Departament de Disseny i Programació de Sistemes Electrònics
UNIVERSITAT POLITÈCNICA DE CATALUNYA

Arquitectura de computadors

Pràctica 4

Disseny de la unitat de control

Antoni Escobet

PRÀCTICA 4 – Disseny de la unitat de control

1. Objectius

En aquesta practica es pretén:

- Repassar els conceptes relacionats amb la unitat de control d'un sistema microprocessador.
- Realitzar el disseny d'un sistema seqüencial mitjançant un autòmat de Moore en VHDL.
- Dissenyar la unitat de control del camí de dades.
- Amplia els coneixements sobre el llenguatge VHDL i l'edició amb QUARTUS II.

2. Material

L'únic material necessari per la realització d'aquesta pràctica és el paquet de programari d'ALTERA, el Quartus II web edition instal·lat als ordinadors del laboratori, i que us podeu descarregar gratuïtament de la pàgina web d'Altera (http://www.altera.com).

Recordeu baixar-vos el programa de simulació: ModelSim-Altera Edition per la versió de Quartus II que tingueu instal·lada.

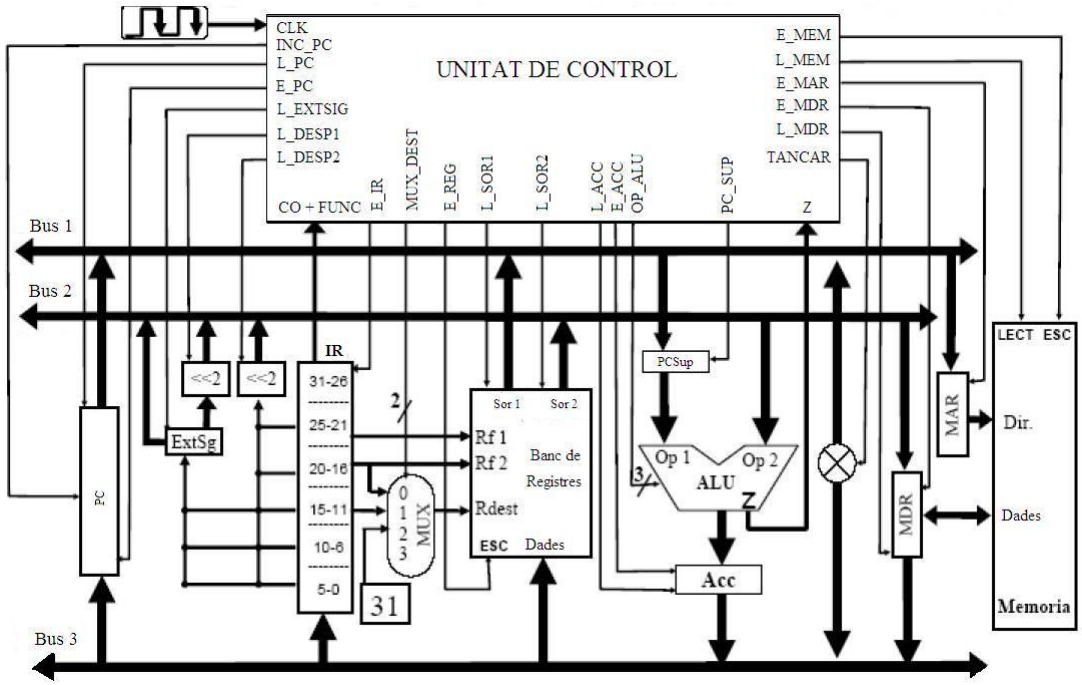
3. Problema proposat

En aquesta pràctica l'alumne ha de realitzar el disseny de la Unitat de Control (UC) pel processador que s'ha estudiat a classe. Com s'ha vist, la UC és la part del processador que dirigeix i coordina totes les operacions que realitza. La UC és un sistema seqüencial que dissenyarem com un Autòmat de Moore.

Convé recordar el joc d'instruccions del processador a dissenyar:

Operacions aritmètiques - lògiques	
ADD	Suma aritmètica
SUB	Resta
AND	Producte lògic
OR	Suma lògica
SLT	Comparació: menor que
ADDI	Suma aritmètica immediata
ANDI	Producte lògic immediat
OR	Suma lògica immediata
Operacions de transferència amb memòria	
LW	Càrrega paraula de memòria
SW	Emmagatzemament de paraula a memòria
Operacions de ruptura de seqüència	
BEQ	bot condicional (si igual)
BNE	Bot condicional (si distint)
J	Bot incondicional
JAL	Bot subrutina
JR	Bot indirecte

En el següent esquema es recorda la ruta de dades que ha de controlar la unitat de control:



Com es pot veure a l'esquema, la UC rep les següents entrades:

- El camp “codi d'operació” del registre d'instruccions (IR) (bits 31..26)
- El camp “funció” del IR (bits 5..0)
- Indicador Z i C de l’ALU

Recordeu que el format de les instruccions del processador a dissenyar és el següent:

31 26	25 21	20 16	15 11	10 6	5 0
co	rs	rt	rd	0	func
6 bits	5 bits	5 bits	5 bits	5 bits	6 bits

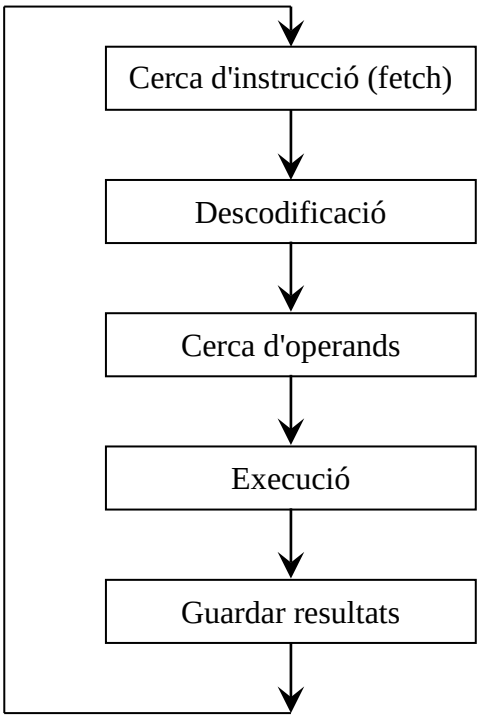
Els codis d'operació (co) i funció (func) són els següents per a les diferents instruccions que executa el processador:

	c.o.						**Funció**					
Instrucció	**C5**	**C4**	**C3**	**C2**	**C1**	**C0**	**F5**	**F4**	**F3**	**F2**	**F1**	**F0**
add	0	0	0	0	0	0	1	0	0	0	0	0
sub	0	0	0	0	0	0	1	0	0	0	1	0
and	0	0	0	0	0	0	1	0	0	1	0	0
or	0	0	0	0	0	0	1	0	0	1	0	1
slt	0	0	0	0	0	0	1	0	1	0	1	0
addi	0	0	1	0	0	0	x	x	x	x	x	x
andi	0	0	1	1	0	0	x	x	x	x	x	x
ori	0	0	1	1	0	1	x	x	x	x	x	x
lw	1	0	0	0	1	1	x	x	x	x	x	x
sw	1	0	1	0	1	1	x	x	x	x	x	x
beq	0	0	0	1	0	0	x	x	x	x	x	x
bne	0	0	0	1	0	1	x	x	x	x	x	x
J	0	0	0	0	1	0	x	x	x	x	x	x
jr	0	0	0	0	0	0	0	0	1	0	0	0
jal	0	0	0	0	1	1	x	x	x	x	x	x

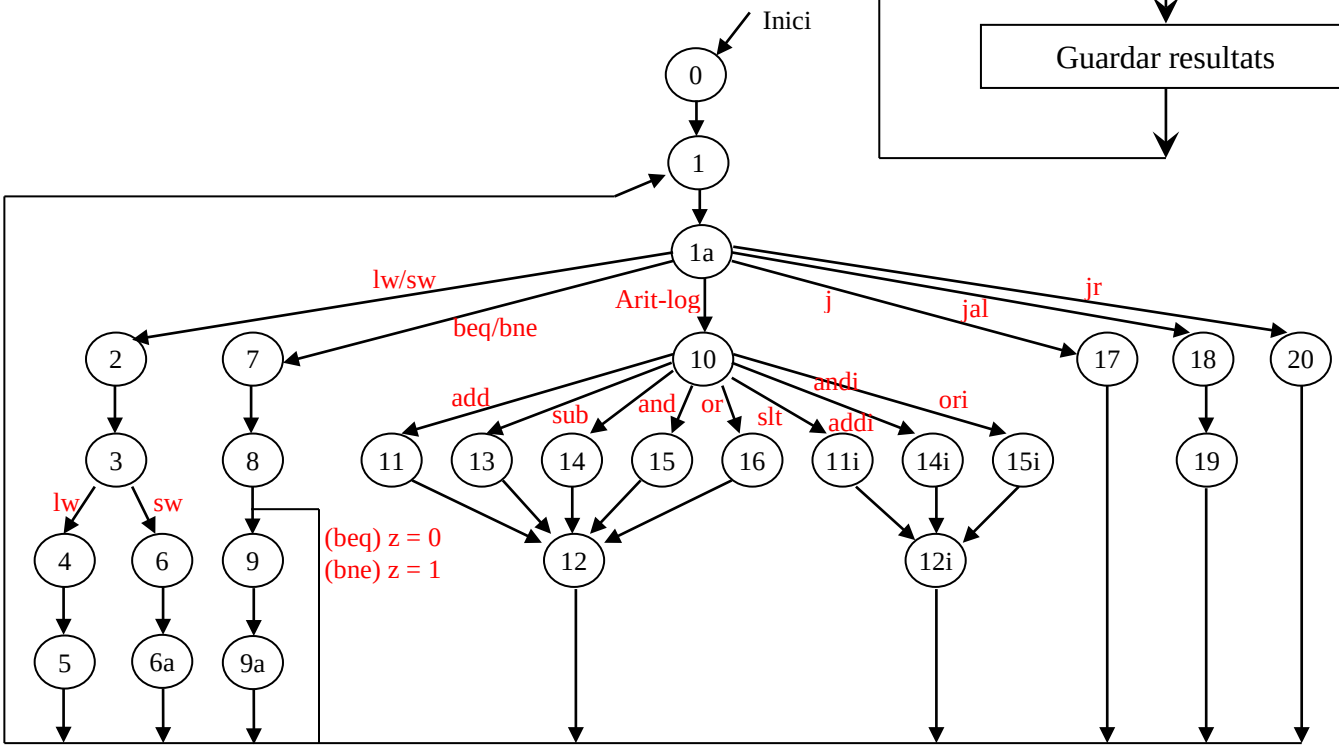
La missió de la unitat de control a més és generar una sèrie de senyals que controlen els altres elements del processador:

- Banc de registres.
- Multiplexor del banc de registres.
- Memòria: a través dels registres MAR i MDR.
- Altres registres.
- Unitats funcionals diverses.
- Obertura i tancament de l'enllaç de bus.
- ALU

La UC és un sistema seqüencial que per a la generació d'aquests senyals de control passa per una sèrie d'estats en funció de les etapes (o fases) per a executar les instruccions. Una primera classificació d'aquests estats o fases de l'execució d'una instrucció podria ser:



Aquestes fases són genèriques, i cal particularitzar-les per a cada instrucció, tenint en compte les diferents instruccions que ha d'executar el processador que s'està dissenyant. Es poden definir els següents estats o fases per a la UC:



A cadascun dels estats representats, la UC haurà d'activar una sèrie de senyals per a controlar els diferents elements del camí de dades.

Els senyals de control dels diferents elements del camí de dades:

Senyal	Descripció
	CONTROL DEL BANC DE REGISTRES
l_sor1,l_sor2	Treuen al bus (posen en baixa impedància) les sortides Sor1 i Sor2 del banc de registres
e_reg	Ordena l'escriptura al banc de registres
mux_dest	Dues línies per a seleccionar el registre destí
	CONTROL DE LA MEMÒRIA
l_mem	Lectura de memòria. S'obté a DADES el contingut de l'adreça de memòria apuntada per MAR
e_mem	Escriptura a memòria. La dada de MDR s'escriu a l'adreça de memòria apuntada per MAR
e_mar	Ordena l'emmagatzemament al registre MAR
l_mdr	Posar al bus el contingut del registre MDR
e_mdr	Ordena l'emmagatzemament al registre MDR
	CONTROL ALTRES REGISTRES
e_ir	Càrrega del registre d'instrucció (IR)
l_pc	Posa al bus el contingut del comptador de programa (PC)
e_pc	Càrrega del registre PC
pc_sup	Filtra l'entrada a l'ALU, deixa passar els 4 bits més significatius del PC i posa la resta a zero
inc_pc	Incrementa en 4 el registre PC
l_desp1,l_desp2	Posen al bus les sortides dels mòduls de desplaçament
l_extsign	Posen al bus la sortida del mòdul d'extensió de signe
e_acc	Càrrega del registre acumulador
l_acc	Posa al bus el contingut del registre acumulador
	CONTROL DE BUSOS
Tancar	El BUS1 i el BUS3 queden interconnectats
	ALU
op_alu	Tres línies per a seleccionar l'operació de l'ALU

En cadascuna de les fases o estats de la UC els senyals a activar són les següents:

Estat	l_sor1	l_sor2	e_reg	mux_dest	l_mem	e_mem	e_mar	l_mdr	e_mdr	e_ir	l_pc	e_pc	pc_sup	inc_pc	l_desp1	l_desp2	l_extsign	l_acc	e_acc	Tancar	op_alu
1	0	0	0	XX	1	0	1	0	1	0	1	0	0	0	0	0	0	0	0	0	XXX
1a	0	0	0	XX	0	0	0	1	0	1	0	0	0	1	0	0	0	0	0	0	XXX
2	0	0	0	XX	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	XXX
3	1	0	0	XX	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	010
4	0	0	0	XX	1	0	1	0	1	0	0	0	0	0	0	0	0	1	0	1	XXX
5	0	0	1	00	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	XXX
6	0	1	0	XX	0	1	1	0	1	0	0	0	0	0	0	0	0	1	0	1	XXX
6a	0	1	0	XX	0	1	1	0	0	0	0	0	0	0	0	0	0	1	0	1	XXX
7	0	0	0	XX	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	XXX
8	1	1	0	XX	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	110
9	0	0	0	XX	0	0	0	0	0	0	1	0	0	0	1	0	0	1	1	0	010
9a	0	0	0	XX	0	0	0	0	0	0	0	1	0	0	0	0	0	1	0	0	000
10	0	0	0	XX	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	XXX
11	1	1	0	XX	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	010
11i	1	0	0	XX	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	010
12	0	0	1	01	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	XXX
12i	0	0	1	00	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	XXX
13	1	1	0	XX	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	110
14	1	1	0	XX	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	000
14i	1	0	0	XX	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	000
15	1	1	0	XX	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	001
15i	1	0	0	XX	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	001
16	1	1	0	XX	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	111
17	0	0	0	XX	0	0	0	0	0	0	1	1	1	0	0	1	0	1	1	0	010
18	0	0	1	10	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	1	XXX
19	0	0	0	XX	0	0	0	0	0	0	1	1	1	0	0	1	0	1	1	0	010
20	1	0	0	XX	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1	XXX

4. Descripció de la practica

En aquesta secció es descriu com realitzar la unitat de control descrita a l'apartat anterior. En primer lloc es presenta la definició de l'entitat a realitzar. Tindrà la següent definició d'entrades i sortides:

```
entity UnitatDeControl is
Port ( Reset : in STD_LOGIC;
    clk : in STD_LOGIC;
    co : in STD_LOGIC_VECTOR (5 downto 0);
    func : in STD_LOGIC_VECTOR (5 downto 0);
    z : in STD_LOGIC;
    c : in STD_LOGIC;
    l_sor1 : out STD_LOGIC;
    l_sor2 : out STD_LOGIC;
    e_reg : out STD_LOGIC;
    mux_dest : out STD_LOGIC_VECTOR (1 downto 0);
    l_mem : out STD_LOGIC;
    e_mem : out STD_LOGIC;
    e_mar : out STD_LOGIC;
    l_mdr : out STD_LOGIC;
    e_mdr : out STD_LOGIC;
    e_ir : out STD_LOGIC;
    l_pc : out STD_LOGIC;
    e_pc : out STD_LOGIC;
    pc_sup : out STD_LOGIC;
    inc_pc : out STD_LOGIC;
    l_desp1 : out STD_LOGIC;
    l_desp2 : out STD_LOGIC;
    l_extsign : out STD_LOGIC;
    l_acc : out STD_LOGIC;
    e_acc : out STD_LOGIC;
    Tancar : out STD_LOGIC;
    op_alu : out STD_LOGIC_VECTOR (2 downto 0)
);
end UnitatDeControl;
```

on reset, clk, co, func, z i c són entrades de la UC corresponents al reset de la unitat de control que la porta a l'estat inicial, el codi d'operació de la instrucció, la funció de la mateixa i l'indicador de zero i carry de l'ALU. La resta es correspon als senyals que ha d'activar la UC per governar tots els elements del camí de dades que permeten l'execució de les diferents instruccions. Cadascun d'aquest senyals s'ha descrit a l'apartat anterior.

Com s'ha comentat, la unitat de control és un sistema seqüencial síncron que s'ha d'implementar com un autòmat de Moore. La implementació a realitzar en VHDL serà comportamental. Per tant, tindrem dos processos: un per obtindre l'estat següent i un altre per aconseguir les sortides en funció de l'estat en què ens trobem.

La plantilla de codi pel procés que controla l'estat següent pot ser:

```
architecture Behavioral of UnitatDeControl is
type Tipus_Estats is (E1, E2, E3, E4, E5, E6, E7, E8, E9,
E10, E11, E11i, E12, E12i, E13, E14, E14i, E15, E15i, E16,
E17, E18, E19, E20);
signal Estat: Tipus_Estats;
begin
    transicions: process (clk)
        begin
            if falling_edge(clk) then
                if (reset = '1') then
                    Estat <= E0;
                else
                    case Estat is
                        when E0 => Estat <= E1;
                        when E1 => ...
                        when E2 => ...
                        when E3 =>
                            if co = "100011" then
                                Estat <= E4;
                            else
                                Estat <= E6;
                            end if;
                        when E4 =>
                            ...
                        ...
                    end case;
                end if;
            end if;
        end process;
```

La plantilla de codi pel procés que controla les sortides pot ser:

```
sortides: process (Estat)
    begin
        case Estat is
            when E0 =>
                l_sor1 <= '0';
                l_sor2 <= '0';
                e_reg <= '0';
                mux_dest <= "00";
                l_mem <= '0';
                e_mem <= '0';
                e_mar <= '0';
                l_mdr <= '0';
                e_mdr <= '0';
                e_ir <= '0';
                l_pc <= '0';
                e_pc <= '0';
                pc_sup <= '0';
                inc_pc <= '0';
                l_desp1 <= '0';
                l_desp2 <= '0';
                l_extsign <= '0';
                l_acc <= '0';
                e_acc <= '0';
                Tancar <= '0';
                op_alu <= "000";
            when E1 =>
                l_sor1 <= '0';
                l_sor2 <= '0';
                e_reg <= '0';
                mux_dest <= "00";
                l_mem <= '1';
                e_mem <= '0';
                e_mar <= '1';
                l_mdr <= '0';
                e_mdr <= '1';
                e_ir <= '0';
                l_pc <= '1';
                e_pc <= '0';
                pc_sup <= '0';
                inc_pc <= '0';
                l_desp1 <= '0';
                l_desp2 <= '0';
                l_extsign <= '0';
                l_acc <= '0';
                e_acc <= '0';
                Tancar <= '0';
                op_alu <= "000";
            when E2 =>
                ...
        end case;
    end process;
end Behavioral;
```

5. Realització pràctica

5.1. Exercici 1

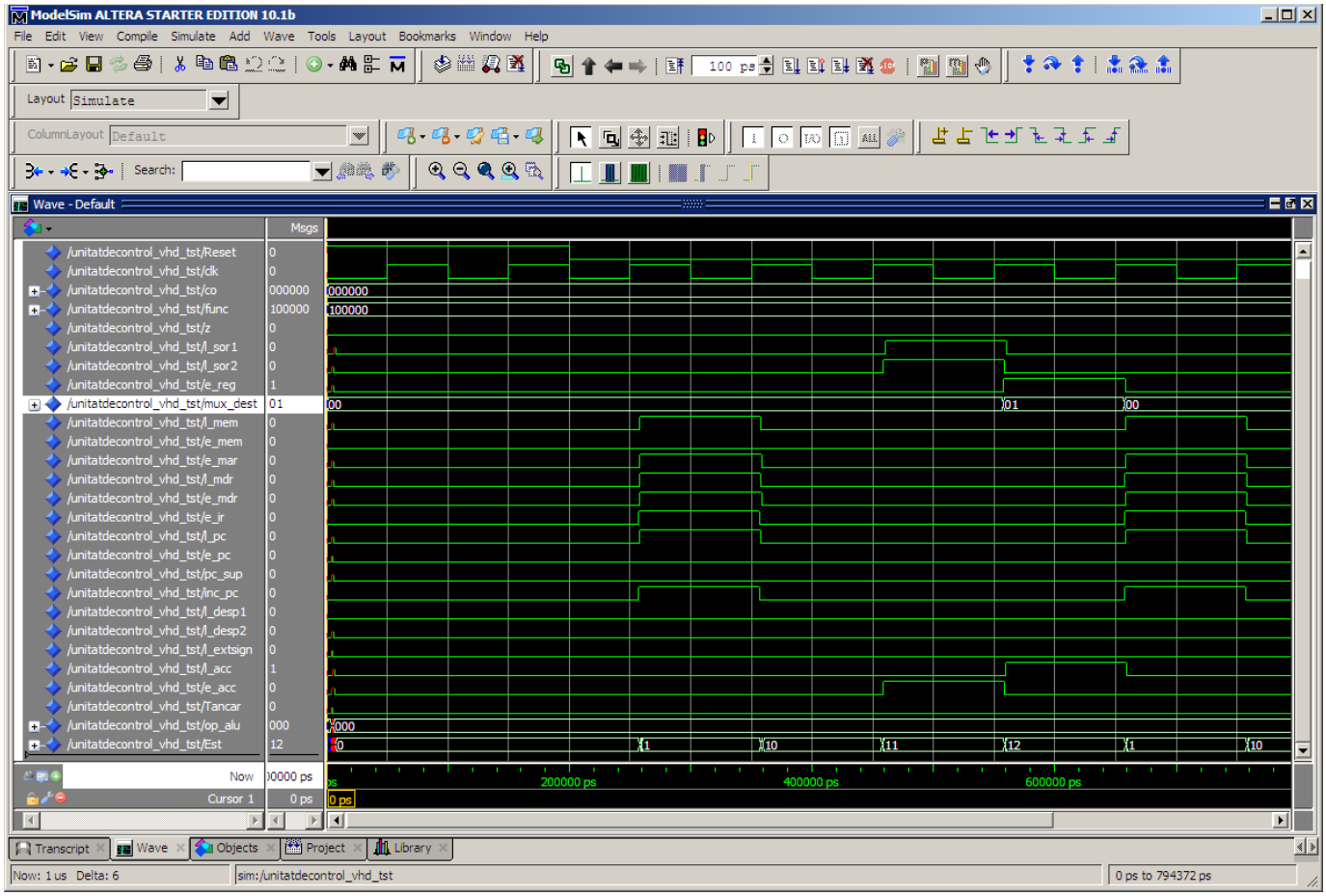
Realitzeu el disseny corresponent al circuit de transició entre estats. Per a cada estat caldrà indicar la transició a l'estat següent en funció de les entrades. Com exemple es mostra el codi corresponent a l'estat 3.

5.2. Exercici 2

Realitzeu el disseny corresponent al circuit de sortides. Per a cada estat caldrà indicar les sortides. Com exemple es mostra l'activació de les sortides corresponents als estats 0 i 1.

5.3. Exercici 3

A continuació s'ha de comprovar mitjançant la simulació que el disseny funciona correctament. Per a això es realitzaran un conjunt de simulacions comportamentals per mitjà de l'opció Simulate Behavioral Model. Cadascuna representarà la execució separada de les instruccions **add**, **lw** i **jal**.



Per a cadascuna d'aquestes simulacions, s'establirà un senyal de rellotge amb un període de 100 ns i un temps de simulació total de 1000 ns. Es recomana començar cada simulació amb l'activació del senyal de reset. A més, s'haurà d'establir els valors de les entrades de la UC (reset, clk, co, func, Z i C) per a executar la instrucció corresponent i comprovar que els senyals de sortida implicades en cada una de les instruccions que es simulin s'activen correctament.

Per a poder seguir millor l'execució de cada instrucció es recomana visualitzar també a la simulació l'estat en què es troba en cada moment la UC. Això es pot aconseguir modificant el disseny i afegint la sortida “Estat” al dispositiu.

Ompliu la següent taula indican quantes fases té cada instrucció i quins senyals s’activen en cadascuna. Per a simplificar, indiqueu únicament quins senyals prenen un valor diferent del inicial (‘0’, “00”, o “000” segon el cas). En el cas del senyal op_alu, indiqueu també quan ha de valdre “000”.

	add					**jal**					**lw**				
Etapa	**1**	**2**	**3**	**4**	**5**	**1**	**2**	**3**	**4**	**5**	**1**	**2**	**3**	**4**	**5**
l_pc															
inc_pc															
e_pc															
l_extsig															
l_desp1															
l_desp2															
e_ir															
mux_desp															
e_reg															
l_sor1															
l_sort2															
e_acc															
l_acc															
pc_sup															
op_alu															
Tancar															
e_mar															
e_mdr															
l_mdr															
l_mem															

5.4. Exercici 4

Com en les practiques anteriors, amb el disseny de la unitat de control creada als apartats anteriors creeu un símbol nou. Amb l’editor d’esquemàtics del “Quartus” dissenyeu un esquema que utilitzin aquest nou component (només l’heu de connectar a unes entrades i sortides) i verifiqueu el seu correcte funcionament amb el “ModelSim”.

En tots els casos, heu de presentar el codi VHDL o esquemàtic realitzat per a cadascun dels exercicis i les simulacions que demostren el funcionament correcte dels circuits realitzats.