Centro de Investigación y de Estudios Avanzados del Instituto Politécnico Nacional (CINVESTAV)

Diplomado en Desarrollo de Sistemas Embebidos.

Curso: Metodología de diseño de System-on-Chip.

Hassan Jaret Bolard Melendrez.

Dr. Vidkar Anibal Delgado Gallardo.

CONVOLUCIOADOR

INDICE.

Descripción del problema.

Implementar un coprocesador de convolución utilizando el enfoque de arriba hacia abajo.

Entradas: Datos almacenados en la memoria X y la memoria Y. Tamaño de las señales almacenadas en cada memoria. Señal de inicio.

Salidas: Señal de finalización (de un solo uso). Señal de ocupado. Almacenar el resultado en la memoria X.

Diagrama de caja negra y definición de señales de entrada y salida.

sizeY, sizeX: Serán entradas con un tamaño de 5 bits cada una.

dataX,dataY: Serán las entradas por donde se recibirán datos de la memoria "X" y la memoria "Y" respectivamente.

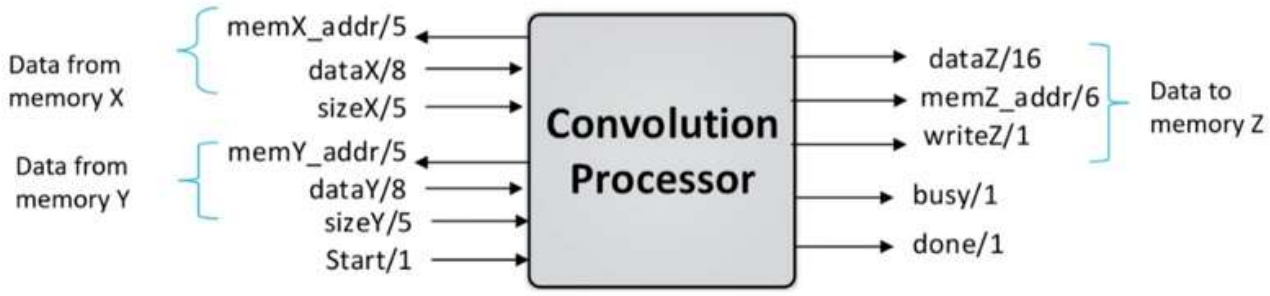
dataZ: Sera la dirección de salida del procesador de convolución y se almacenara en una memoria "Z".

memX_addr,memY_addr: Serán entradas donde se estará captando las direcciones de memoria de donde se están proviniendo dataX y dataY.

Start: Esta entrada permite la iniciasion del proceso de convolución al recibir un 1, esta provendrá de una máquina de estados.

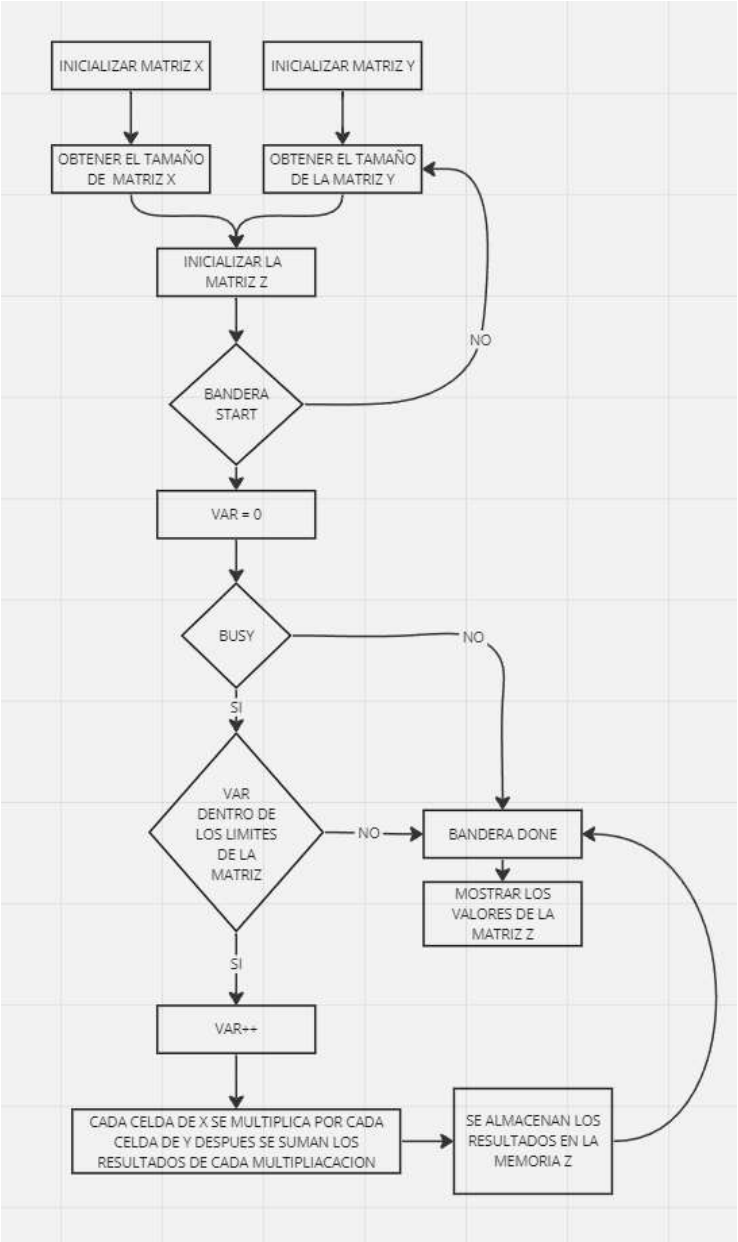
busy: Sera una salida que nos indica si el procesador de convolución ya esté operando en algún estado.

done: Esta salida nos indicara si nuestro proceso de convolución ya se concluyó con la finalidad de pasar al siguiente estado o dejar de operar



Pseudocodigo.

Para la implementación de nuestro sistema como segundo paso diseñaremos un diagrama de flujo para ilustrar los pasos que seguirá nuestro código.



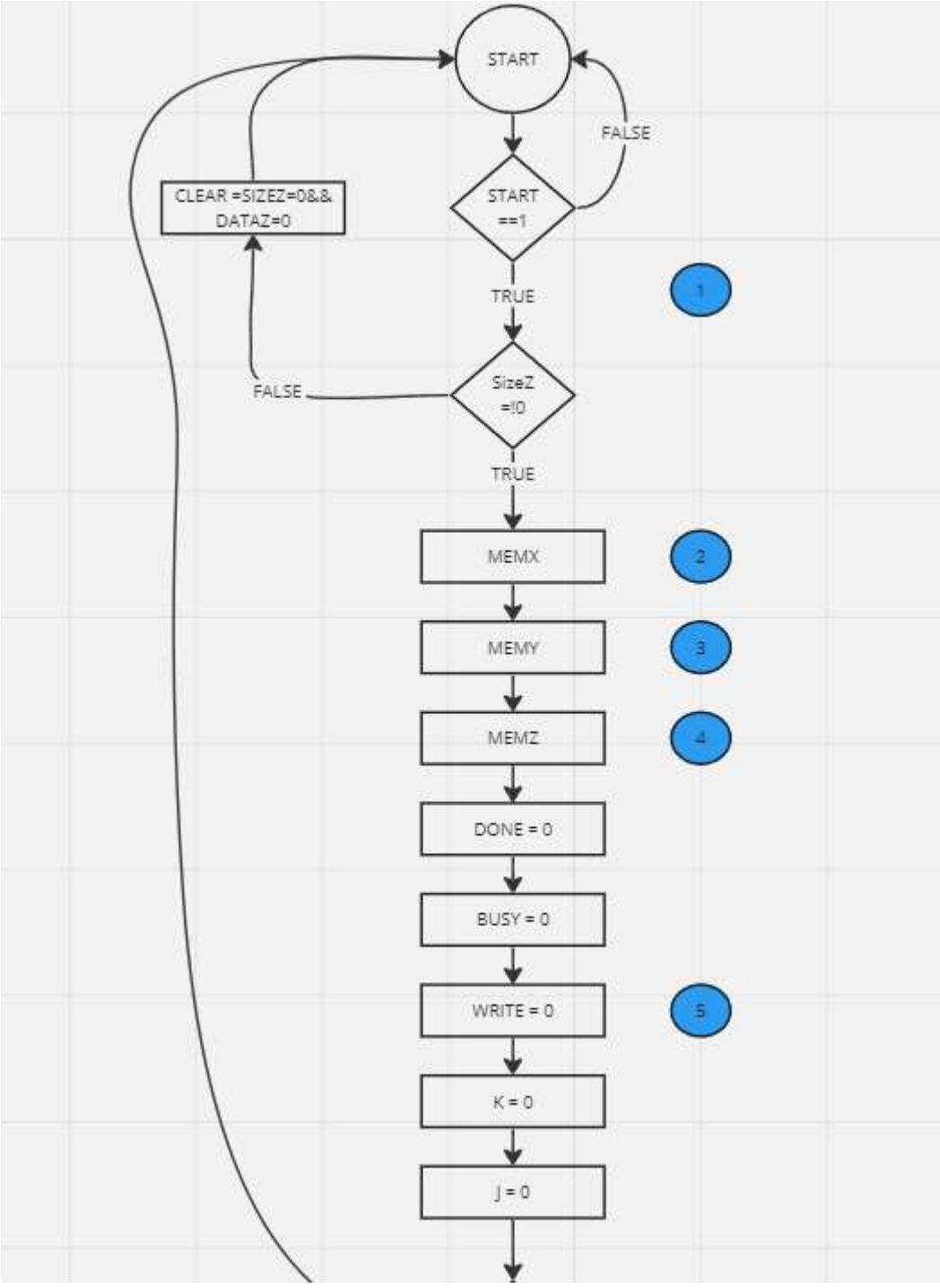
El diagrama del pseudocodigo nos explica que al inicio se definen las matrices “X”, “Y” y “Z” que tomaran la función de memorias además se inicializan las variables busy y start para indicar el estado en el que se encuentra nuestro código. Si ambas matrices cumplen con el tamaño adecuado se inicializa el proceso de convolución multiplicando y sumando los valores de las matrices es en este instante que se activa la bandera busy y se desactivara hasta que los resultados de la convolución se almacenen en la matriz Z.

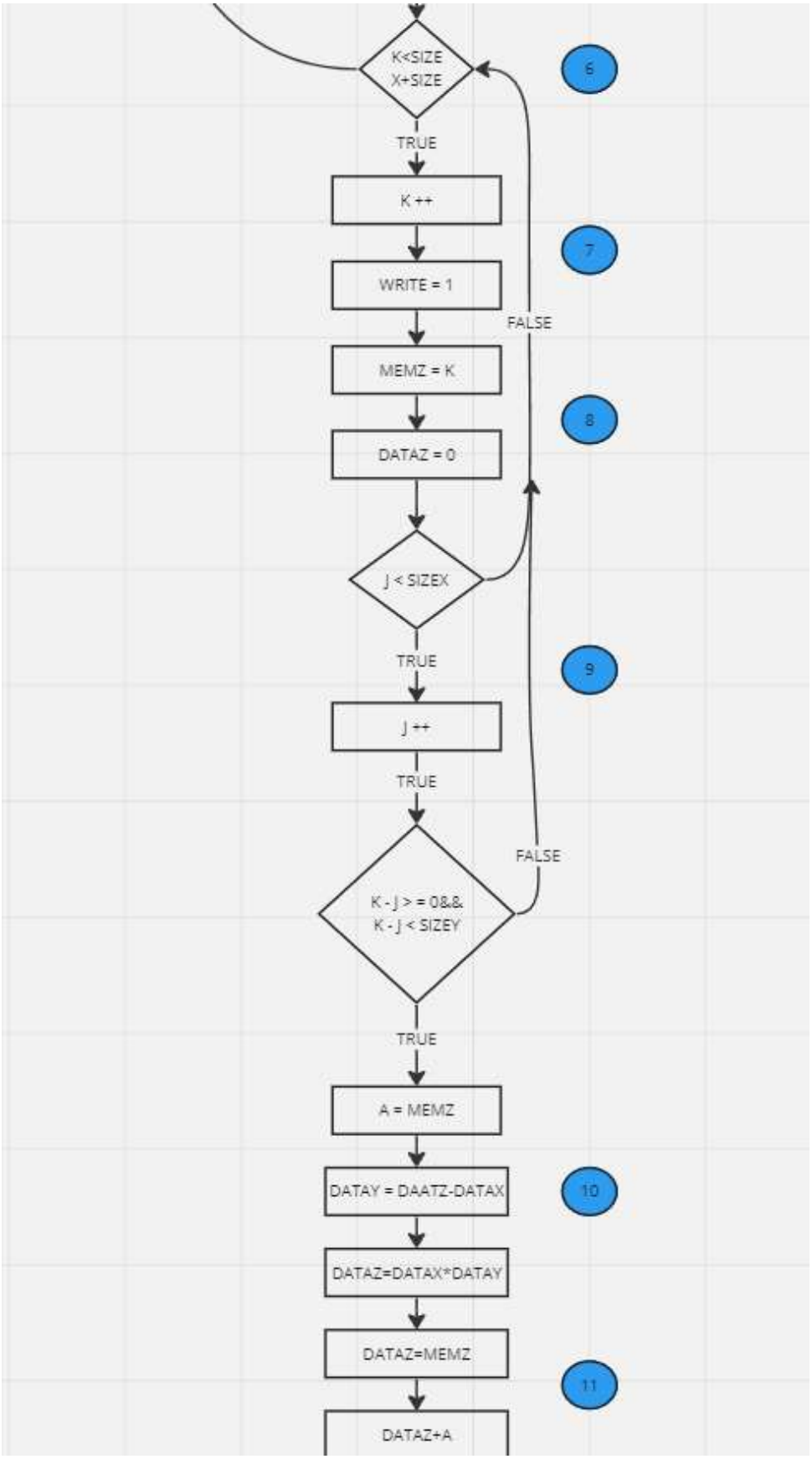
Código en C y resultados.

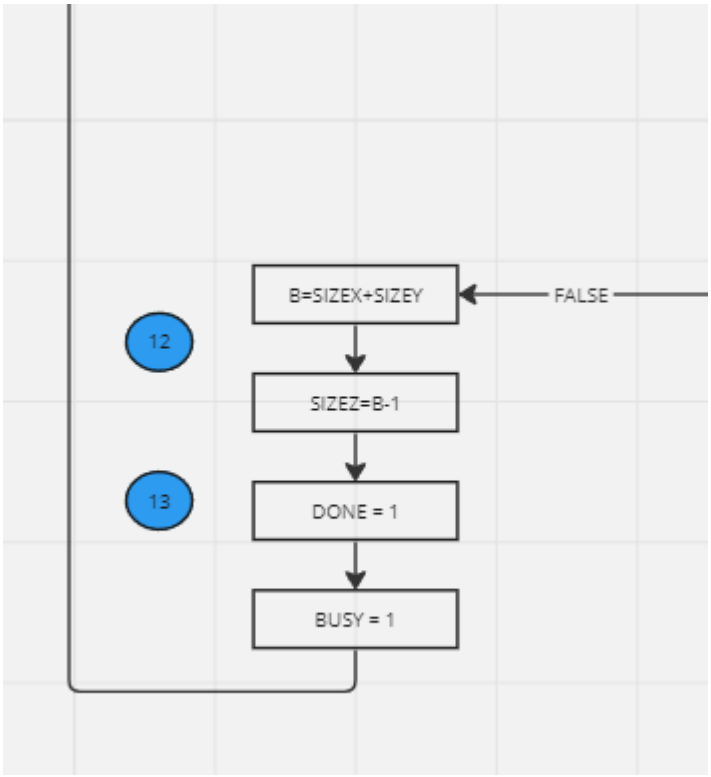
```c
#include <stdio.h>
#include <stdlib.h>

void convolve(int N, int x[N][N], int y[N][N], int z[2*N-1][2*N-1]) {
    int size = 2 * N - 1;

    // Inicializar la matriz z con ceros
    for (int i = 0; i < size; i++) {
        for (int j = 0; j < size; j++) {
            z[i][j] = 0;
        }
    }

    // Realizar la convolución
    for (int i = 0; i < N; i++) {
        for (int j = 0; j < N; j++) {
            for (int k = 0; k < N; k++) {
                for (int l = 0; l < N; l++) {
                    z[i + k][j + l] += x[i][j] * y[k][l];
                }
            }
        }
    }
}

int main() {
    int N = 3; // Tamaño de las matrices x e y

    // Ejemplo de matrices x e y
    int x[3][3] = {
        {1, 2, 3},
        {4, 5, 6},
        {7, 8, 9}
    };

    int y[3][3] = {
        {9, 8, 7},
        {6, 5, 4},
        {3, 2, 1}
    };

    int z[2*N-1][2*N-1];

    convolve(N, x, y, z);

    // Imprimir la matriz z
    printf("Resultado de la convolucion:\n");
    for (int i = 0; i < 2*N-1; i++) {
        for (int j = 0; j < 2*N-1; j++) {
            printf("%d ", z[i][j]);
        }
        printf("\n");
    }

    return 0;
}
```

```
Resultado de la convolucion:
9 26 50 38 21
42 94 154 106 54
90 186 285 186 90
54 106 154 94 42
21 38 50 26 9

Process returned 0 (0x0)   execution time : 0.050 s
Press any key to continue.
```

Diagrama ASM

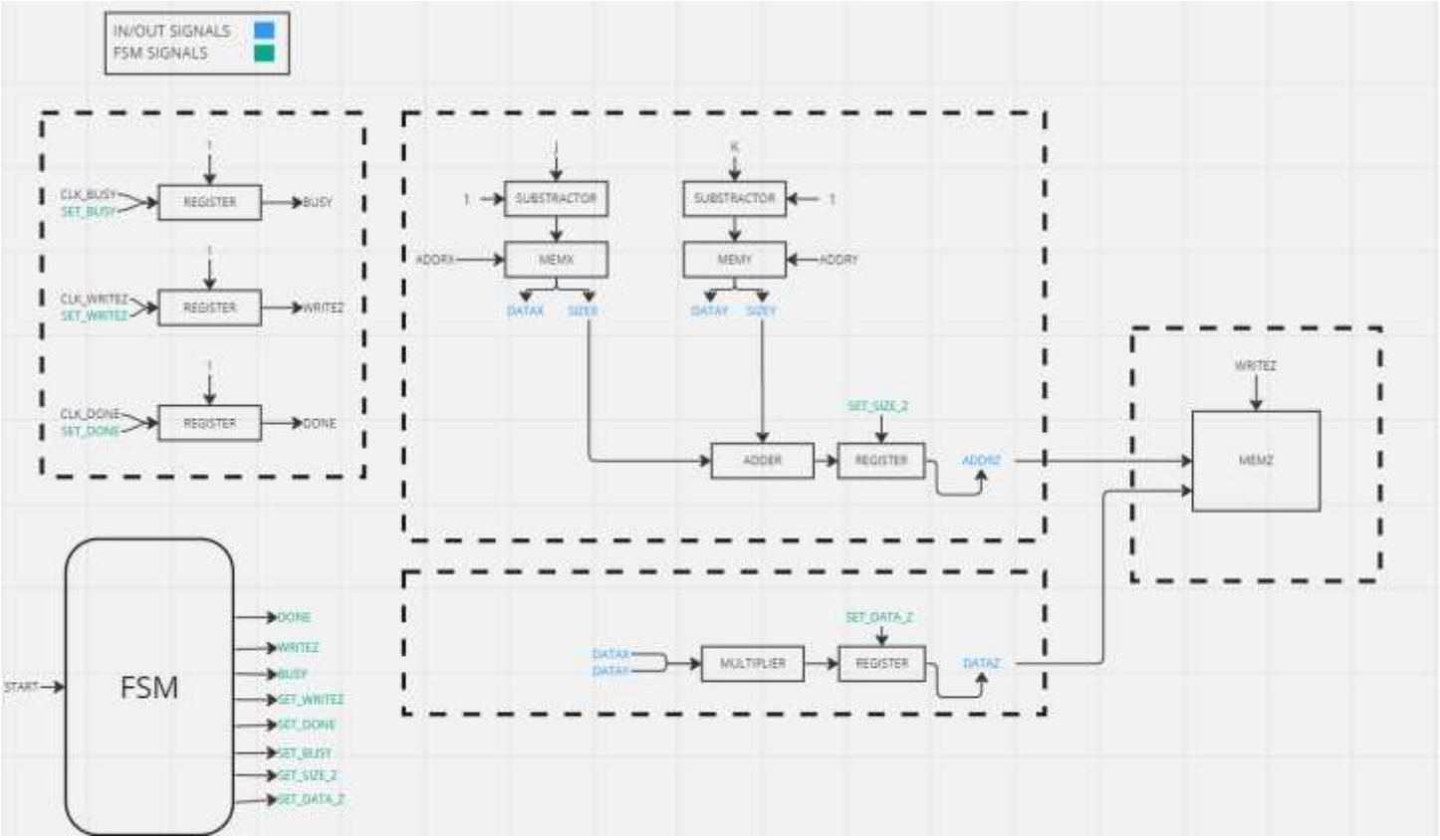



K<SIZE
X+SIZE
6
TRUE
K ++
7
WRITE = 1
FALSE
MEMZ = K
8
DATAZ = 0
J < SIZEX
TRUE
9
J ++
TRUE
FALSE
K - J > = 0&&
K - J < SIZEY
TRUE
A = MEMZ
DATAY = DAATZ-DATAX
10
DATAZ=DATAX*DATAY
DATAZ=MEMZ
11
DATAZ+A


B=SIZEX+SIZEY
FALSE
12
SIZEZ=B-1
13
DONE = 1
BUSY = 1

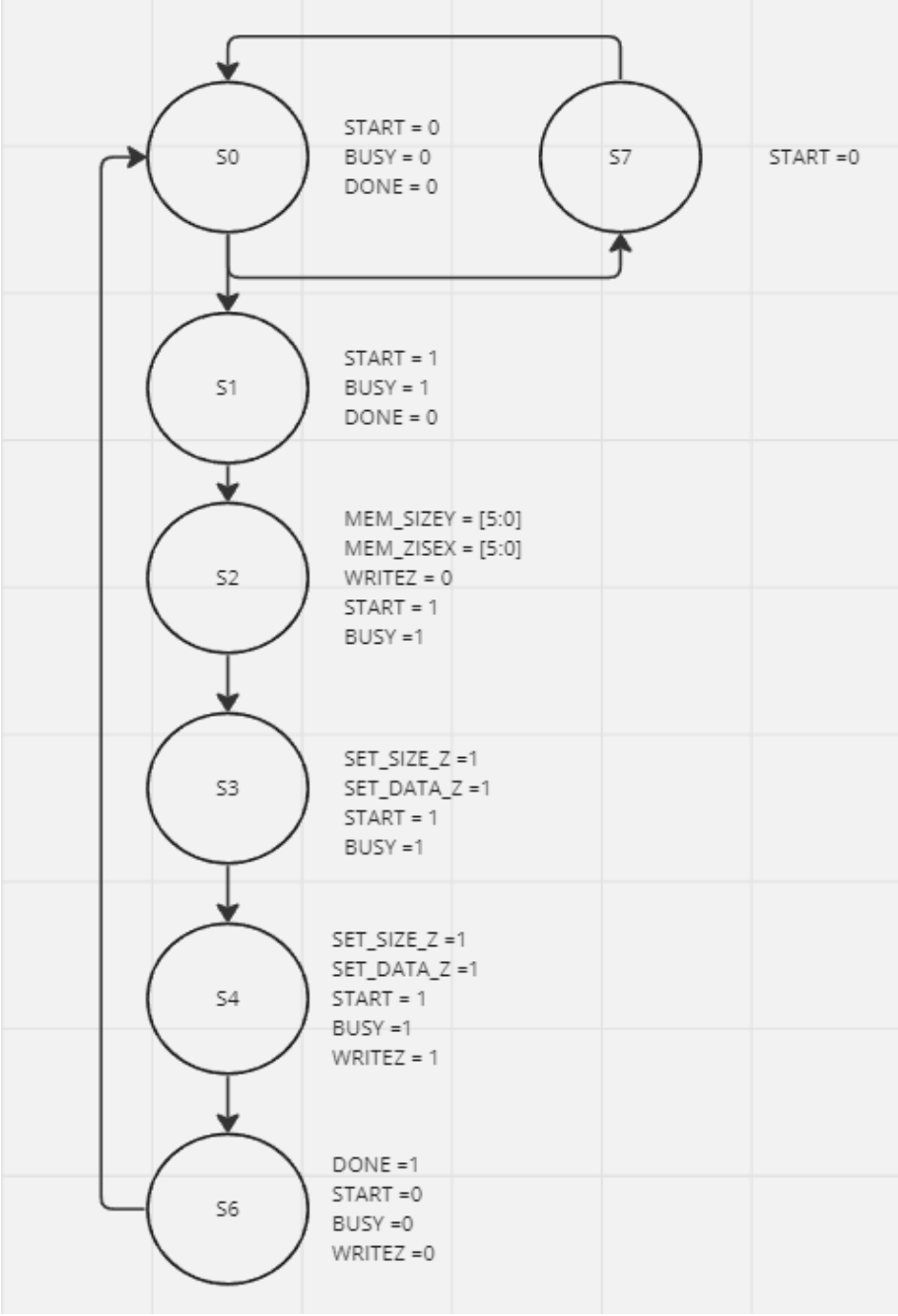
DATAPATH.

A partir de nuestro diagrama ASM podemos extraer los componentes principales que conformaran nuestro core, dentro de estos elementos encontramos 3 memorias para las entradas y salidas de datos, flip flops, sumadores, restadores, multiplicadores y registros. Considerando todos nuestros elementos y nuestro diagrama ASM procedemos a desarrollar nuestro datapath donde podemos hacer más comprensible nuestro diseño eliminando etapas y bloques que se repitan siempre y cuando estas tengan una señal en común.



MAQUINA DE ESTADOS (FSM).

El siguiente esquema representa la forma en la que se espera que se comporte nuestra maquina de estados, esta se encargara de gestionar el comportamiento que tendrá nuestro core convolucionador en base a los acontecimientos o estímulos que se esté perciba así se estará intercalando entre estados o “S” como se muestra de forma gráfica a continuación.



GENERADOR DE ESTMULOS.

```c
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

#define NUMEROS_A_GENERAR 5

int main() {
    FILE *archivo;
    int i, numero;

    // Abre el archivo para escribir
    archivo = fopen("numeros_hex.txt", "w"); // Cambia el nombre del archivo

    // Verifica si el archivo se abrió correctamente
    if (archivo == NULL) {
        printf("Error al abrir el archivo.");
        return 1;
    }

    // Semilla para la generación de números aleatorios
    srand(time(NULL));

    // Genera y escribe los números aleatorios en el archivo en formato hexadecimal
    for (i = 0; i < NUMEROS_A_GENERAR; i++) {
        numero = rand() % 16; // Números aleatorios entre 0 y 15
        fprintf(archivo, "%X\n", numero); // Usa "%X" para formato hexadecimal
    }

    // Cierra el archivo
    fclose(archivo);

    printf("Archivo generado correctamente.\n");

    return 0;
}
```