr.

Relación entre señales

Cuando start está en un estado lógico bajo, la señal de busy, done y writeZ son cero, indicando que no se está aplicando ninguna operación.

Cuando start se pone en un estado lógico alto, busy también tomará un valor alto durante el tiempo que tarde en acabar el proceso, writeZ se pondrá en 1 cada que un dato esté listo para escribirse en la memoria de salida.

Al finalizar el proceso, busy toma un valor bajo y la señal de done toma el valor lógico alto durante un ciclo de reloj.

3) Pseudocódigo

En el *código 1* vemos que cuando la señal start se activa y pone en estado alto la señal busy señala el comienzo de la inicialización de variables y señales auxiliares necesarias para la ejecución del algoritmo. Una de estas señales es sizeZ, que determina el tamaño de la señal de salida y juega un papel crucial en la lógica del programa, reiniciando los registros de la variable temporal Z y la dirección de memoria memY_addr si sizeZ es mayor que la dirección actual de la memoria interna. Si este caso no se cumple, se asume que la convolución está completa y el sistema se prepara para recibir una nueva señal de inicio después de asegurarse que busy está bajo.

El flujo del algoritmo continúa en un bucle principal donde se verifica si la dirección de memoria de Y es menor que sizeY si se cumple, se usa una señal auxiliar shifted para gestionar el desplazamiento adecuado de la señal Y, permitiendo la multiplicación de todos sus valores por un valor fijo de la señal H. Si no se cumple, se procede a marcar el dato de Z[H] como listo, elevando la señal writeZ, liberando la dirección y el dato de salida, y se incrementa la dirección de la memoria H para preparar la siguiente iteración.

Dentro de este bucle, con la señal shifted actualizada, se asegura el desplazamiento correcto de la señal Y el número de veces necesario. Con todos los datos preparados y recuperados de las direcciones de memoria respectivas, se ejecuta la operación de convolución y se incrementa la dirección de memoria de Y.

CENTRO DE INVESTIGACION Y DE ESTUDIOS AVANZADOS DEL IPN

El proceso culmina con la activación de la señal done a través de un one shot, señalando que el proceso está completamente finalizado.

```
while Start = 0
end while

done = 0;
busy = 0;
memY_addr = 0;
sizeY_temp = sizeY;
shifted = 0;
writeZ = 0;

sizeZ = sizeH + sizeY_temp -1;

while memH_addr < sizeZ

    busy = 1
    dataZ_temp=0;
    memY_addr=0;

    while memY_addr < sizeY

        shifted = memH_addr - memY_addr
        if (shifted >= 0 && shifted < sizeH)
            dataH = H[shifted];
            dataY = Y[memY_addr];
            dataZ_temp += dataH * dataY
        end if
        memY_addr++
    end while

    dataZ = dataZ_temp;
    memZ_addr = memH_addr;

    writeZ = 1;
    writeZ = 0;
    memH_addr++

end while

    busy = 0;
    done = 1;
if start = 0
goto 1
else goto 40
```

CENTRO DE INVESTIGACION Y DE ESTUDIOS AVANZADOS DEL IPN

```
int main() {
    int sizeH = 5;
    int H[sizeH] = {0,8,9,2,1};
    bool writeZ = false;
    bool done = false;
    bool busy = false;
    bool start = false;
    int sizeY = 5;
    int Y[sizeY] = {0,8,9,2,1};
    int memY_addr=0;
    int memZ_addr=0;
    int memH_addr=0;
    int dataZ_temp=0;
    int shifted=0;
    int dataH=0;
    int dataY=0;
    int dataZ=0;
    int sizeZ = sizeH + sizeY - 1;
    int Z[sizeZ] = {0};
    start = true;
    while(start){
        while (memH_addr < sizeZ) {
            busy=true;
            dataZ_temp = 0;
            memY_addr = 0;
            while (memY_addr < sizeY) {
                shifted = memH_addr - memY_addr;
                if (shifted >= 0 && shifted < sizeH) {
                    dataH = H[shifted];
                    dataY = Y[memY_addr];
                    dataZ_temp += dataH * dataY;
                }
                memY_addr++;
            }
            dataZ = dataZ_temp;
            memZ_addr=memH_addr;
            Z[memZ_addr] = dataZ;
            memH_addr++;
            writeZ = true;
            writeZ = false;
        }
        start = false; }
    busy = false;
    done = true;
    return 0;
}
```

CENTRO DE INVESTIGACION Y DE ESTUDIOS AVANZADOS DEL IPN

En la tabla 1 se muestran los vectores utilizados para comparar los resultados obtenidos entre el código de validación y la herramienta de software matemático, Matlab.

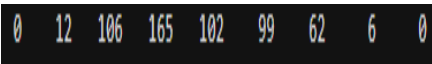
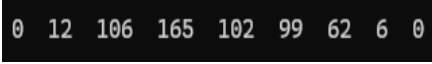


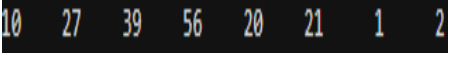


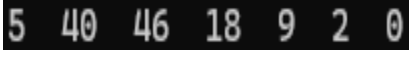




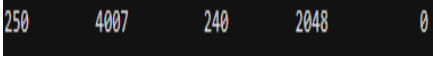

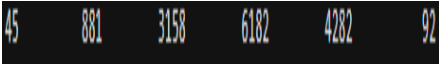
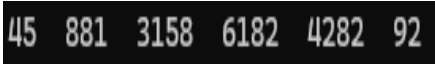
Señales		Resultados	
H[n]	Y[n]	Matlab	Algoritmo en C++
{0, 1, 8, 7, 2, 6}	{12, 10, 1, 0}		
{1, 0, 5}	{8, 2, 0, 4, 6}		
{2, 5, 6, 8, 0, 1}	{5, 1, 2}		
{5, 0, 1, 0}	{1, 8, 9, 2}		
{1, 0}	{5, 9}		
{0, 8, 9, 2, 1}	{0, 8, 9, 2, 1}		
{250, 7, 128, 0}	{1, 16}		
{9, 25, 46}	{5, 84, 92, 2}		

Tabla 1.- Comparación de resultados entre Matlab y código en C++.

Una vez que el pseudocódigo ha sido probado para asegurar su correcto funcionamiento, podemos afirmar que hemos establecido un “golden model”. Este modelo representa una versión funcional del algoritmo que se desea implementar. Contar con un algoritmo validado es fundamental, ya que sirve como referencia precisa para el resto de la metodología. Con este modelo, estamos ahora en condiciones de avanzar a la siguiente etapa, la elaboración de los diagramas del sistema.

CENTRO DE INVESTIGACION Y DE ESTUDIOS AVANZADOS DEL IPN

4) Diagrama ASM

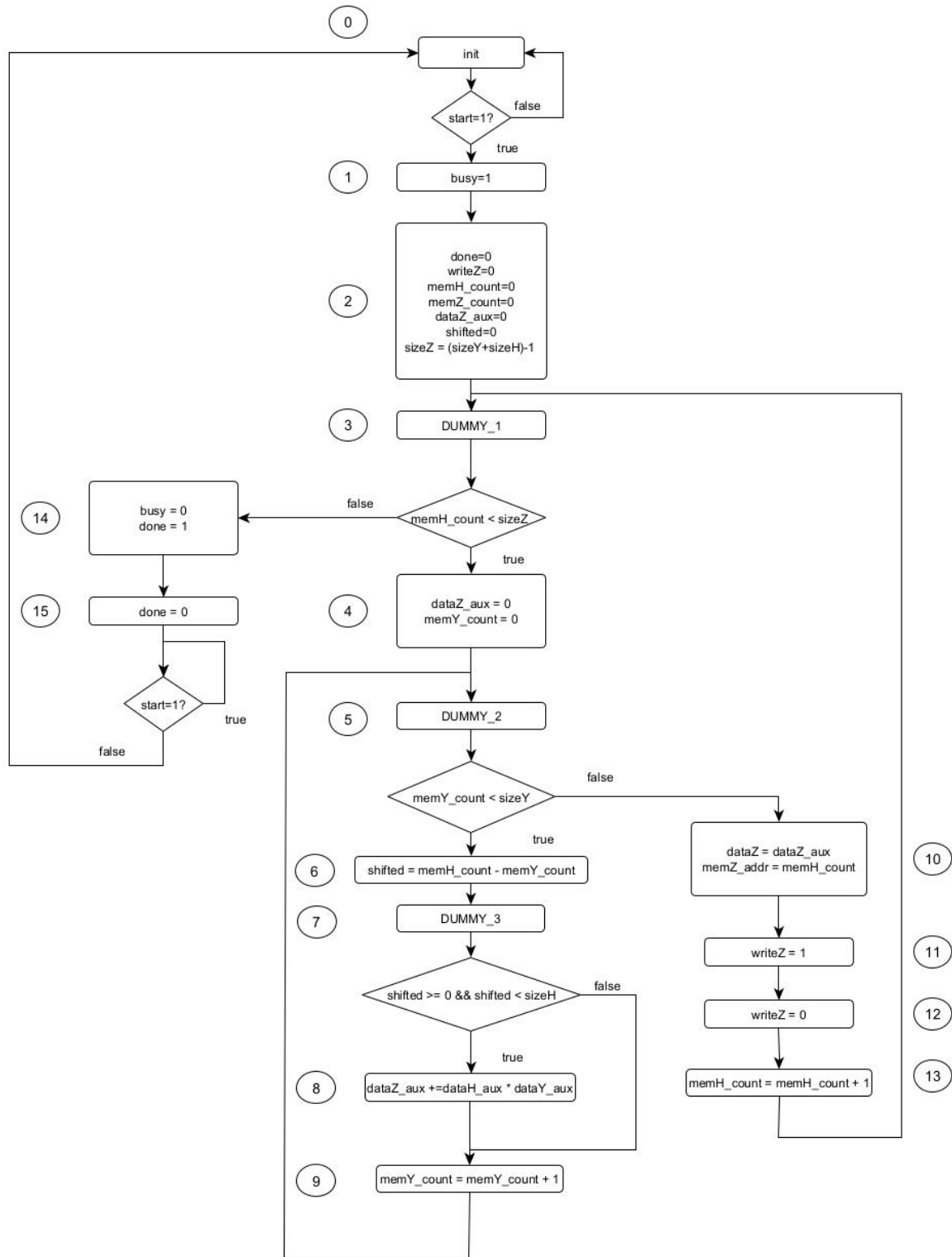


Figura 2.- Diagrama ASM.

CENTRO DE INVESTIGACION Y DE ESTUDIOS AVANZADOS DEL IPN

Observando el diagrama ASM final de la *figura 2* y los bloques óptimos de la *figura 3* correspondientes a una versión sin optimizar, notamos que se eliminaron bloques que pasaron a ser combinacionales, se realizó la unión de bloques gracias a que las señales involucradas no dependían del estado anterior, caso como el del bloque 2 y 3 y del par 8 y 9.

Importante mencionar que parte de las modificaciones con versiones anteriores también tuvo que ver con la aplicación de las expresiones RTL en todos los estados y con la suma de estados “dummy”, que colaboran con las sincronización de las señales en la maquina de estados.

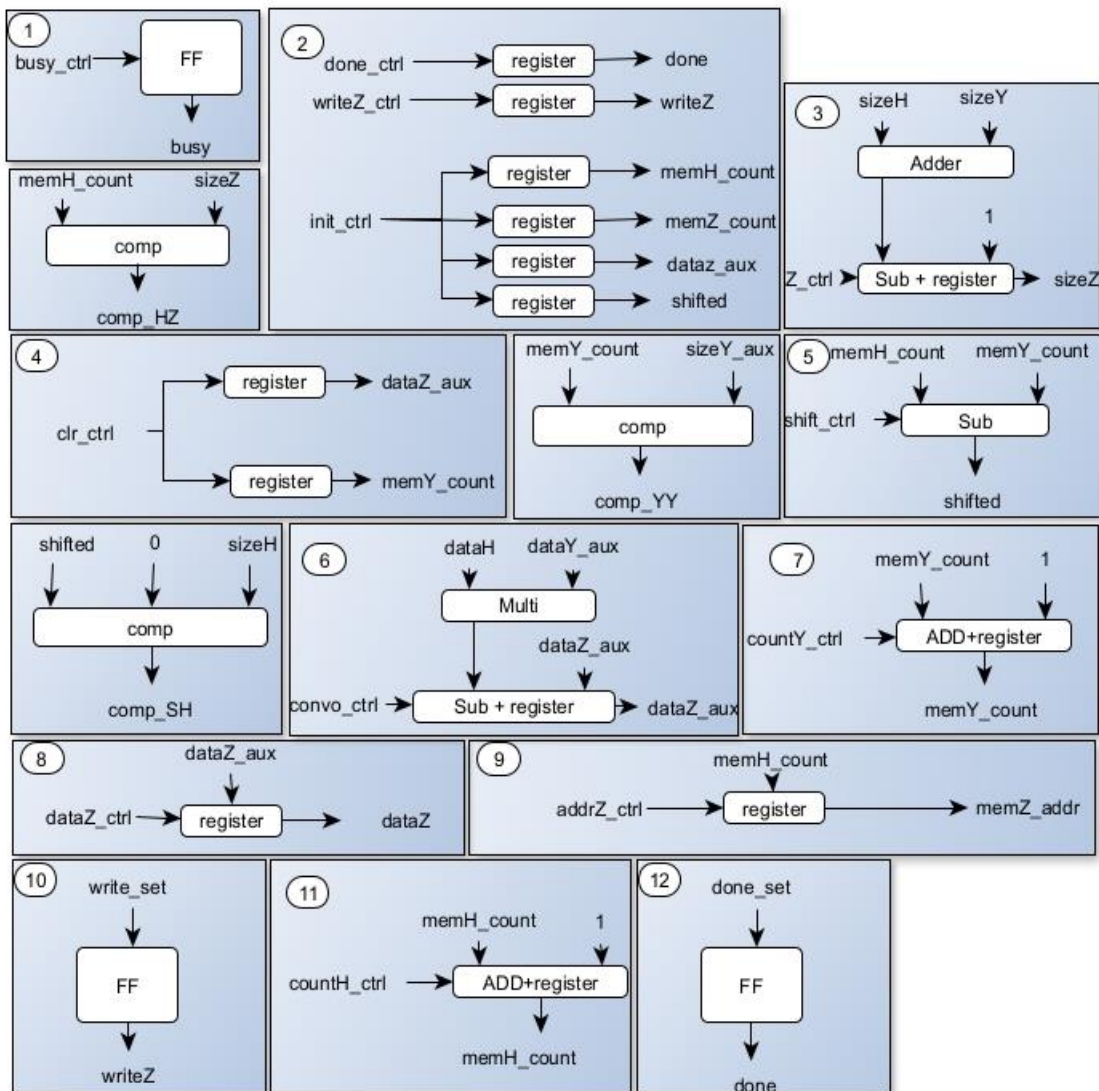


Figura 3.- Bloques óptimos.

CENTRO DE INVESTIGACION Y DE ESTUDIOS AVANZADOS DEL IPN

5) Datapath y máquina de estados

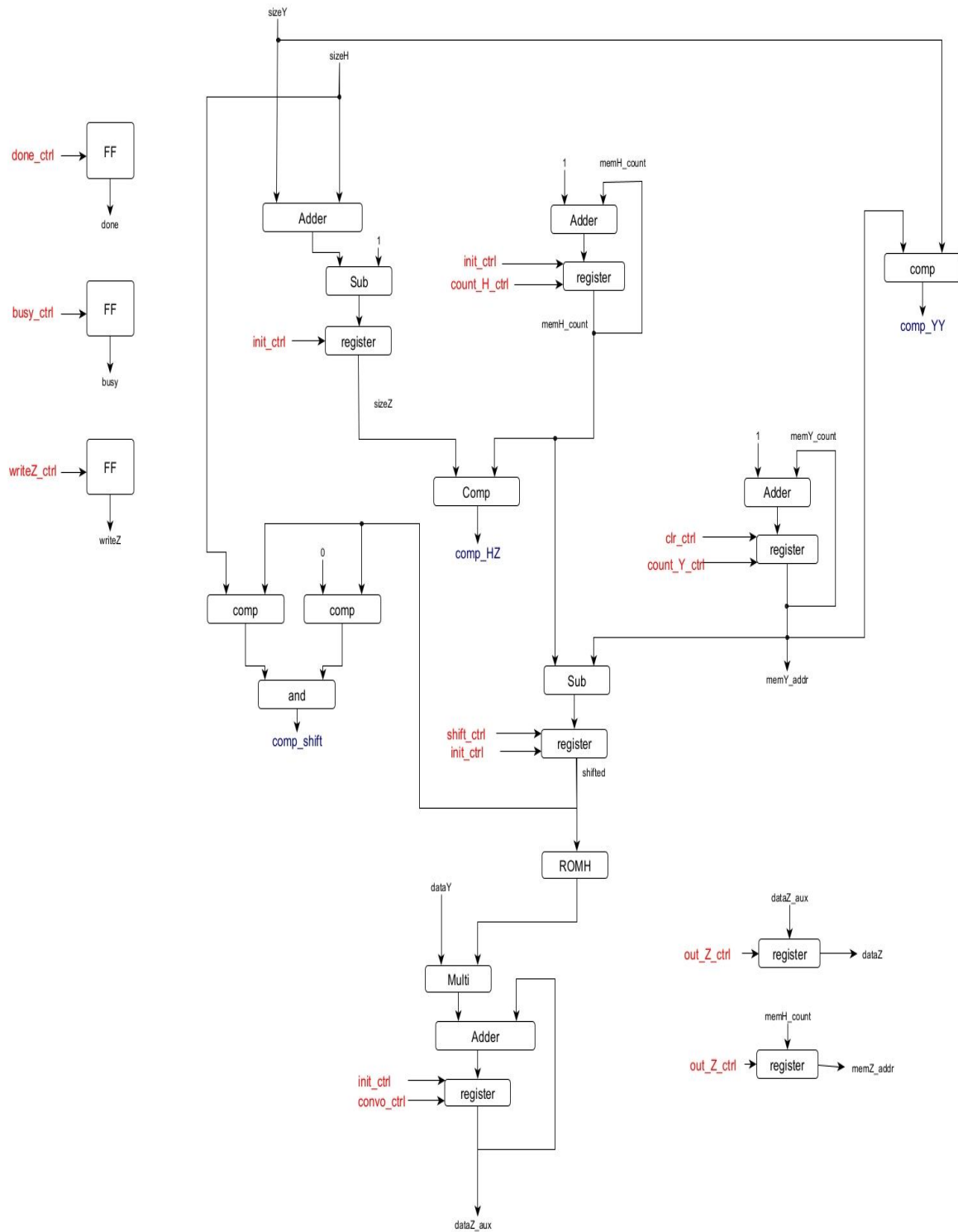


Figura 4.- Datapath.

CENTRO DE INVESTIGACION Y DE ESTUDIOS AVANZADOS DEL IPN

Tanto el datapath de la *figura 4* como la máquina de estados de la *figura 5* son producto del seguimiento de la metodología planteada en el curso, de la retroalimentación del docente y de las optimizaciones anteriormente mencionadas.

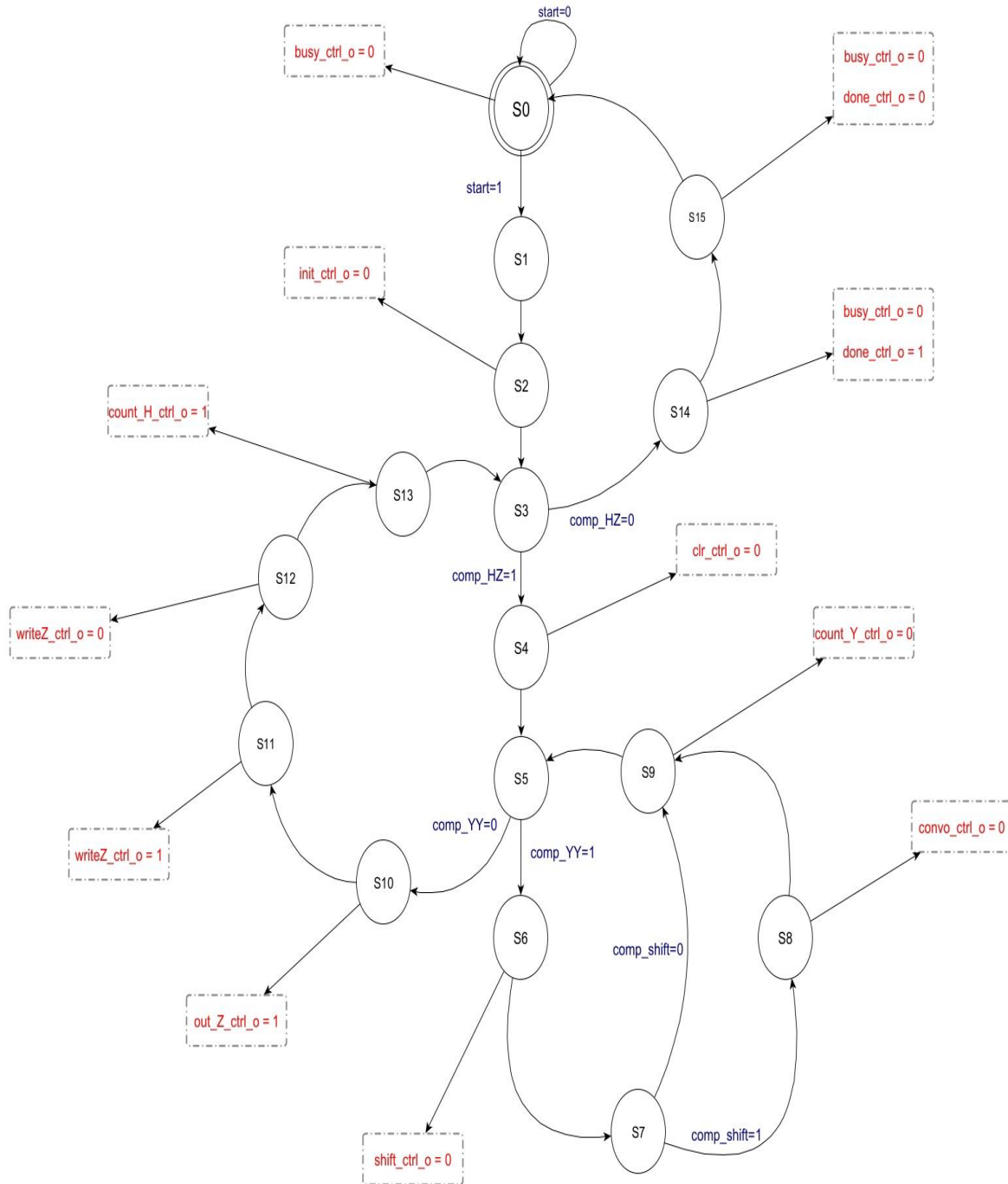


Figura 5.- Máquina de estados.

CENTRO DE INVESTIGACION Y DE ESTUDIOS AVANZADOS DEL IPN

6) Resultados de simulación

En la figura 6 se muestra el esquemático Top Level, donde se observa que se aplicó un estilo de descripción estructural, por lo que la creación de bloques y la conexión entre ellos con el uso de cables se hace presente.

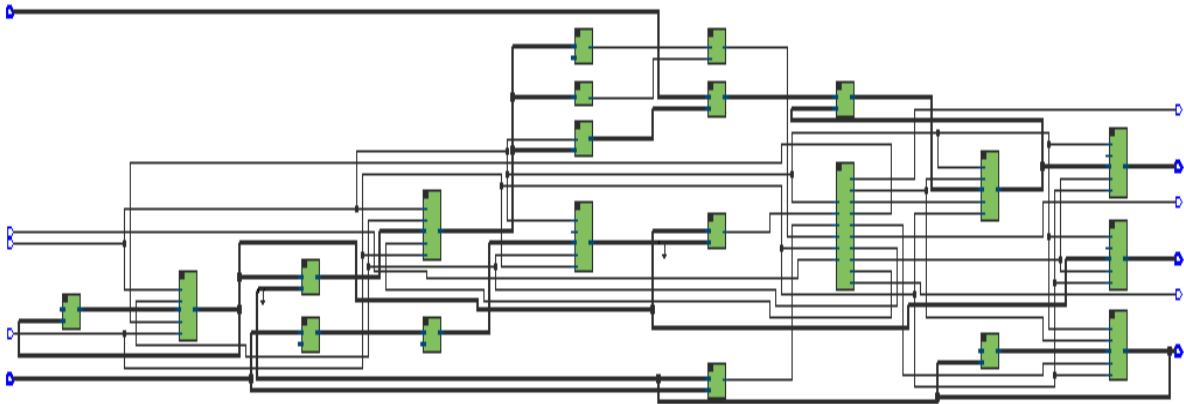


Figura 6.- Esquemático Top Level.

Para agilizar las pruebas del coprocesador de convolución se realizó un código que genera de manera aleatoria vectores con la cantidad de elementos que se desee y los guarda en archivos de texto correspondientes a las memorias utilizadas en el core, este es mostrado en el *código 3*.

```
#include <iostream>
#include <cstdlib>
#include <cstdliblib>
#include <ctime>
#include <iomanip>
int main() {
    int size_H = 5;
    int size_Y = 10;

    const char* ruta_MEMH = "D:/emms1/Escritorio/DIPLOMADO/Metodologia
SoC/MEMORY_H";
    const char* ruta_MEMORY = "D:/emms1/Escritorio/DIPLOMADO/Metodologia
SoC/MEMORY_Y";

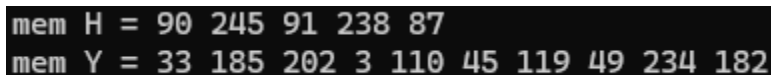
    int* mem_H = new int[size_H];
    int* mem_Y = new int[size_Y];
    FILE *archivo_H = fopen(ruta_MEMH, "w");
    FILE *archivo_Y = fopen(ruta_MEMORY, "w");
    ...
}
```

CENTRO DE INVESTIGACION Y DE ESTUDIOS AVANZADOS DEL IPN

```
...
    if (archivo_H == NULL || archivo_Y == NULL) {
        std::cerr << "Error al abrir los archivos.\n";
        if (archivo_H) fclose(archivo_H);
        if (archivo_Y) fclose(archivo_Y);
        delete[] mem_H;
        delete[] mem_Y;
        return 1;
    }
    srand(time(NULL));
    for (int i = 0; i < size_H; i++) {
        mem_H[i] = rand() % 255;
        fprintf(archivo_H, "%02X\n", mem_H[i]);
    }
    for (int i = 0; i < size_Y; i++) {
        mem_Y[i] = rand() % 255;
        fprintf(archivo_Y, "%02X\n", mem_Y[i]);
    }
    fclose(archivo_H);
    fclose(archivo_Y);
    std::cout << "mem H = ";
    for (int i = 0; i < size_H; i++) {
        std::cout << mem_H[i] << " ";
    }
    std::cout << std::endl;
    std::cout << "mem Y = ";
    for (int i = 0; i < size_Y; i++) {
        std::cout << mem_Y[i] << " ";
    }
    std::cout << std::endl<<std::endl;
    delete[] mem_H;
    delete[] mem_Y;
    std::cout << "Valores generados correctamente";
    return 0;
}
```

Código 3.- Código generador de archivos de memorias de prueba.

Al correr el código nos muestra en base decimal los valores guardados en cada archivo de texto identificados con el nombre de la memoria a las que se les relacionó, como se muestra en la figura 7.



```
mem H = 90 245 91 238 87
mem Y = 33 185 202 3 110 45 119 49 234 182
```

Figura 7.- Resultado de la ejecución del código 3.

CENTRO DE INVESTIGACION Y DE ESTUDIOS AVANZADOS DEL IPN

Si se revisan los archivos de texto se encuentran los valores correspondientes, guardados de manera vertical y en hexadecimal como lo mostrado en la figura 8.

1	5A
2	F5
3	5B
4	EE
5	57

1	21
2	B9
3	CA
4	03
5	6E
6	2D
7	77
8	31
9	EA
10	B6

Figura 8.- Archivos de texto generados con el código 3.

Una vez realizada la simulación y muestreadas las señales de interés se obtiene el waveform de la figura 9. En dicha simulación se muestra el rango completo de una convolución entre una señal con 5 valores y otra con 10.

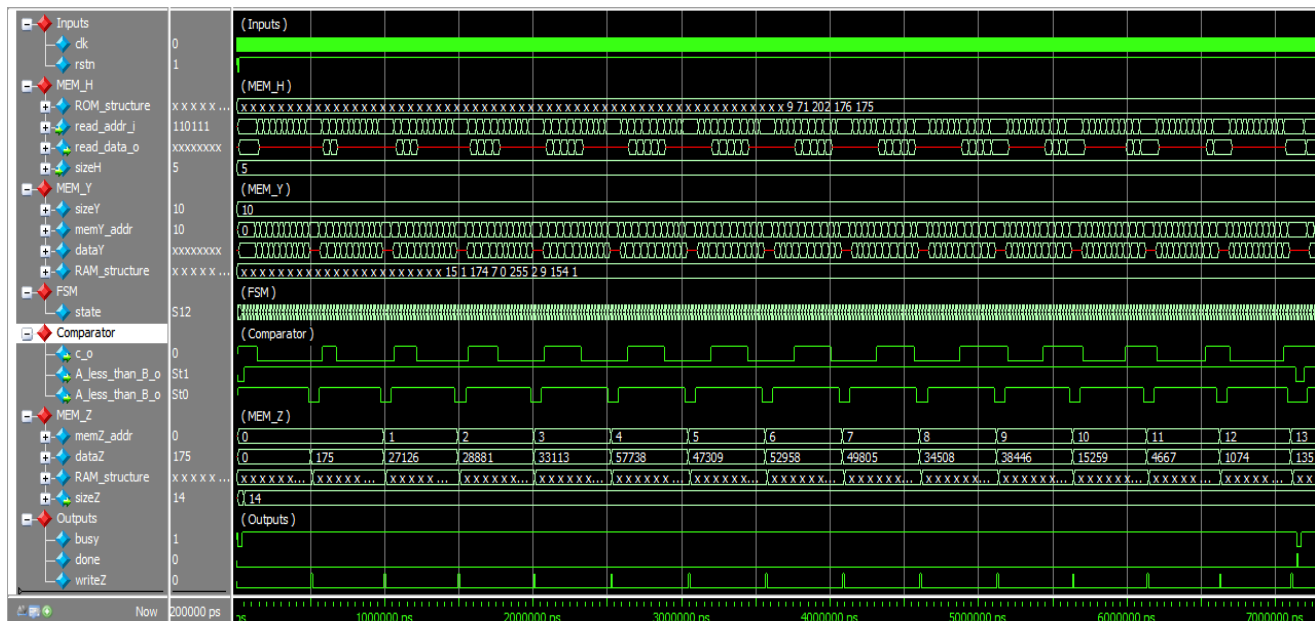


Figura 9.- Waveform del convolucionador.

Se observa como la señal de done se activa durante un ciclo de reloj una vez que el proceso de convolución termina, como las direcciones de las memorias avanzan y como el resultado de salida se va actualizando junto con una señal de enable para que se pueda escribir en la memoria RAM de salida. En esta simulación también se observa que el tiempo que le tomó en activarse a la bandera de done fue de aproximadamente 700 ciclos de reloj, lo anterior se sabe gracias a la unidad de tiempo que marca la herramienta y el periodo declarado en la cama de pruebas.

CENTRO DE INVESTIGACION Y DE ESTUDIOS AVANZADOS DEL IPN

En la figura 10 de muestra el área correspondiente al IP core una vez que se compiló y sintetizo en Quartus.

Logic utilization (in ALMs)	43 / 15,880 (< 1 %)
Total registers	91
Total pins	46 / 314 (15 %)
Total virtual pins	0
Total block memory bits	0 / 2,764,800 (0 %)
Total DSP Blocks	1 / 84 (1 %)

Figura 10.- Área del IP core.

Por último, en la figura 11 se muestra la frecuencia máxima a la que el reloj podrá funcionar sin problemas dentro del diseño propuesto.

	Fmax	Restricted Fmax	Clock Name	Note
1	186.88 MHz	186.88 MHz	clk	

Figura 11.- Frecuencia máxima.

Conclusión:

El desarrollo del coprocesador de convolución a partir de un enfoque top-down, diseñado de manera estructural mediante bloques básicos, siguiendo un datapath y una máquina de estados claramente definidos por la metodología adoptada, culminó cumpliendo las especificaciones y objetivos iniciales, dando por hecho que el core realiza la operación de convolución de manera correcta utilizando las señales tanto de entrada como de salida indicadas para el Top Level. Sin embargo, a pesar de que el diseño cumplió con la funcionalidad, la velocidad de operación podría mejorar, por lo que me lleva a considerar una mayor optimización del diseño en futuras revisiones.