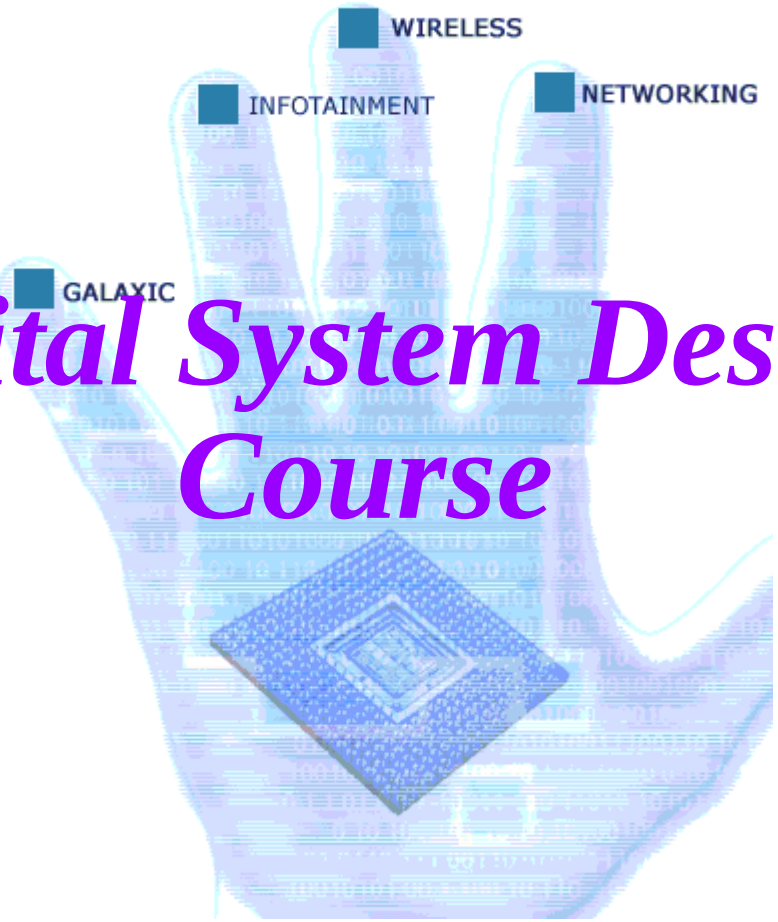
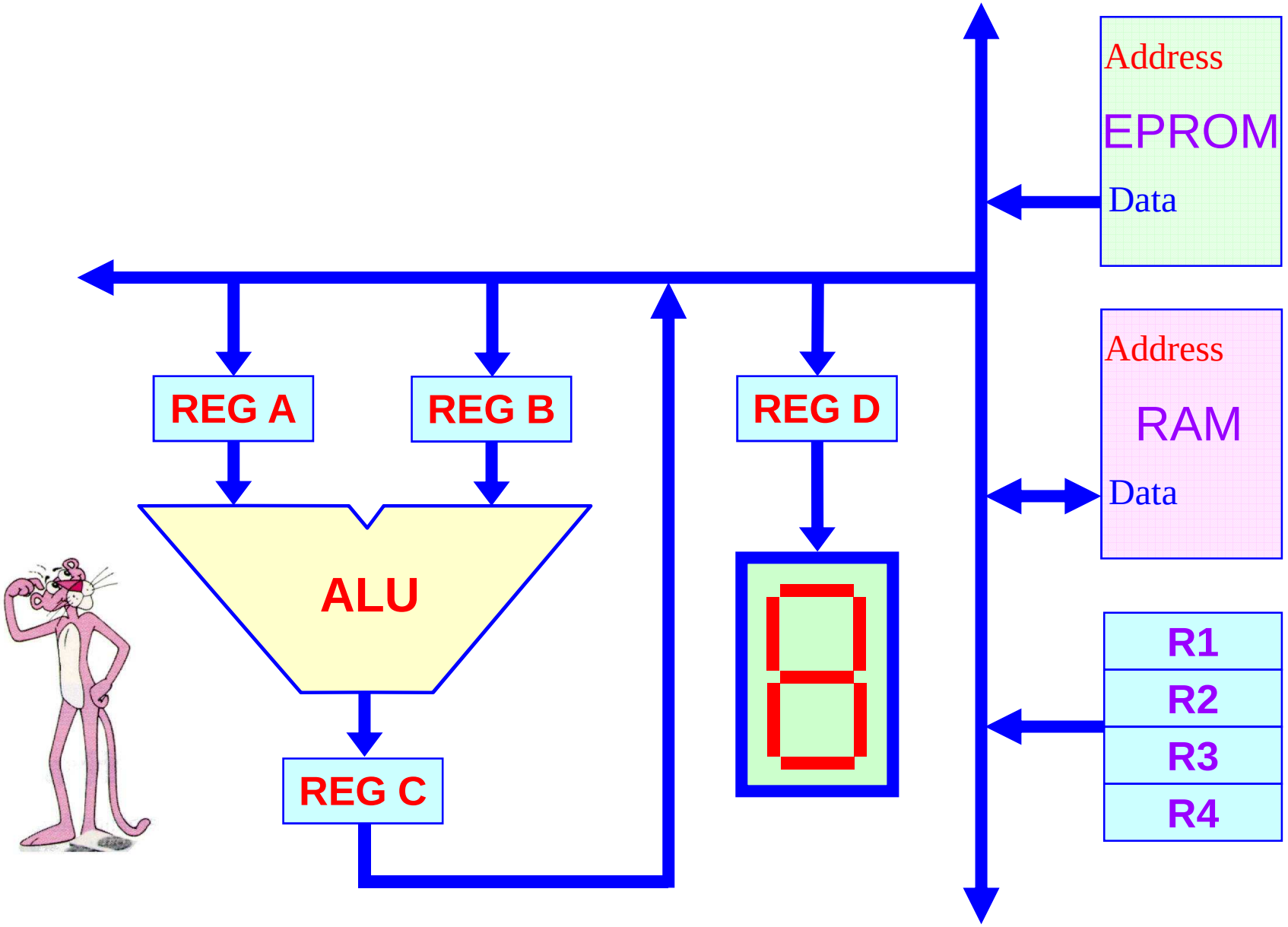


Digital System Design Course

Diseño de Controladores

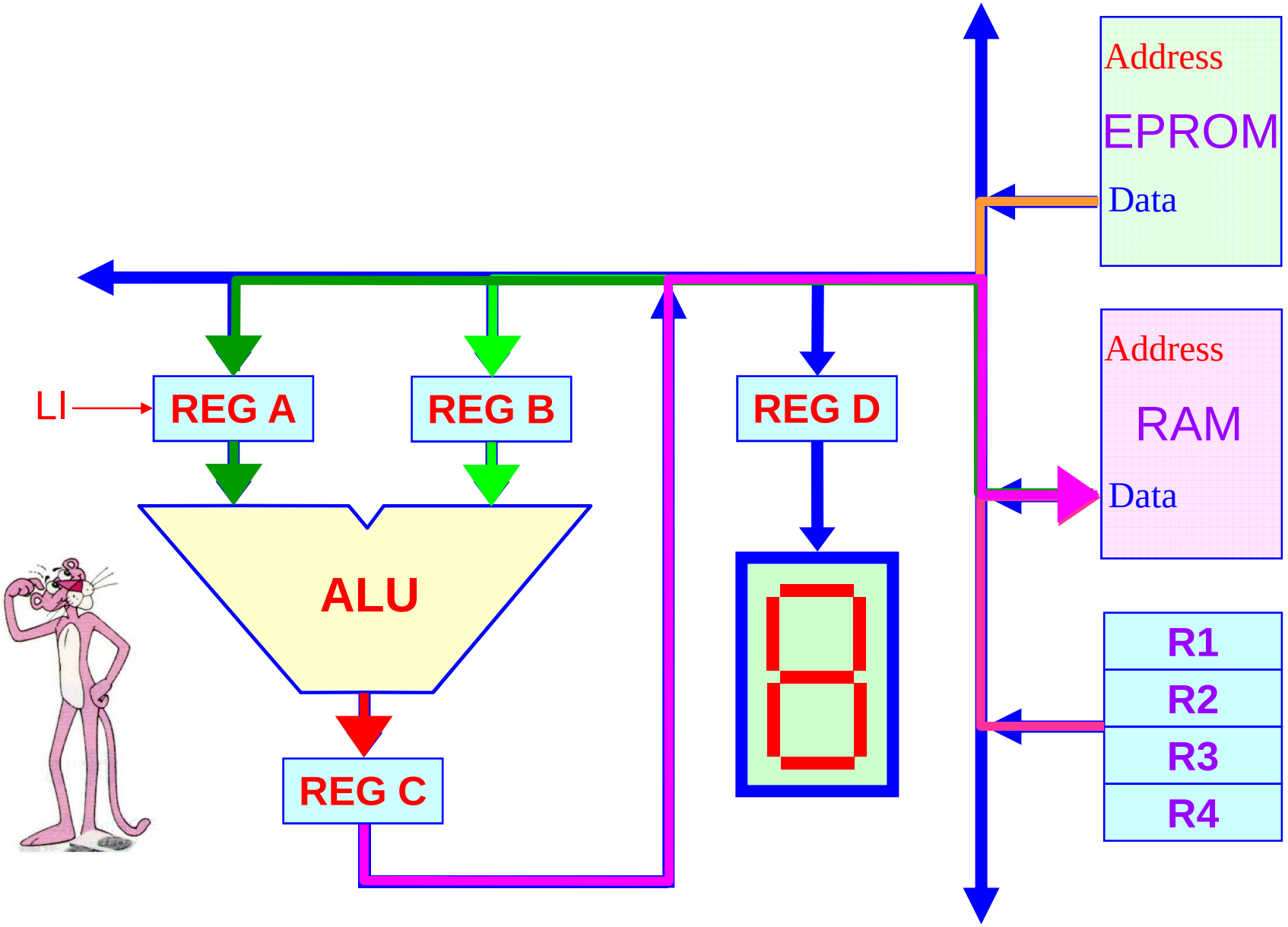
Diseño de Circuitos Controladores



Diseño de Circuitos Controladores

- Diseñar un circuito controlador o una FSM que permita generar las *señales de control* para los bloques funcionales del datapath y que además disponga de dos señales de control para realizar las siguientes operaciones:
  - Transferir cuatro datos desde la EPROM hacia la RAM y visualizar cada dato transferido
  - Transferir cuatro datos desde los registros hacia la RAM y visualizar cada dato transferido
  - Sumar cada dato de la EPROM con cada dato de los registros y almacenar el resultado en la RAM. visualizar cada resultado

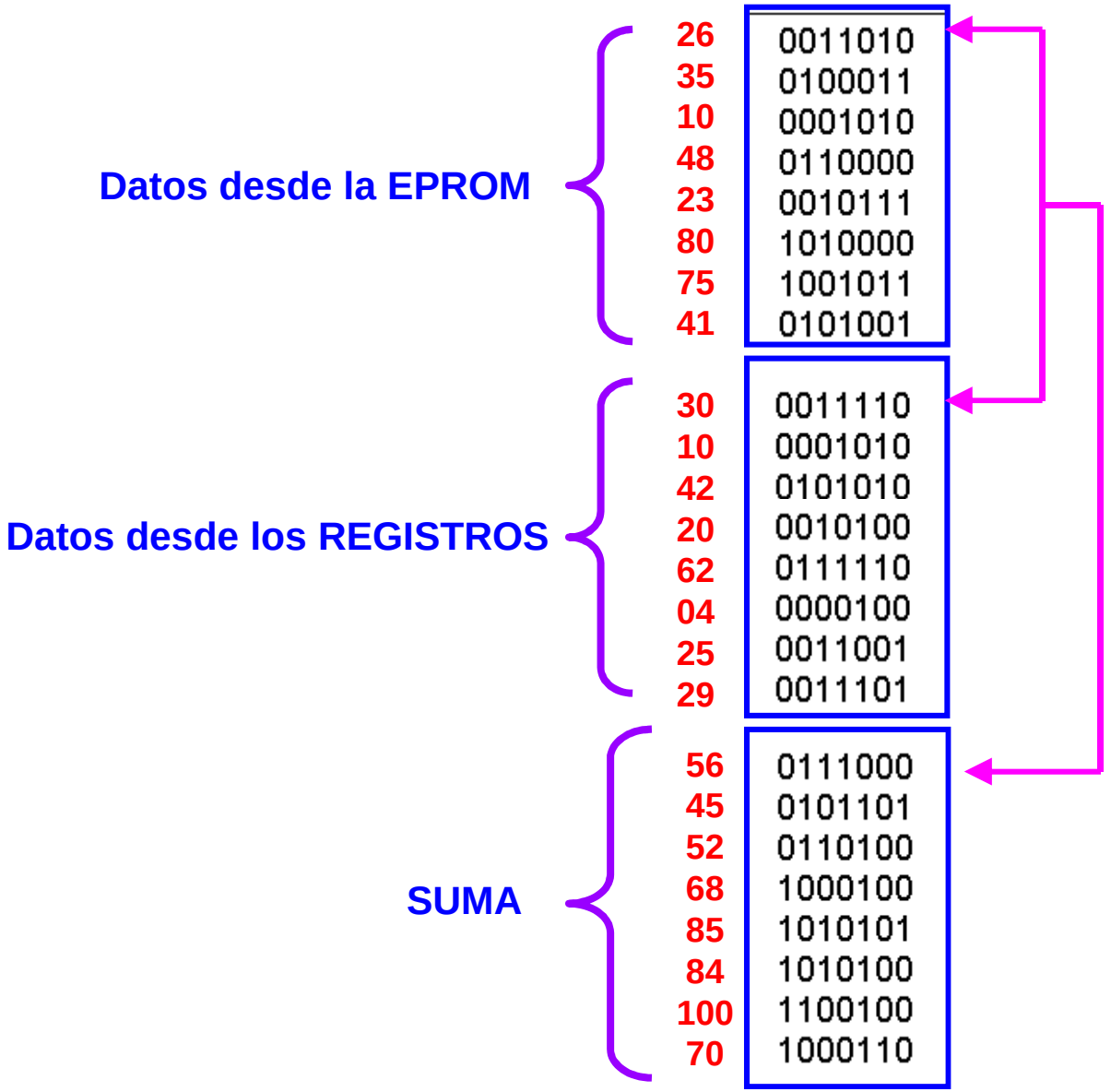
Diseño de Circuitos Controladores



Diseño de Circuitos Controladores

- Realizar un circuito de control que permita:
  - Transferir ocho datos desde la EPROM hacia la RAM y guardarlos desde la dirección 00000 hasta la 00111
  - Transferir ocho datos desde los registros hacia la RAM y guardarlos desde la dirección 01000 hasta la 01111
  - Sumar el primer dato de la EPROM con el primer dato de los registros y almacenar el resultado en la RAM en la dirección 10000, realizar esta suma con cada dato y visualizar cada resultado

Diseño de Circuitos Controladores



Entradas y Salidas para la FSM

Entradas:

Start (señal de arranque de mi máquina)

Fin8 (cuarto bit que me permite traspasar 8 datos debido a su cambio, Por ejemplo los primeros ocho datos van del 0000 al 0111 al pasar a 1000 se me activa y cambia de estado)

Reset (señal de reinicio)

Clk (señal de reloj)

Salidas:

Clear (limpiar todo)

Conteo (conteo del contador ascendente)

Bit4 (me permite guardar en la RAM de la posición 9 a la 16)

Bit5 (me permite guardar en la RAM de la posición 17 a la 24)

WRC (escribir en RC)

RRC (leer en RC)

REP (leer EPROM)

RRAM (leer RAM)

WRAM (escribir en la RAM)

RREG (leer registro)

WA (escribir en RA)

WB (escribir en RB)

Se logró hacer el control en ocho estados en los cuales se describen a continuación:

S0: Es mi estado de comienzo, hago clear en general

S1: Traspaso 8 datos que han sido guardados previamente en la EPROM a las direcciones 0000 a la 0111 de la RAM, se me activan las salidas:
Conteo, REP, WRAM

S2: Traspaso 8 datos del Registro a las direcciones 1000 a la 1111 de la RAM se me activan las salidas:
Conteo, Bit4, RREG, WRAM

S3: Cada dato que ha sido guardados en la RAM provenientes de EPROM lo traspaso al registro RA, se me activan las salidas:
RRAM, WA

S4: Cada dato que ha sido guardados en la RAM provenientes del Registro lo traspaso al registro RB, se me activan las salidas:

Bit4, RRAM, WB

S5: Se suma RA+RB y lo cargo en RC, se me activan las salidas:

WRC, Bit5

S6: Traspaso los datos que provengan de RC a las direcciones 10000 a la 10111 de la RAM, se me activan las salidas:

RC, WRAM, Bit5, Conteo

S7: Visualizo los datos sumados de la RAM, se me activan las salidas:

RRAM, Bit5, Conteo

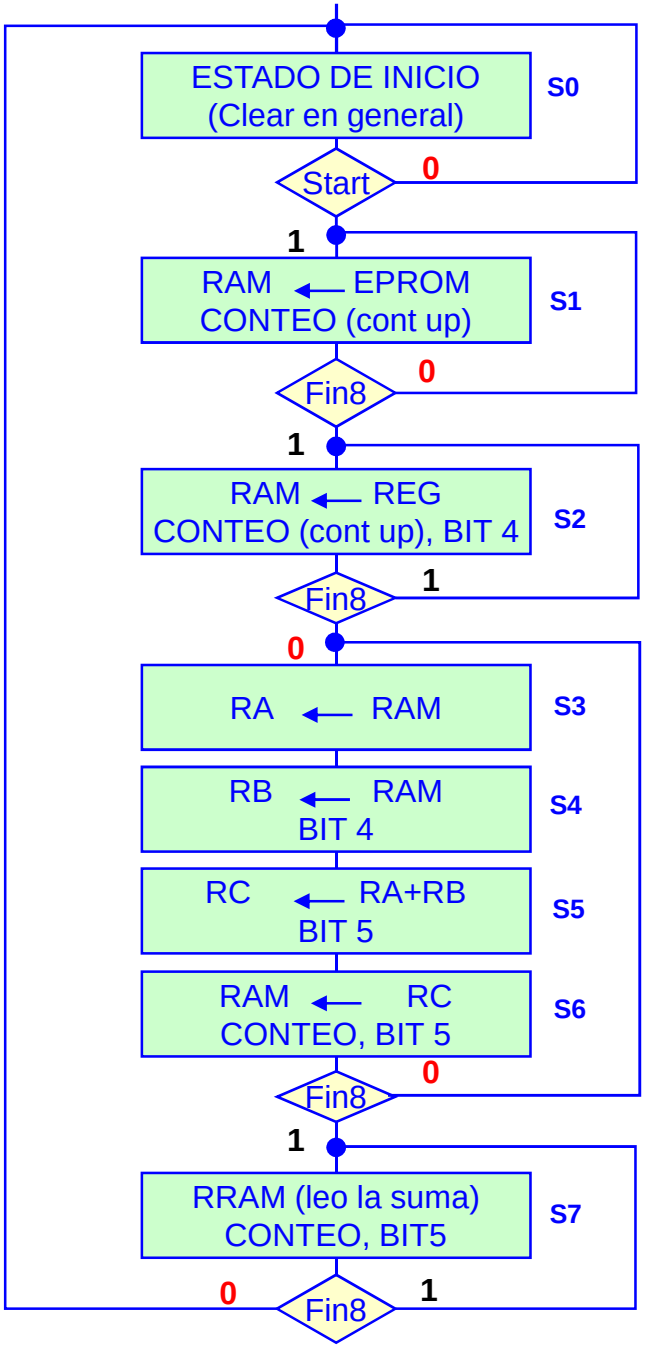
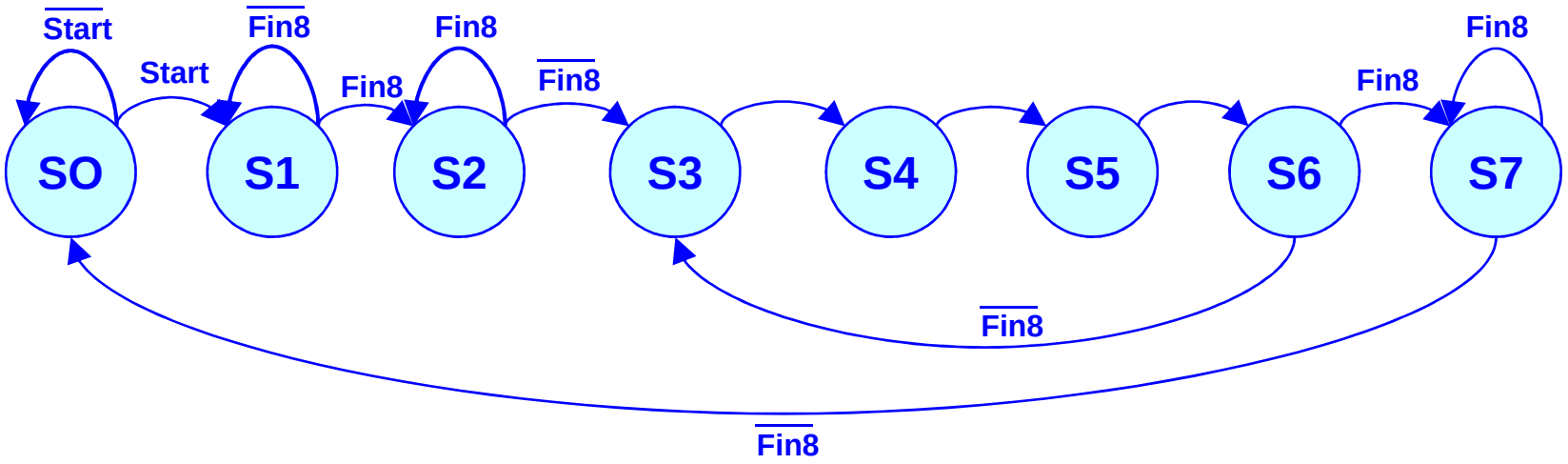

ESTADO DE INICIO
(Clear en general)
S0
Start
0
1
RAM ← EPROM
CONTEO (cont up)
S1
Fin8
0
1
RAM ← REG
CONTEO (cont up), BIT 4
S2
Fin8
1
0
RA ← RAM
S3
RB ← RAM
BIT 4
S4
RC ← RA+RB
BIT 5
S5
RAM ← RC
CONTEO, BIT 5
S6
Fin8
0
1
RRAM (leo la suma)
CONTEO, BIT5
S7
0
Fin8
1

Diagrama de Estados



Lenguaje AHDL

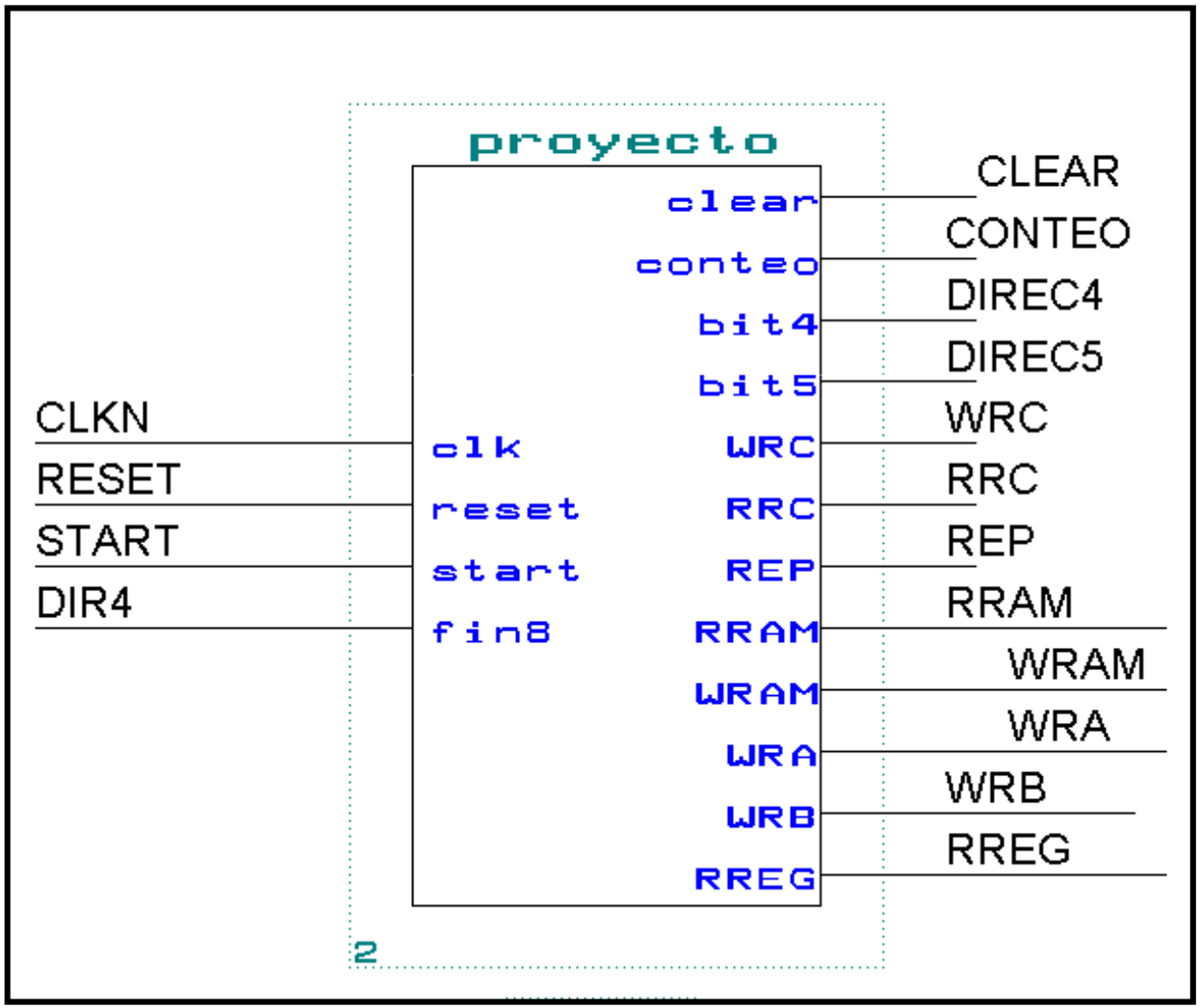
Para implementar esta máquina moore en max plus se uso el lenguaje AHDL de la siguiente manera:

```
SUBDESIGN proyecto
(
   clk   : INPUT;
   reset : INPUT;
   start, fin8 : INPUT;
   clear, conteo, bit4, bit5, WRC, RRC, REP, RRAM, WRAM, WRA,
WRB, RREG : OUTPUT;
)
VARIABLE

   %   current   current %
   %   state     output  %
ss: MACHINE OF BITS (clear, conteo, bit4, bit5, WRC, RRC, REP,
RRAM, WRAM, WRA, WRB, RREG)
```

```
WITH STATES  (
            s0    =   B"100000000000",
            s1    =   B"010000101000",
            s2    =   B"011000001001",
            s3    =   B"000000010100",
            s4    =   B"001000010010",
            s5    =   B"000110000000",
            s6    =   B"010101001000",
            s7    =   B"010100010000");
BEGIN
   ss.clk   = clk;
   ss.reset = reset;
   TABLE
   %  current  current     next  %
   %  state    input       state %
```

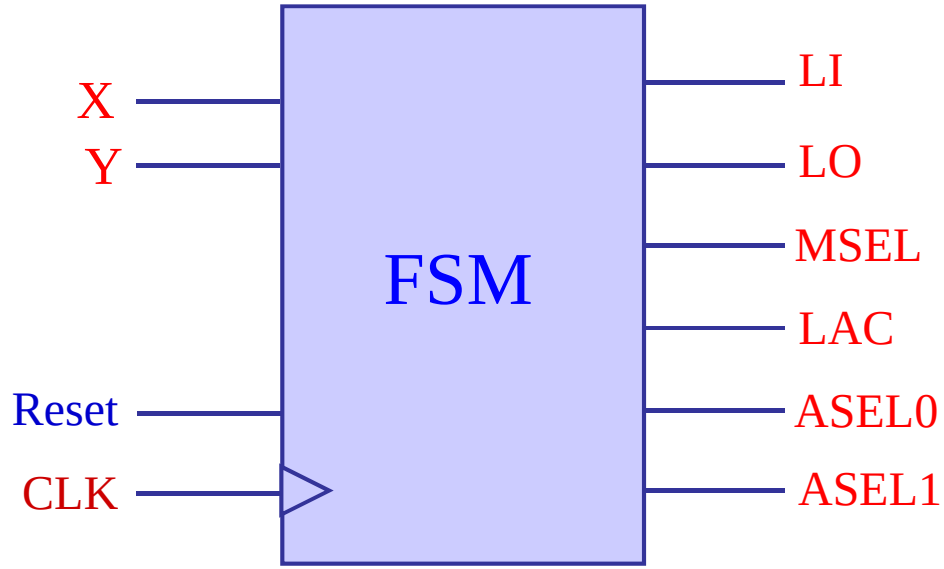
```
        ss,     start, fin8 =>  ss;
        s0,      0, x       =>  s0;
        s0,      1, x       =>  s1;
        s1,      x, 0       =>  s1;
        s1,      x, 1       =>  s2;
        s2,      x, 1       =>  s2;
        s2,      x, 0       =>  s3;
        s3,      x, x       =>  s4;
        s4,      x, x       =>  s5;
        s5,      x, x       =>  s6;
        s6,      x, 0       =>  s3;
        s6,      x, 1       =>  s7;
        s7,      x, 1       =>  s7;
        s7,      x, 0       =>  s0;
    END TABLE;

END;
```

De esta forma se diseñó el controlador para el circuito



Diseño de Circuitos Controladores

- Diagrama lógico



Datapath (Flujo de Informacion)