TDC_4_2. Diseño de un procesador (VHDL)

UCA
Universidad de Cádiz

Objetivos:

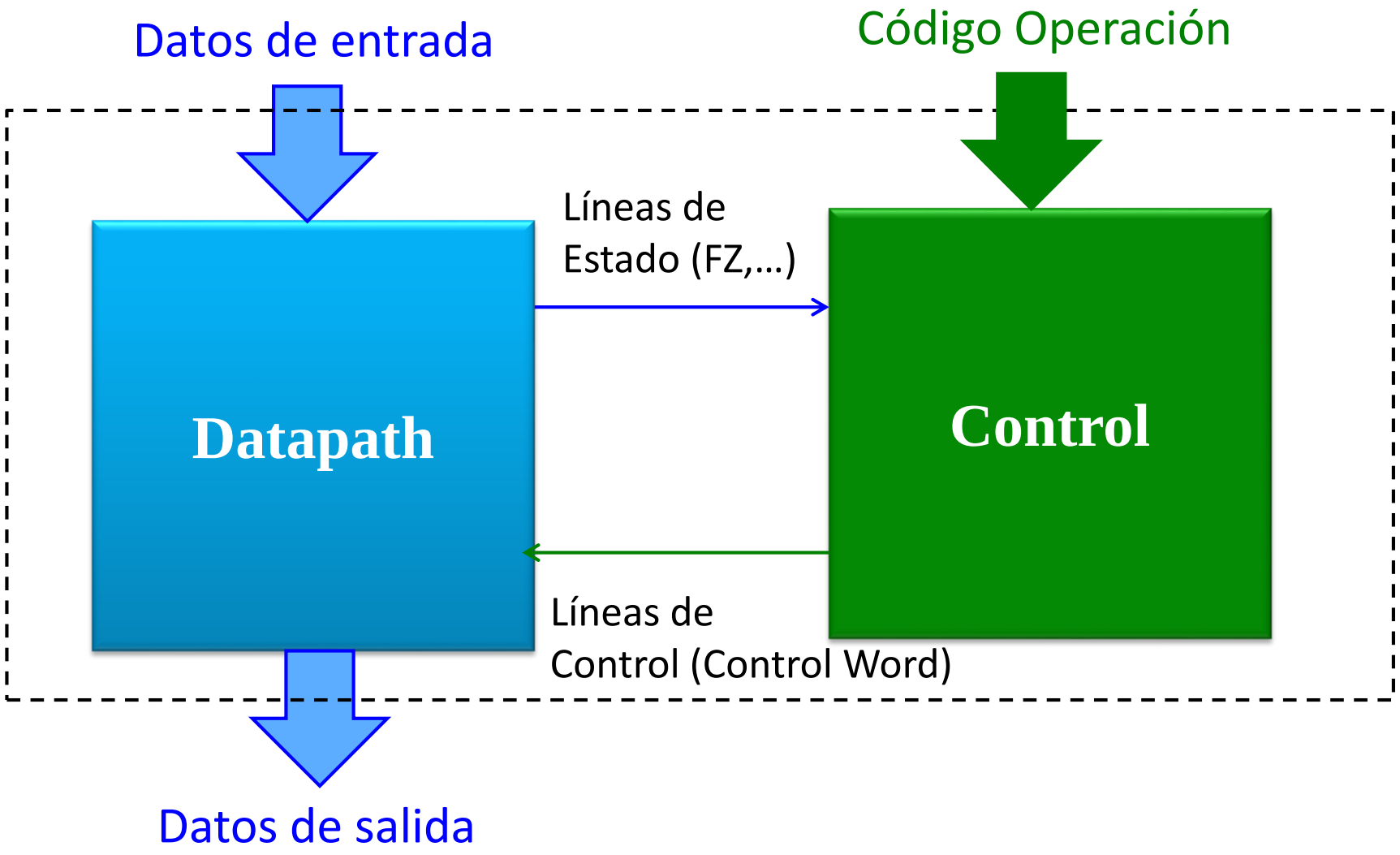
- Descripción VHDL de la unidad de control de un computador.
- Diseño de un procesador simple mediante VHDL
- Verificacion del diseño en placa desarrollo

UCA
Universidad de Cádiz

4.2.1. Elementos de un computador simple

Elementos de un computador simple

Unidad central de proceso (CPU)

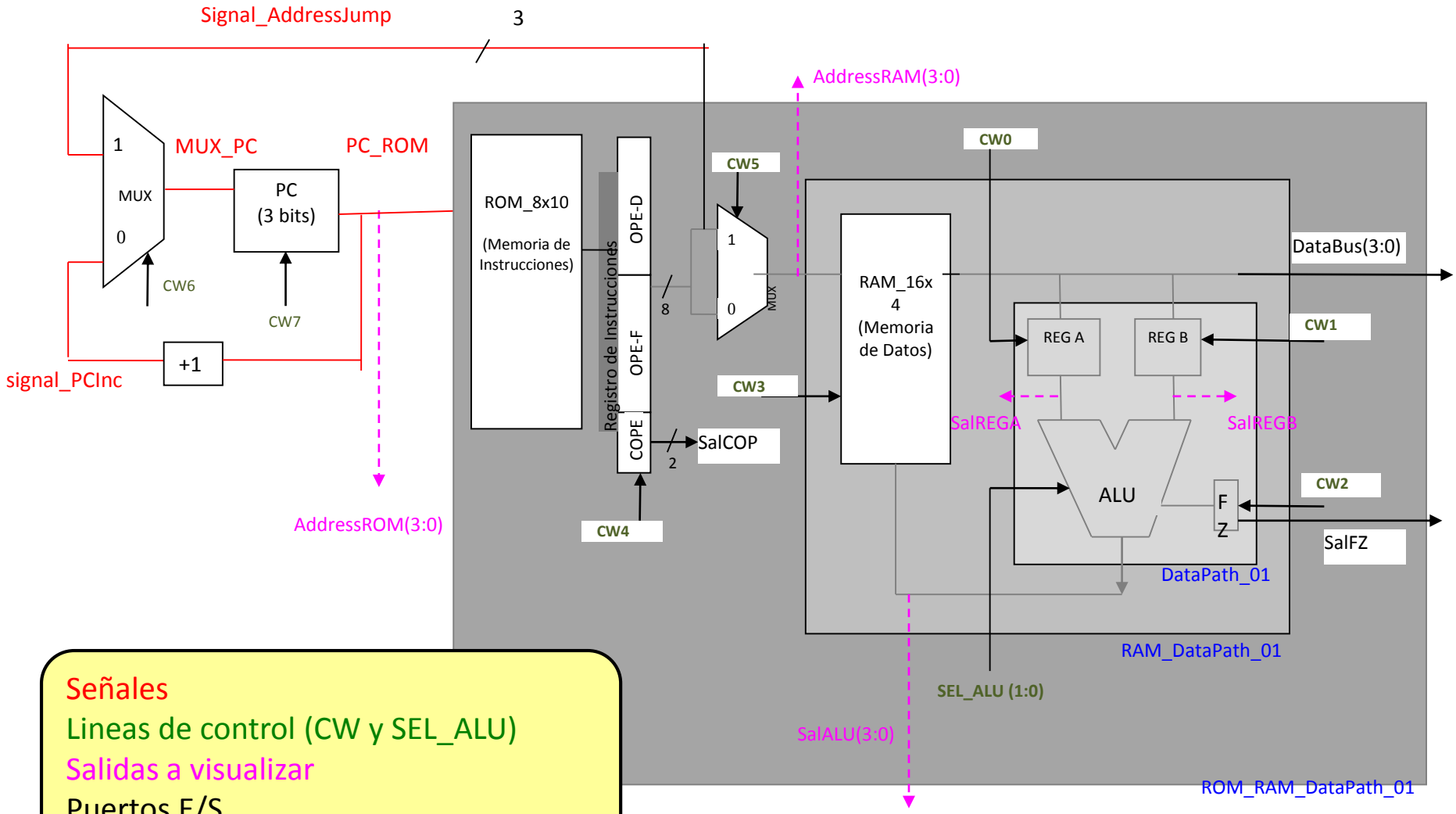


4.2.2. Unidad de control (DidaComp)

4.2.2.1 FSM para instrucción ADD (DidaComp)

Microarquitectura de DidaComp

Estructura de salto+PC+RI+ROM+RAM+DataPath_01

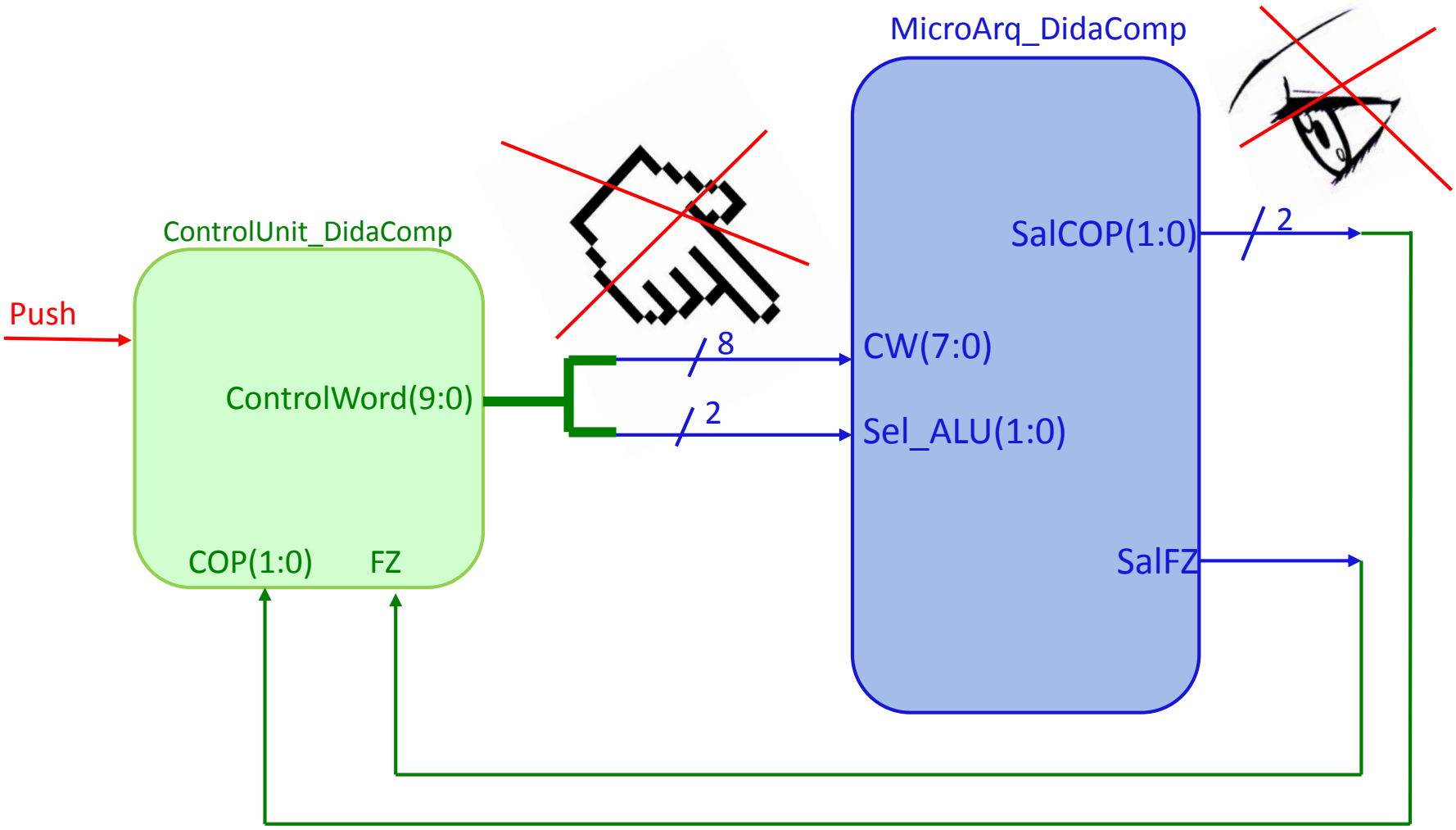


Señales
Lineas de control (CW y SEL_ALU)
Salidas a visualizar
Puertos E/S

Unidad de control (FSM)

- Es la unidad responsable de dar valores 0 ó 1 a todas las lineas de control (CW ó ControlWord ó Bus de control) para completar la ejecución de cada instrucción.
- La Unidad de control envía una secuencia de “órdenes” diferente en función de la instrucción a ejecutar (COP y FZ)
- Microprograma: secuencia de microinstrucciones que consiguen la ejecución de una instrucción.
- Firmware: Conjunto de microprogramas del ISA de un procesador

Unidad de control (DidaComp)



Unidad de control (DidaComp)

1. Microprograma de la Instrucción "Suma dos datos ADD A,B"

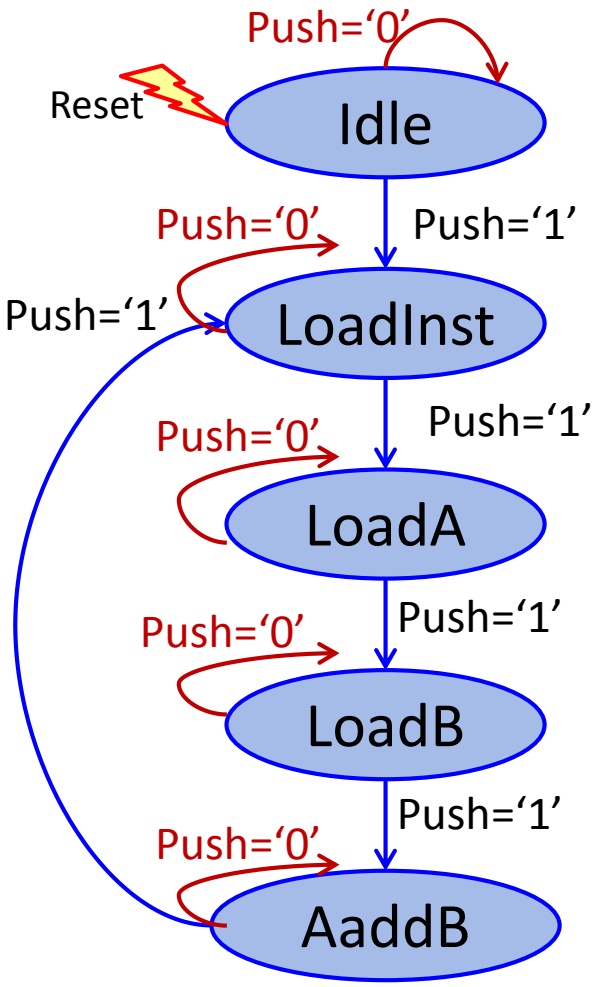
Formato tabla

Bus de control									
Acción / **ESTADO**	**SEL_ALU 1-0 (CW9-CW8)**	**CW7**	**CW6**	**CW5**	**CW4**	**CW3**	**CW2**	**CW1**	**CW0**
	Selección de operación	Actualiza PC	Origen de PC	Dir RAM	Carga RI	Escribe RAM	Carga FZ	Carga REGB	Carga REGA
Leer instrucción Incrementar PC	XX	1	0	0	1	0	0	0	0
Observar COP (Decodificación)									
Carga OPE1 desde RAM al registro A	XX	0	0	0	0	0	0	0	1
Carga OPE2 desde RAM al registro B	XX	0	0	1	0	0	0	1	0
Seleccionar suma en ALU, guardar resultado y actualizar FZ	10	0	0	1	0	1	1	0	0

Unidad de control (DidaComp)

1. Microprograma de la Instrucción : "Suma dos datos ADD A,B"

Formato grafo



Acción	ESTADO	ControlWord (salida)
Esperando	Idle	0000000000
Leer instrucción Incrementar PC	LoadInst	XX10010000
Carga OPE1 desde RAM al registro A	LoadA	XX00000001
Carga OPE2 desde RAM al registro B	LoadB	XX00100010
Selec. suma, guarda resultado y actualiza FZ	AaddB	1000101100

Suponer que ADD fuera la única instrucción del ISA y que se controla la transición entre estados con un pulsador externo, PUSH.

Unidad de control (DidaComp)

Proyecto31. Unidad de control: Automatizar la ejecución de la instrucción ADD.

```
architecture Behavioral of ControlUnit_ADD is
-------------------------------------------------------
-- DEFINITION of STATES
-------------------------------------------------------
type States_FSM is (Idle,LoadInst,LoadA,LoadB, AaddB);
signal Next_State: States_FSM;

-------------------------------------------------------
-- DEFINITION of the OUTPUTS for each STATE
-------------------------------------------------------
constant Outputs_Idle:     std_logic_Vector(9 downto 0):="0000000000";
constant Outputs_LoadInst: std_logic_Vector(9 downto 0):="0010010000";
constant Outputs_LoadA:    std_logic_Vector(9 downto 0):="0000000001";
constant Outputs_LoadB:    std_logic_Vector(9 downto 0):="0000100010";
constant Outputs_AaddB:    std_logic_Vector(9 downto 0):="1000101100";
-------------------------------------------------------------------
```

Unidad de control (DidaComp)

Proyecto31. Unidad de control: Automatizar la ejecución de la instrucción ADD.

Primer proceso → Secuencial

```
Begin
 process (CLK,RESET)
 begin
 if RESET = '1' then
        Next_State <= Idle;          -- INICIO si RESET
 elsif rising_edge(CLK)then
        case Next_State is

-- State "Idle" -
        when Idle=>
          if (Push = '1') then
             Next_State<= LoadInst;
          end if;
-- State "LoadInst"
when LoadInst=>
    if (Push = '1') then
       Next_State <= LoadA;
    end if;
-- State "LoadA"
when LoadA=>
    if (Push = '1') then
       Next_State <= LoadB;
    end if;
```

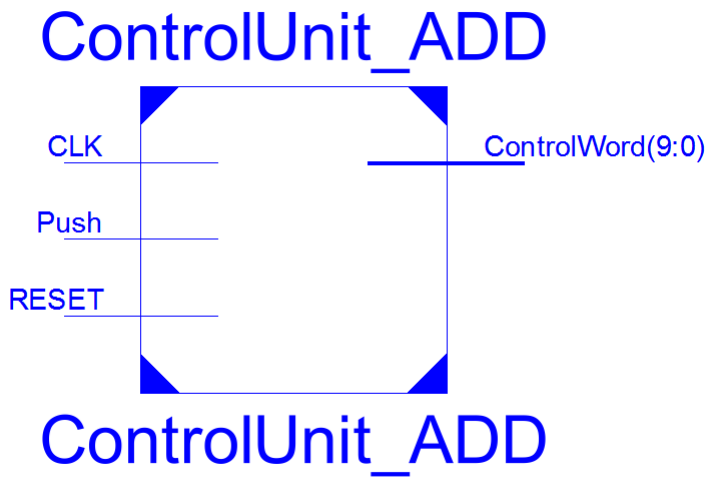
Unidad de control (DidaComp)

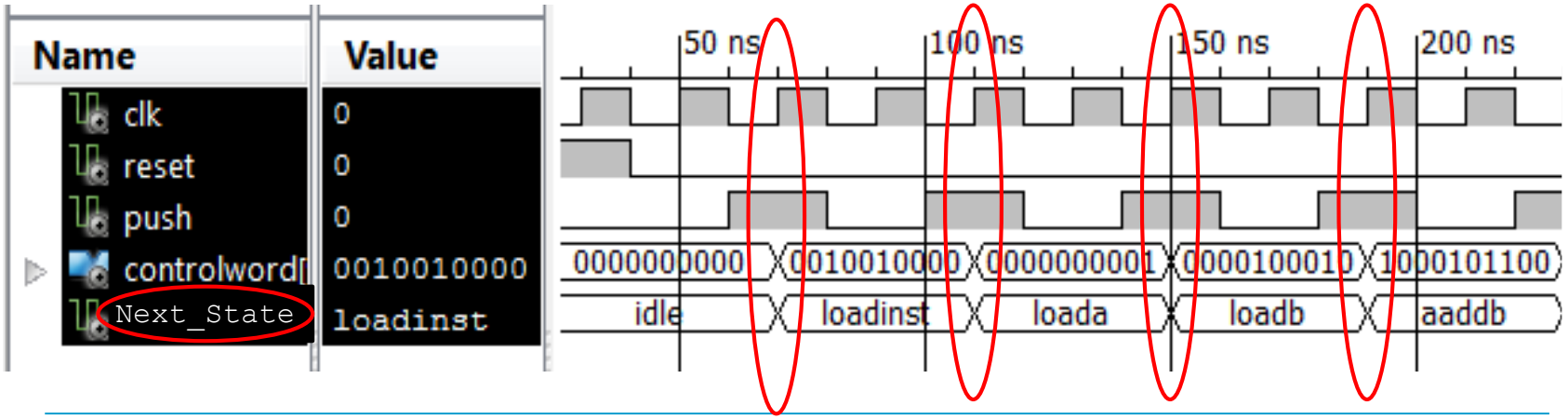
```
-- State "LoadB"
 when LoadB =>
    if (Push = '1') then
       Next_State <= AaddB;
    end if;
-- State "AaddB"
 when AaddB =>
    if (Push = '1') then
         Next_State <= LoadInst;
    end if;
    when others =>
      Next_State <= Idle;
 end case;
end if;
end process;

with Next_State select
 ControlWord <= Outputs_Idle    when Idle,
                Outputs_LoadInst when LoadInst,
                Outputs_LoadA    when LoadA,
                Outputs_LoadB    when LoadB,
                Outputs_AaddB    when AaddB,
                Outputs_Idle     when others;
```

Unidad de control (DidaComp)

Unidad de control: Automatizar la ejecución de una instrucción

```
RESET<='1'; wait for 40 ns;

RESET<='0'; wait for 40 ns;

PUSH<='1'; wait for 20 ns;
PUSH<='0'; wait for 20 ns;        x5

PUSH<='1'; wait for 20 ns;
PUSH<='0'; wait for 20 ns;

…

end process;
```

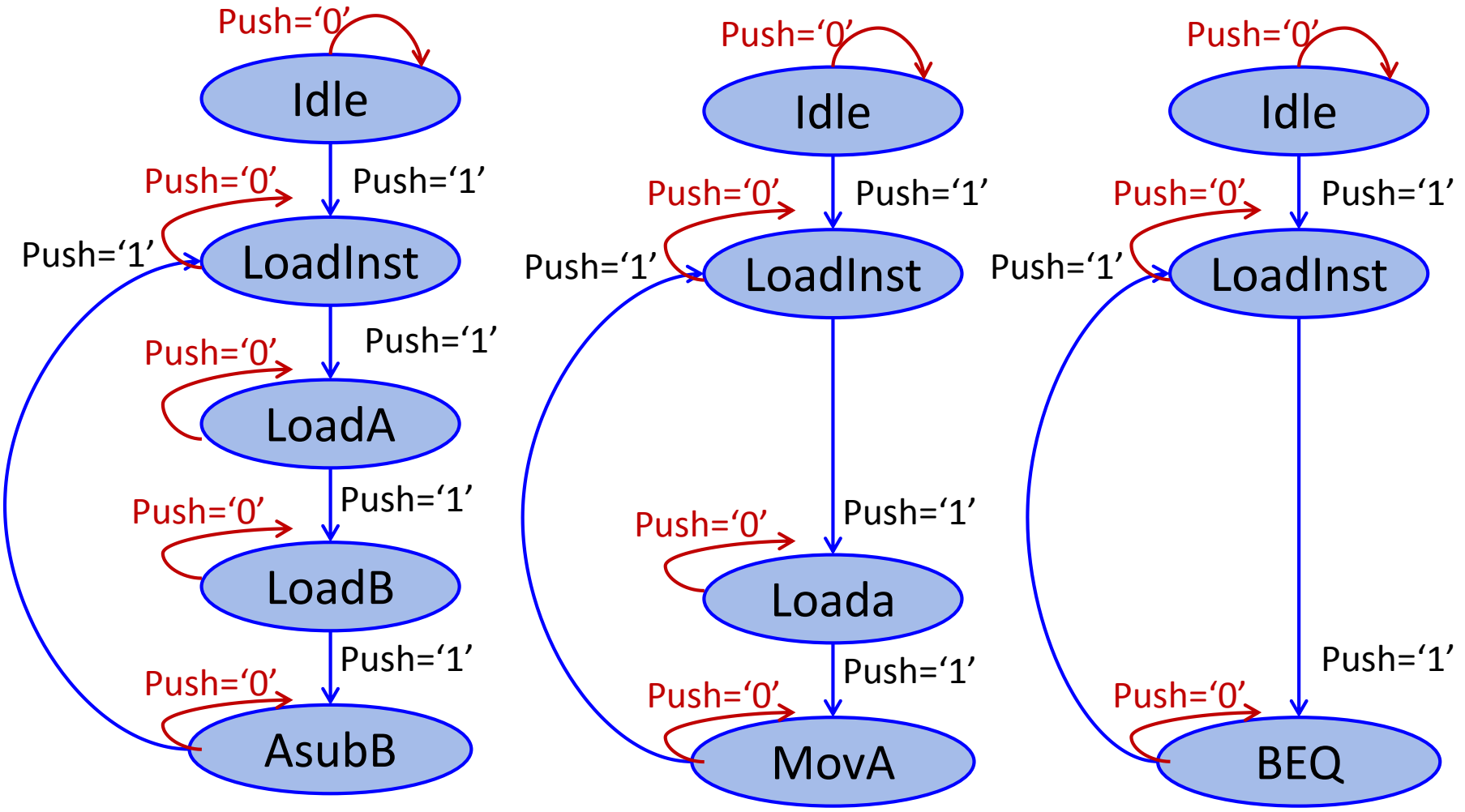




4.2.2.2 FSM Unidad de control completa (DidaComp)

Unidad de control COMPLETA (DidaComp)

Grafos del resto de instrucciones

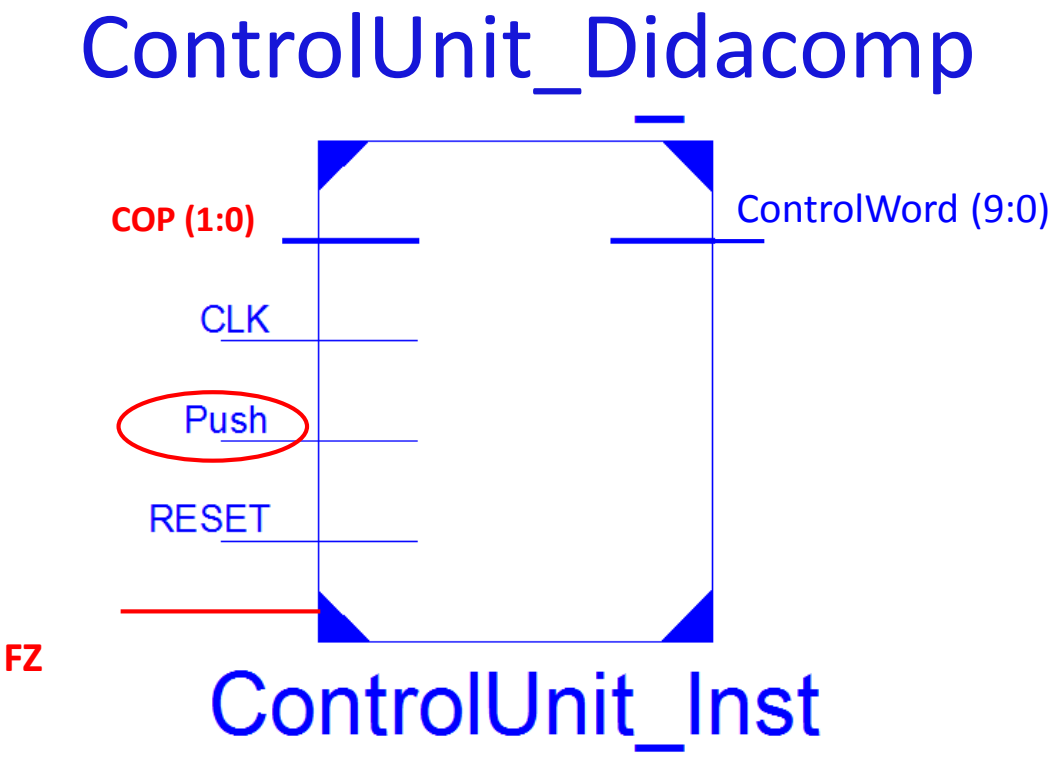


Unidad de control COMPLETA (DidaComp)

Unidad de control (FSM)

Para todo el juego de instrucciones (ISA)

Entradas		Salida
COP	**SalFZ**	**Instrucción**
00	X	MOV F,D
01	1	BEQ dir
01	0	Siguiente
10	X	ADD F,D
11	X	SUB F,D

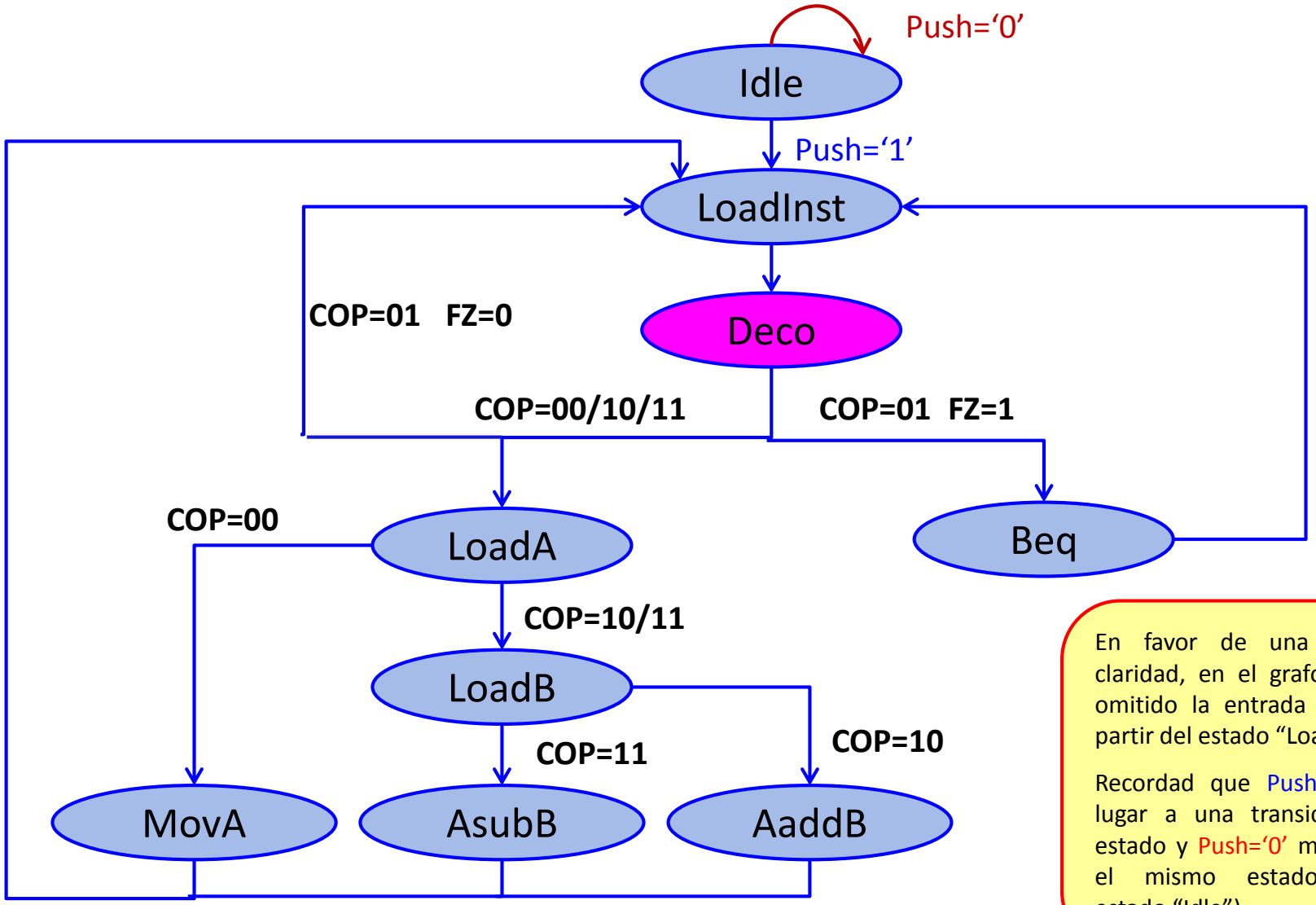


Unidad de control COMPLETA (DidaComp)

Para todo el juego de instrucciones

- Analizar todos los estados de todas las intrucciones
- Simplificar estados comunes entre las instrucciones
- Incluir un estado dónde se determine, según COP, qué instrucción ejecutar. → Decodificación
- Construir un solo grafo con todas las instrucciones

Unidad de control COMPLETA (DidaComp)



En favor de una mayor claridad, en el grafo se ha omitido la entrada **Push** a partir del estado "LoadInst".

Recordad que Push='1' da lugar a una transición de estado y Push='0' mantiene el mismo estado. (Ver estado "Idle").

Unidad de control COMPLETA (DidaComp)

Salida asociada a cada estado (ControlWord)

Estado	CW9-CW8	CW7	CW6	CW5	CW4	CW3	CW2	CW1	CW0
	SelALU	IncPc	Origen PC	Sel. Dir, operandos	Carga RI	R/W RAM	Carga FZ	Carga B	Carga A
Idle	00	0	0	0	0	0	0	0	0
LoadInst	00	1	0	0	1	0	0	0	0
Deco	00	0	0	0	0	0	0	0	0
LoadA	00	0	0	0	0	0	0	0	1
LoadB	00	0	0	1	0	0	0	1	0
AaddB	10	0	0	1	0	1	1	0	0
AsubB	11	0	0	1	0	1	1	0	0
MovA	00	0	0	1	0	1	1	0	0
Beq	01	1	1	1	0	0	0	0	0