DISEÑO DE UN COMPUTADOR BÁSICO CON PLD's

J. GARCÍA ZUBÍA[1]
[1]*Departamento de Arquitectura de Computadores. Facultad de Ingeniería. Universidad de Deusto. Apdo. 1. 48080, Bilbao. zubia@inf.deusto.es*

En el trabajo se muestra que es posible para el alumno implementar un computador básico RISC utilizando PLD's. De esta forma el alumno aprende PLD's, a la vez que asienta y hace reales sus conocimientos de Estructura/Arquitectura de Computadores.

1. Introducción

En una facultad de Ingeniería, la disciplina de Arquitectura/Estructura de Computadores suele ocupar un mínimo de tres asignaturas, si no son más. Dentro de esta formación, el alumno adquiere conocimiento teórico y práctico, pero en ningún caso suele *hacerlos reales*. Las siguientes páginas mostrarán que no sólo es posible implementar un computador básico tipo RISC, sino que además es bastante fácil apoyándose en PLD's.

2. La Máquina Sencilla

La MS es ya un clásico dentro de los computadores básicos, desarrollada por M. Valero y E. Ayquadé en 1989 en la Universidad Politécnica de Cataluña. La MS tiene cuatro instrucciones: ADD F, D, MOV F, D, CMP F, D y BEQ D. Repasaremos brevemente el comportamiento y estructura de la MS en los siguientes párrafos y figuras.

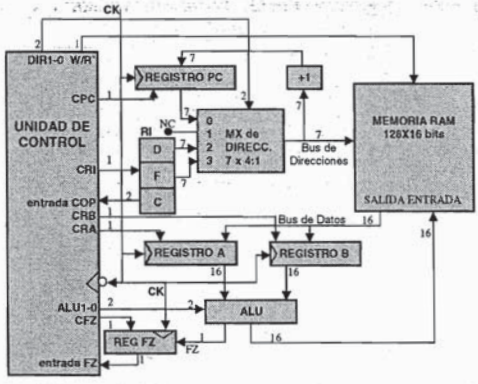

Figura 1: *Estructura de la Máquina Sencilla*

3. Bloques de la Máquina Sencilla

La MS queda descrita por la figura 1, por los siguientes párrafos y por la figura 2

Registros de la MS con línea de carga PC, RI, A, B y FZ
Si habiendo un flanco ascendente en el reloj, la línea de carga fuera 1 (CR=1) entonces el registro se cargaría con el contenido de la entrada, en caso contrario (línea de carga igual a 0) el registro no se cargaría. El comportamiento del registro PC está en la primera tabla de la figura 2 –el resto de registros se comporta de forma similar-.

ALU de la MS
La salida de la ALU es la suma de las entradas, su XOR o el propio operando B según sea el valor de las líneas ALU1-0. Ver la segunda tabla de la figura 2.

Multiplexor de direcciones de la MS
Encamina al bus de direcciones, según sea el valor de las líneas DIR1-0, la dirección contenida en el PC, en FUENTE o en DESTINO. Ver la tercera tabla de la figura 2.

Memoria de lectura/escritura de la MS
La memoria, según sea el valor de W/R', almacenará en la posición indicada por el bus de direcciones el contenido del bus de datos (W/R'=1, escritura), o situará en el bus de datos el contenido de la posición indicada por el bus de direcciones (W/R'=0, lectura). También dispone de la línea CP_t cuya activación recarga la memoria con el programa y datos originales. Ver la cuarta tabla de la figura 2.

Sumador +1 de la MS
Siempre suma 1 al contenido del registro PC, de este modo PC siempre apuntará a la posición de la siguiente instrucción a procesar.

Unidad de Control de la Máquina Sencilla.
La Unidad de Control quizá sea el elemento más complejo de la MS, y se describe mediante el autómata de Moore y la tabla de salidas correspondiente de la figura 2.

CPC	Operación
0	Qt=Qt-1
1	Qt=Et

ALU1-0	Operación
00	S=A+B
01	S=A⊕B
10	S=B
11	Libre

DIR1-0	Operación
00	B. DIR=PC
01	Libre
10	B. DIR=FUENTE
11	B. DIR=DESTINO

W/R'	Operación
0	B. DATOS=(B. DIR)
1	(B. DIR)=B. DATOS

	COP	Dir. Fuente							Dir. Destino							
ADD F, D (D)←(D)+(F) carga FZ	00	F6	F5	F4	F3	F2	F1	F0	D6	D5	D4	D3	D2	D1	D0
CMP F, D (D) ⊕ (F) carga FZ	01	F6	F5	F4	F3	F2	F1	F0	D6	D5	D4	D3	D2	D1	D0
MOV F, D (D) ← (F) cargaFZ	10	F6	F5	F4	F3	F2	F1	F0	D6	D5	D4	D3	D2	D1	D0
BEQ D, si FZ=1 RI ← (D), PC←PC+1	11	X	X	X	X	X	X	X	D6	D5	D4	D3	D2	D1	D0

Si	OPERACIÓN	DIR1	DIR0	ALU1	ALU0	W/R'	CPC	CRI	CA	CB	CFZ
S0	RI ← (PC)	0	0	X	X	0	1	1	0	0	0
S1	Decodificar	X	X	X	X	0	0	0	0	0	0
S2	B ← (F)	1	0	X	X	0	0	0	0	1	0
S3	A ← (D)	1	1	X	X	0	0	0	1	0	0
S4	(D) ← A + B	1	1	0	0	1	0	0	0	0	1
S5	A ⊕ B	X	X	0	1	0	0	0	0	0	1
S6	(D) ← B	1	1	1	0	1	0	0	0	0	1
S7	RI ← (D)	1	1	X	X	0	1	1	0	0	0

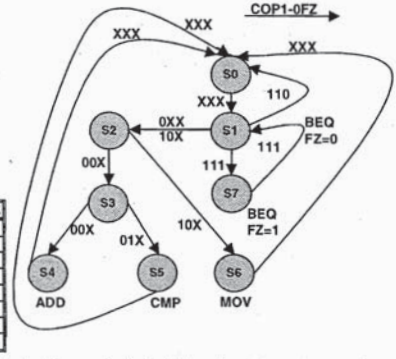

Figura 2: *Distintos aspectos de la Unidad de Control de la Máquina Sencilla*

4. La Máquina Sencilla en PLD's

Para implementar la MS simplemente hay que ir paso por paso:

- programar y simular cada bloque en ORCAD-PLD,
- cablear cada bloque por separado y comprobarlo con una barra de diodos led,
- unir y comprobar la Unidad de Control (generador de secuencia y obtención de salidas),
- unir y comprobar los dos bloques de la ALU,
- unir y comprobar los bloques del Bus de Datos,
- unir y comprobar los bloques del Bus de Direcciones (PC, RI y Mx de Direcciones),
- unir y comprobar los 8 bloques de la memoria,
- unirlo todo y comprobarlo.

Antes de implementar la MS, hay que decir que lo implementado es una reducción de ella: el bus de datos es de 8 bits, por lo tanto el bus de direcciones será de 3 bits, resultando que la memoria tendrá 8 posiciones de 8 bits. En la tabla 1 se muestra el código en ORCAD-PLD para PAL 22V10 correspondiente a los bloques: registro A, salida de la Unidad de Control, generador de estados de la Unidad de Control y la ALU del Camino de Datos dividida en dos: ALU1 y ALU2. El resto de los bloques se codificará de modo semejante, siendo todos ellos de una gran sencillez.

Registro A
Salidas de U. Control

```
| PAL22V10
| in: (CA, E[7..0],
CLEAR), io: S[7..0],
clock:CK,
reset:RESET
|Title:"Registro MS"
| Registers: CK //
S[7..0]
| RESET=CLEAR

| Map: S[7..0] ->
S[7..0]
|(n -> E[7..0], CA
| n -> n,          CA')

| PAL22V10
| in: (C[2..0]), io:
(DIR1, DIR0, ALU1,
ALU0, WR, PC, CRI,
CA, CB, CFZ)
|Title:"Salidas UC"

| Table: C[2..0] ->
DIR1, DIR0, ALU1,
ALU0, WR, PC, CRI,
CA, CB, CFZ
|(000 -> 0000011000b
| 001 -> 0000000000b
| 010 -> 1000000010b
| 011 -> 1100000100b
| 100 -> 1100100001b
| 101 -> 0001000001b
| 110 -> 1110100001b
| 111 -> 1100011000b
| )
```

Generador de estados de la Unidad Control

```
| PAL22V10
| in: (INIC,C[1..0],FZ),
io: (Q[2..0]), Clock:CK,
reset:RESET
| Title: "Maquina de
Estados de la Unidad de
Control"
| Register: CK // Q[2..0]
| RESET=INIC


| Procedure: INIC, Q[2..0]
| {
| States: E0=0, E1=1,
E2=2, E3=3, E4=4, E5=5,
E6=6, E7=7
|E0.                -> E1
|E1.(C[1..0]==3)&FZ ?-> E7
|   (C[1..0]==3)&FZ'?-> E0
|                   -> E2
|E2.(C1==0)         ?-> E3
|   (C[1..0]==2)    ?-> E6
|                   -> E0
|E3.(C[1..0]==0)    ?-> E4
|   (C[1..0]==1)    ?-> E5
|                   -> E0
|E4.(C[1..0]==0)    ?-> E0
|                   -> E0
|E5.(C[1..0]==1)    ?-> E0
|                   -> E0
|E7.                -> E1
|E6.(C[1..0]==2)    ?-> E0
|                   -> E0
| )
```

ALU de la MS: ALU1 y ALU2

```
| PAL22V10
| in: (A[3..0], B[3..0], alu[1..0]),
io:(DATOS[3..0], FZ,  C[3..0])
| Title: "Primera parte de la ALU de la MS"

|DATOS[3..0]=(A[3..0]##B[3..0])&(ALU[1..0]==1)
|DATOS[3..0]=(B[3..0])&(ALU[1..0]==2)
|DATOS[0]=((A[0]+B[0])[0])&(ALU[1..0]==0)
|C[0]=((A[0]+B[0])[1])&(ALU[1..0]==0)
|DATOS[1]=((A[1]+B[1]+C[0])[0])&(ALU[1..0]==0)
|C[1]=((A[1]+B[1]+C[0])[1])&(ALU[1..0]==0)
|DATOS[2]=((A[2]+B[2]+C[1])[0])&(ALU[1..0]==0)
|C[2]=((A[2]+B[2]+C[1])[1])&(ALU[1..0]==0)
|
DATOS[3]=((A[3]+B[3]+C[2])[0])&(ALU[1..0]==0)
|C[3]=((A[3]+B[3]+C[2])[1])&(ALU[1..0]==0)
|FZ=(DATOS[3..0]==0)


| PAL22V10
| in: (A[7..4],B[7..4],ALU[1..0],C[3],EFZ),
io:(DATOS[7..4],FZ,C[6..4])
| Title: "Bloque 2 de la ALU de la MS"

|DATOS[7..4]=(A[7..4]##B[7..4])&(ALU[1..0]==1)
|DATOS[7..4]=(B[7..4])&(ALU[1..0]==2)
|DATOS[4]=((A[4]+B[4]+C[3])[0])&(ALU[1..0]==0)
|C[4]=((A[4]+B[4]+C[3])[1])&(ALU[1..0]==0)
|DATOS[5]=((A[5]+B[5]+C[4])[0])&(ALU[1..0]==0)
|C[5]=((A[5]+B[5]+C[4])[1])&(ALU[1..0]==0)
|DATOS[6]=((A[6]+B[6]+C[5])[0])&(ALU[1..0]==0)
|C[6]=((A[6]+B[6]+C[5])[1])&(ALU[1..0]==0)
|DATOS[7]=((A[7]+B[7]+C[6])[0])&(ALU[1..0]==0)
|FZ=(DATOS[7..4]==0)&(EFZ==1)
```

Tabla 1: *Código en ORCAD-PLD de algunos bloques de la Maquina Sencilla*

5. Implementación y uso de la Máquina Sencilla

La foto de la figura 3 muestra el aspecto general de la MS y algunos bloques en particular.

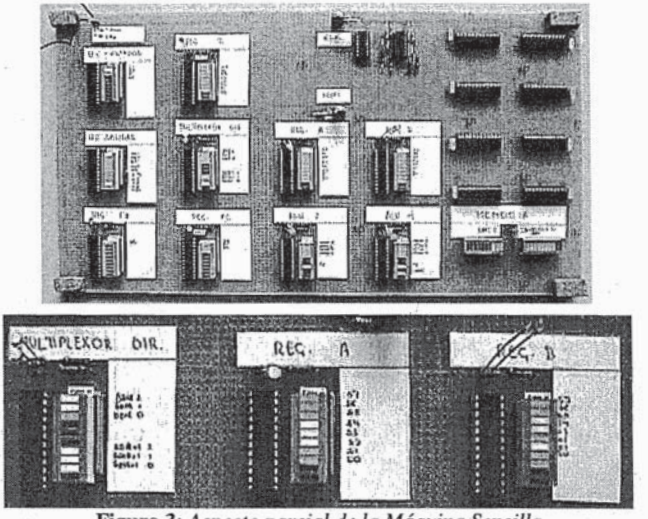

Figura 3: *Aspecto parcial de la Máquina Sencilla*

En cuanto al uso, la MS implementada evoluciona microinstrucción a microinstrucción, perimitiendo que el alumno contemple de forma exhaustiva el comportamiento del computador. Para utilizar la MS primero habrá que activar el interruptor de RESET, recargándose el programa original en memoria. En este momento la MS estará en el estado 0, seguidamente generaremos un flanco ascendente mediante el interruptor del CK, lo que conllevará la activación de los distintos bloques en función de los valores generados por la Unidad de Control. Ya está procesado el estado 0. Si volvemos a activar el CK, generaremos un flanco descendente, pasando la UC a su nuevo estado, en este caso el 1. Activando el CK iremos procesando microinstrucción a microinstrucción todo el programa. Si quisiéramos cambiar de programa habría que reprogramar los PLD's de la memoria.

6. Conclusiones

Ya sea en un curso de Estructura/Arquitectura de Computadores o en un curso de PLD's, se puede implementar con facilidad un computador básico RISC, en nuestro caso la Máquina Sencilla, con PLD's. La implementación de la MS cubre dos objetivos, por un lado el alumno cursa PLD's, y por otro asume y comprende mejor la estructura y comportamiento de un computador.

Referencias

[1] J.M. Angulo, *Introducción a los Computadores*, Ed. Paraninfo (1995)
[2] M. Barrón, *Lógica Programable*, Ed. McGraw Hill (1994)