TDC_4_2. Diseño de un procesador (VHDL)

UCA
Universidad de Cádiz

Objetivos:

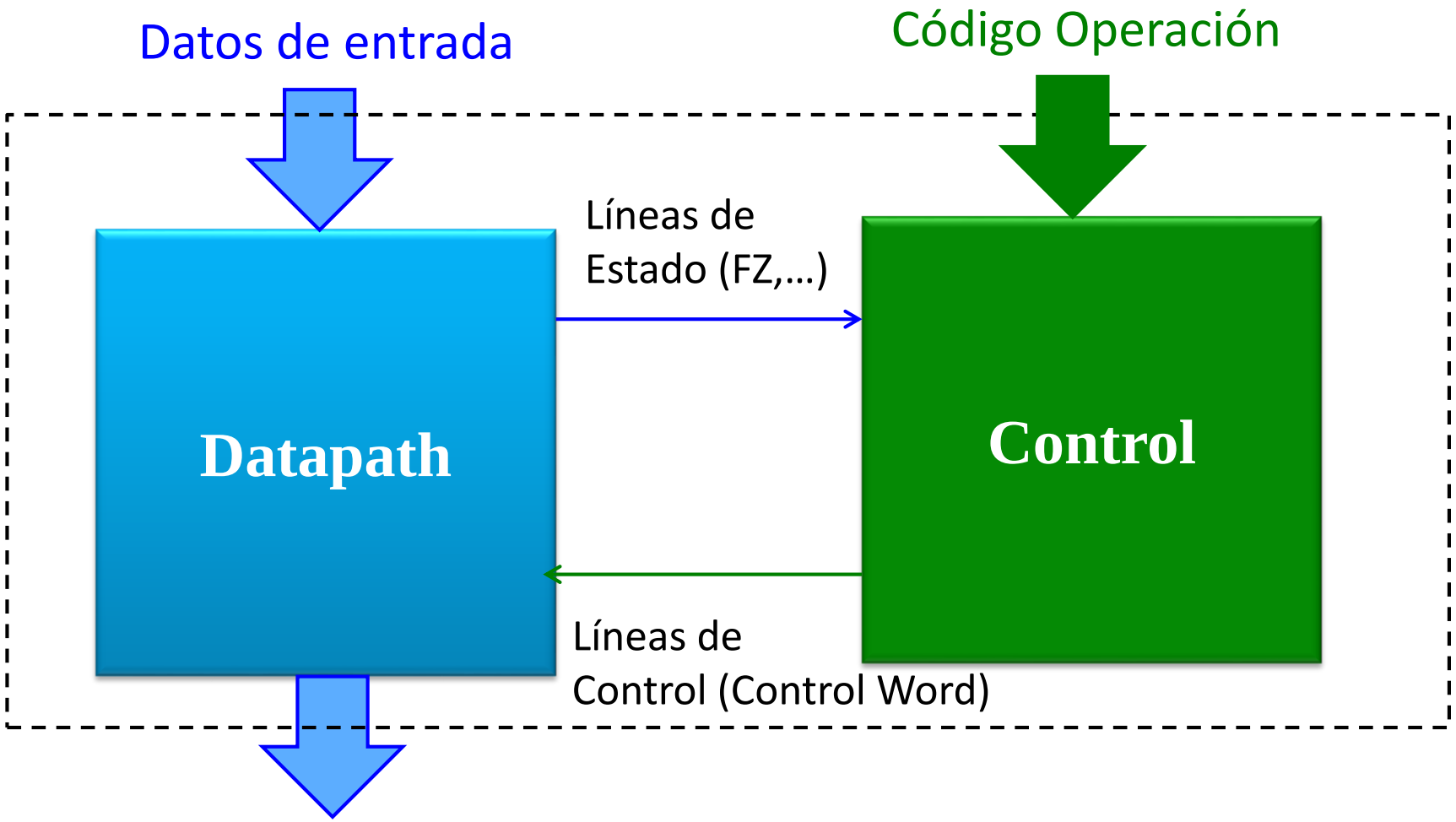
- Descripción VHDL de la unidad de control de un computador.
- Diseño de un procesador simple mediante VHDL
- Verificacion del diseño en placa desarrollo

UCA
Universidad de Cádiz

4.2.1. Elementos de un computador simple

Elementos de un computador simple

Unidad central de proceso (CPU)

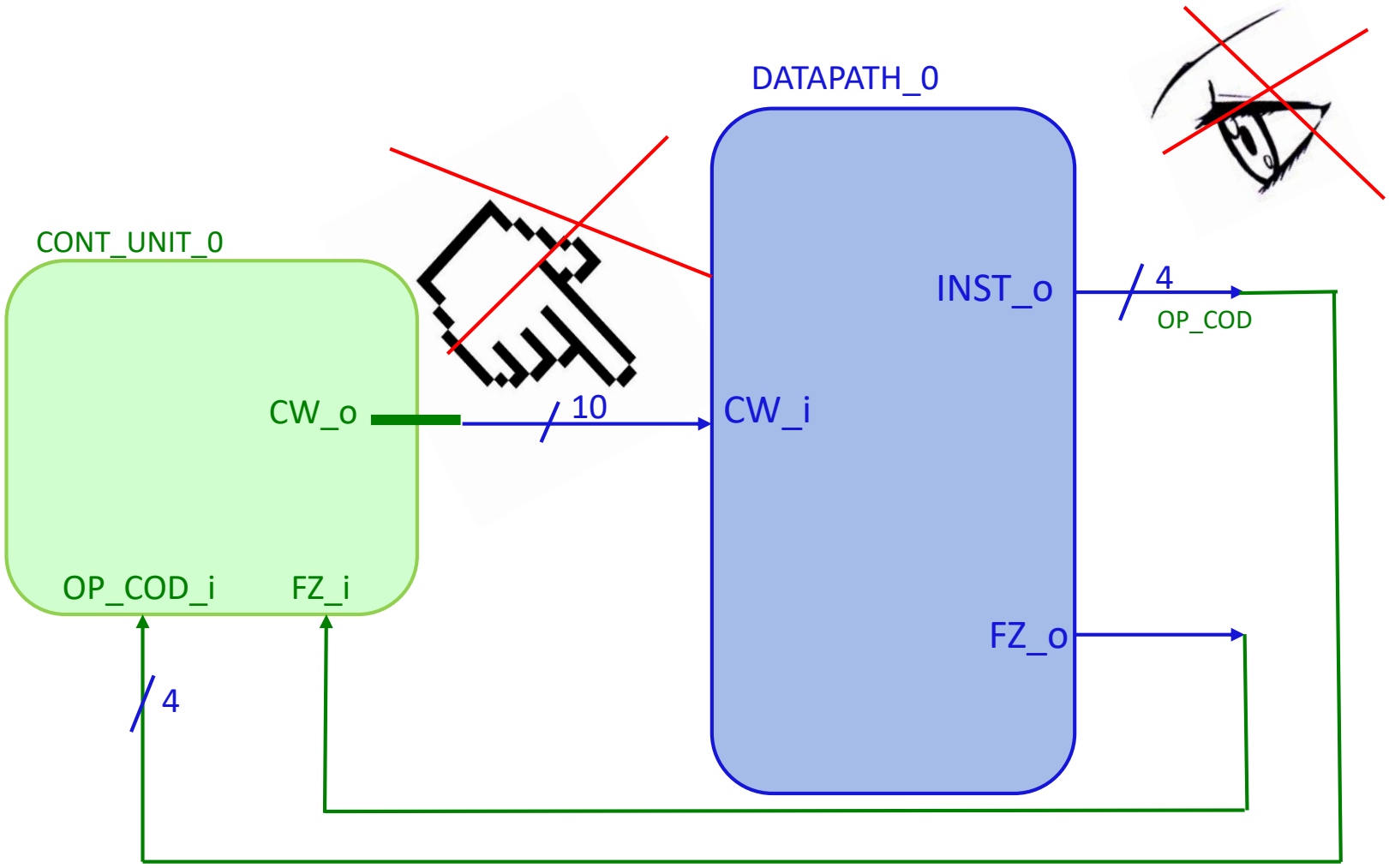


4.2.2. Diseño de la Unidad de control (DidaComp)

Unidad de control

- 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). Esto se consigue diseñándola como una **máquina de estados** (FSM).
- 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)

- Tras diseñar la micro-arquitectura, se han extraído una serie de tablas que recopilan las operaciones que debe realizar la unidad de control para ejecutar cada instrucción del juego de instrucciones (ISA).
- Cada una de esas tablas es el microprograma de cada una de las instrucciones.
- El juego de instrucciones de Didacomp se ha diseñado con cinco instrucciones en total, por lo tanto habrá cinco microprogramas.

Instrucción	OP_COD	Función de la instrucción
MOV addr1, addr2	0000	[addr1] → addr2
INC addr1	0001	[addr1] + 1 → addr1
ADD addr1,addr2	0010	[addr1] + [addr2] → addr2
SUB addr1,addr2	0011	[addr1] - [addr2] → addr2
BEZ addr2	0100	PC ← addr2 (FZ=1)

Unidad de control (DidaComp)

- La unidad de control consigue efectuar las operaciones necesarias mediante las líneas de control (CW).

- Los cinco microprogramas tendrán operaciones comunes:
  - Búsqueda de la instrucción
  - Incremento del contador de programa
  - Decodificación de la instrucción
  - Lectura de los operandos A y B (según instrucción)

Unidad de control (DidaComp)

P.e. → Microprograma de la Instrucción "Suma dos datos ADD addr1,addr2"

CLK	Bus de control									
	Microinstrucción	**CW(9-8)**	**CW(7)**	**CW(6)**	**CW(5)**	**CW(4)**	**CW(3)**	**CW(2)**	**CW(1)**	**CW(0)**
		ALU OPE	Sel. Fuente Addr. Inst	Incrementa reg PC	Sel. Addr. Ope.	Lee y guarda Inst (RI)	!R/W RAM	Carga REG_A	Carga REG_B	Carga FZ
1	Sel. Addr. ROM. Guarda INST en RI. Incrementa reg PC	XX	0	1	0	1	0	0	0	0
2	DECO/Establece dirección RAM del operando A	XX	0	0	0	0	0	0	0	0
3	Carga OPEA desde RAM al registro A	XX	0	0	0	0	0	1	0	0
4	Establecer dirección RAM del operando B	XX	0	0	1	0	0	0	0	0
5	Carga OPEB desde RAM al registro B	XX	0	0	1	0	0	0	1	0
6	Seleccionar suma en ALU, escribe resultado en RAM, actualiza FZ	10	0	0	1	0	1	0	0	1

Unidad de control (DidaComp)

P.e.: Microprograma de la Instrucción "Salto condicional BEZ"

Si FZ=1

	Acción ESTADO	CW_i (9:8) OpeALU	CW_i (7) Fuente PC	CW_i (6) Actualiza PC	CW_i (5) Dir RAM	CW_i (4) Carga RI	CW_i (3) Escribe RAM	CW_i (2) Carga REGA	CW_i (1) Carga REGB	CW_i (0) Carga FZ
		Bus de control								
LOAD	C1: Leer instrucción Incrementar PC	XX	0	1	0	1	0	0	0	0
DECO	C2: Decodificación COP	00	0	0	0	0	0	0	0	0
BRANCH	C3: Selecciona y almacena en PC la dirección salto (a leer en ROM)	XX	1	1	0	0	0	0	0	0
BR_INST	**C4: Espera que la Instrucción salga de la ROM .**	XX	1	0	0	0	0	0	0	0

Unidad de control (DidaComp)

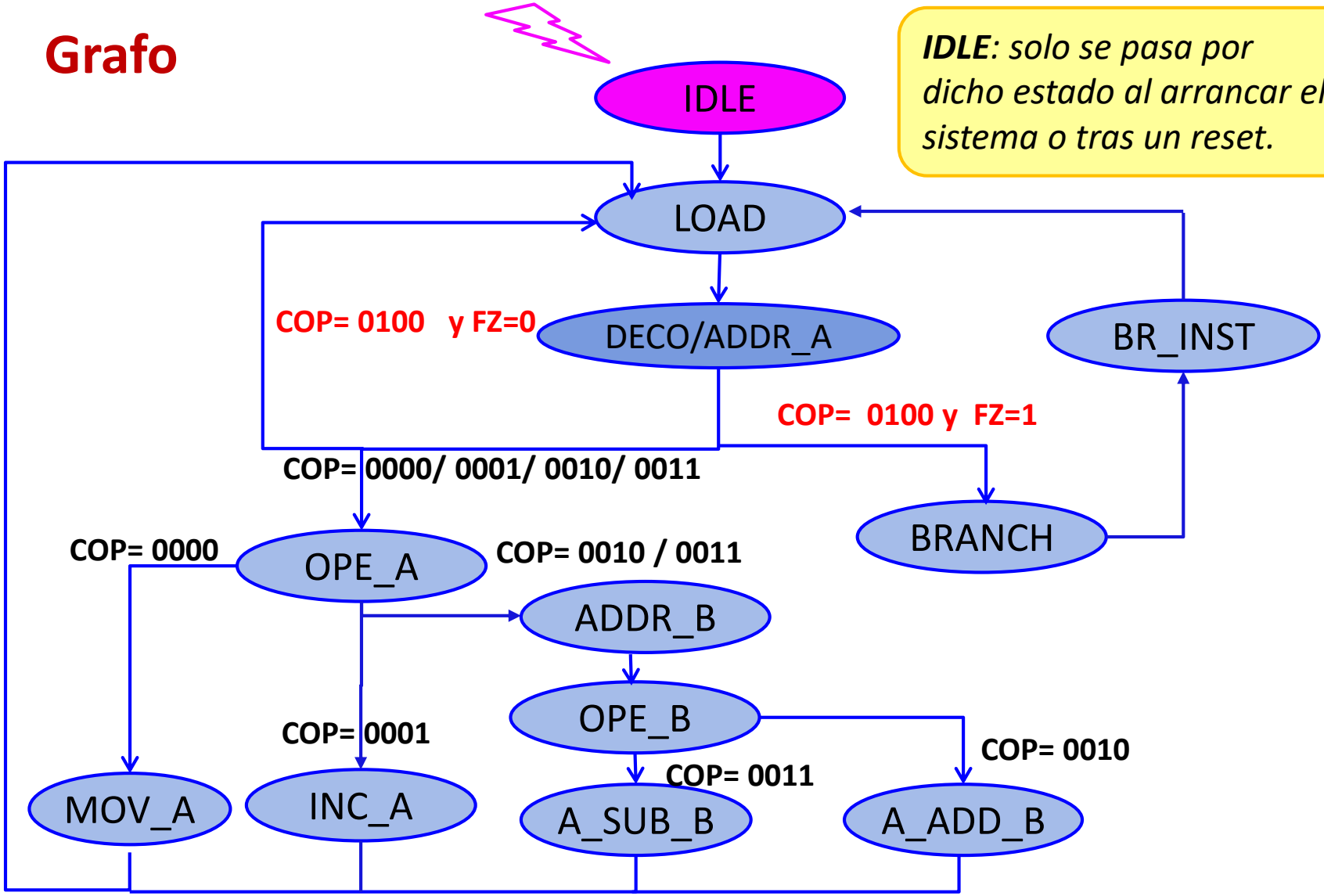
- La unidad de control es una FSM y por tanto su funcionamiento se describe mediante un grafo.
  - Cada estado representa la/las acciones a realizar por la uA en cada ciclo de reloj.
  - Cada estado lleva asociada una salida que se corresponde con un conjunto de valores CW.

CLK	Bus de control									
	Microinstrucción	CW(9-8)	CW(7)	CW(6)	CW(5)	CW(4)	CW(3)	CW(2)	CW(1)	CW(0)
		ALU OPE	Sel. Fuente Addr. Inst	Incrementa reg PC	Sel. Addr. Ope.	Lee y guarda Inst (RI)	!R/W RAM	Carga REG_A	Carga REG_B	Carga FZ
1	Sel. Addr. ROM. Guarda INST en RI. Incrementa reg PC	XX	0	1	0	1	0	0	0	0

  - Las transiciones tienen lugar en cada nuevo ciclo de reloj y dependerán del valor del código de operación de la instrucción en curso.

FSM de la Unidad de control (DidaComp)

Grafo



***IDLE**: solo se pasa por dicho estado al arrancar el sistema o tras un reset.*

Unidad de control COMPLETA (DidaComp)

Salida asociada a cada estado (ControlWord)

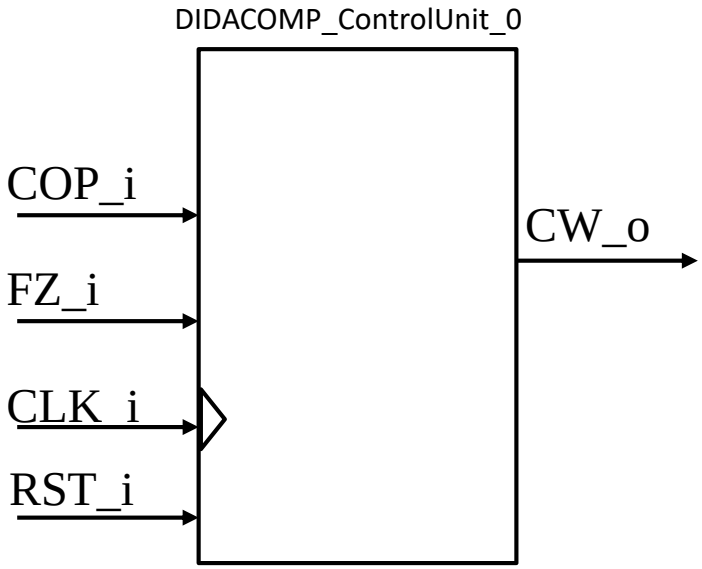
Estado	CW9-CW8	CW7	CW6	CW5	CW4	CW3	CW2	CW1	CW0
	ALU	ADDR_ ROM	Actualiza PC	ADDR_ RAM	Carga RI	R/W RAM	Carga REG A	Carga REG B	Carga FZ
IDLE	XX	0	0	0	0	0	0	0	0
LOAD	XX	0	1	0	1	0	0	0	0
DECO	XX	0	0	0	0	0	0	0	0
OPE_A	XX	0	0	0	0	0	1	0	0
OPE_B	XX	0	0	1	0	0	0	1	0
A_ADD_B	10	0	0	1	0	1	0	0	1
A_SUB_B	11	0	0	1	0	1	0	0	1
MOV_A	00	0	0	1	0	1	0	0	1
BRANCH	XX	1	1	0	0	0	0	0	0
BR_INST	XX	1	0	0	0	0	0	0	0
INC_A	01	0	0	1	0	1	0	0	1
ADDR_B	XX	0	0	1	0	0	0	0	0

4.2.3. Descripción VHDL de la Unidad de control (DidaComp)

Unidad de control COMPLETA (DidaComp)

Unidad de control (FSM) - Entidad

Entradas		Salida (Microprograma de…)
COP_i	**FZ_i**	**CW_o**
0000	X	MOV
0001	X	INC
0010	X	ADD
0011	X	SUB
0100	0	Siguiente instruc.
0100	1	BEZ



Unidad de control (DidaComp)

Project 29: CONTROL_UNIT

File: "ControlUnit_0"

```
architecture Behavioral of ControlUnit_0 is
-------------------------------------------
-- DEFINITION of STATES
-------------------------------------------------------
type STATES_FSM is (IDLE,LOAD,DECO,OPE_A,ADDR_B, OPE_B, BRANCH,
BR_INST, MOV_A, INC_A, A_ADD_B, A_SUB_B);
signal NEXT_STATE: STATES_FSM;
-------------------------------------------------------
-- DEFINITION of the OUTPUTS for each STATE
-------------------------------------------------------
```

Tabla página 14

```
constant  OUTPUT_IDLE:   std_logic_Vector(CW_WIDTH-1 downto 0):="0000000000";
constant  OUTPUT_LOAD:   std_logic_Vector(CW_WIDTH-1 downto 0):="0001010000";
constant  OUTPUT_DECO:   std_logic_Vector(CW_WIDTH-1 downto 0):="0000000000";
constant  OUTPUT_OPE_A:  std_logic_Vector(CW_WIDTH-1 downto 0):="0000000100";
constant  OUTPUT_ADDR_B: std_logic_Vector(CW_WIDTH-1 downto 0):="0000100000";
constant  OUTPUT_OPE_B:  std_logic_Vector(CW_WIDTH-1 downto 0):="0000100010";
constant  OUTPUT_MOV_A:  std_logic_Vector(CW_WIDTH-1 downto 0):="0000101001";
constant  OUTPUT_INC_A:  std_logic_Vector(CW_WIDTH-1 downto 0):="0100101001";
constant  OUTPUT_A_ADD_B:std_logic_Vector(CW_WIDTH-1 downto 0):="1000101001";
constant  OUTPUT_A_SUB_B:std_logic_Vector(CW_WIDTH-1 downto 0):="1100101001";
constant  OUTPUT_BRANCH: std_logic_Vector(CW_WIDTH-1 downto 0):="0011000000";
constant  OUTPUT_BR_INST:std_logic_Vector(CW_WIDTH-1 downto 0):="0010000000";
```

Unidad de control (DidaComp)

```
elsif rising_edge(CLK)then
 case NEXT_STATE is
-- State "IDLE" -
      when IDLE=>
            NEXT_STATE <= LOAD;
-- State "LOAD"
      when LOAD =>
            NEXT_STATE <= DECO;
-- State "DECO"
      when DECO =>
       case(COP_i) is
       -- MOV, ADD or SUB instructions
          when "0000" | "0001"| "0010" | "0011" =>
          -- State "OPE_A"
             NEXT_STATE <= OPE_A;
              -- AGREGAR LOS QUE FALTAN
-- State "OPE_A"
      when OPE_A =>
            NEXT_STATE <= OPE_B;
-- State "ADDR_B"
      when ADDR_B =>
            NEXT_STATE <= LOAD;
-- State "OPE_B"
      when OPE_B =>
            NEXT_STATE <= A_ADD_B;
 -- AGREGAR LOS QUE FALTAN
```

Primer proceso → Secuencial

¿Sería posible una descripción más corta del estado DECO?

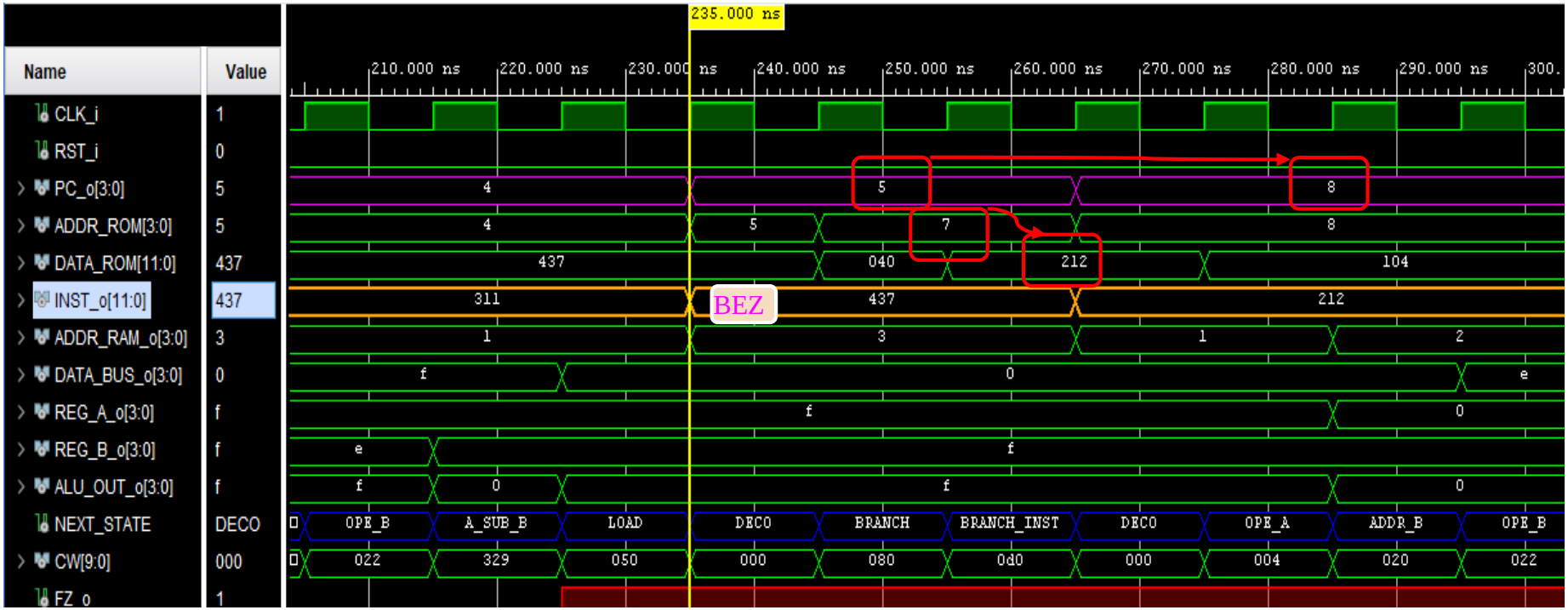
Unidad de control (DidaComp)

```
    when others =>
      Next_State <= IDLE;
 end case;
end if;
end process;
-------------------------------------------------------------
with NEXT_STATE select
 CW_o <= OUTPUT_IDLE          when IDLE,
         OUTPUT_LOAD          when LOAD,
         OUTPUT_OPE_A         when OPE_A,
         OUTPUT_ADDR_B        when ADDR_B,
         OUTPUT_OPE_B         when OPE_B,
         OUTPUT_A_ADD_B       when A_ADD_B,
         -- AGREGAR LOS QUE FALTAN
         OUTPUT_IDLE          when others;
```

Segundo proceso → Concurrentes

Unidad de control (DidaComp)

Unidad de control (FSM): Estructura de salto, instrucción BEZ



PC almacena la dirección de la instrucción siguiente.

La instrucción indicada por la dirección de salto se ejecuta.