

# Trabajo Práctico N° 1

## Descripción HDL de un diseño

### Objetivos

Dadas las especificaciones funcionales que el diseño debe cumplir se deberá diseñar a nivel RTL un circuito lógico que cumpla las mismas. Asimismo se realizará la descripción de un banco de pruebas (*testbench*) para verificar el funcionamiento del mismo.

### Especificaciones Funcionales

Describir a nivel RTL un procesador sencillo con arquitectura *Hardvard* de tipo acumulador.

- Deberá tener un bus de datos y de direcciones ambos de 8 bits. Es decir que podrá direccionar 256 palabras de 8 bits tanto de programa como de datos.
- Constará de una señal de lectura/escritura de la memoria de datos que cuando es '1' indicará que la operación corresponde a una escritura.
- Tendrá una entrada de reloj para sincronizar todas las operaciones.
- Tendrá una entrada de *power-on-reset*.
- El conjunto de instrucciones será de 24 instrucciones, el cual es descripto posteriormente.
- El contador de programa será de 8 bits de forma tal que pueda direccionar las 256 palabras de memoria.
- Tendrá dos modos de direccionamiento: directo e inmediato.
- Habrá un acumulador de 8 bits. Todas las operaciones aritméticas del procesador serán de tipo AM (acumulador-memoria).
- Habrá tres *flags* de estado: *carry*, *zero* y *negative*.

### Conjunto de instrucciones

- Instrucciones de Movimiento de Datos

Nemónico	Operación	Descripción
Load A,X	$A \leftarrow M[X]$	Carga en el acumulador el contenido de la dirección de memoria X.
Store X,A	$M[X] \leftarrow A$	Almacena en la dirección de memoria X el contenido del acumulador.
Loadi A,I	$A \leftarrow I$	Carga en el acumulador un valor inmediato.
Storei A,I	$M[A] \leftarrow I$	Almacena en la dirección de memoria dada por el acumulador un valor inmediato.

- **Instrucciones Aritméticas**

Nemónico	Operación	Descripción
Add A,X	$A \leftarrow A + M[X]$	Suma al acumulador el contenido de la dirección de memoria X.
Sub A,X	$A \leftarrow A - M[X]$	Sustraer del acumulador el contenido de la dirección de memoria X.
Addc A,X	$A \leftarrow A + M[X] + Cy$	Suma al acumulador el contenido de la dirección de memoria X y el flag de carry.
Subc A,X	$A \leftarrow A - M[X] - Cy$	Sustraer del acumulador el contenido de la dirección de memoria X y el flag de carry.
Addi A,I	$A \leftarrow A + I$	Suma al acumulador un valor inmediato.
Subi A,I	$A \leftarrow A - I$	Sustraer del acumulador un valor inmediato.
Addic A,I	$A \leftarrow A + I + Cy$	Suma al acumulador un valor inmediato y el flag de carry.
Subic A,I	$A \leftarrow A - I - Cy$	Sustraer del acumulador un valor inmediato y el flag de carry.

- **Instrucciones Lógicas**

Nemónico	Operación	Descripción
Nor A,X	$A \leftarrow \sim(A \mid M[X])$	Realiza la operación lógica NOR entre el acumulador y el contenido de la dirección de memoria X.
Nand A,X	$A \leftarrow \sim(A \& M[X])$	Realiza la operación lógica NAND entre el acumulador y el contenido de la dirección de memoria X.
Xor A,X	$A \leftarrow A \wedge M[X]$	Realiza la operación lógica XOR entre el acumulador y el contenido de la dirección de memoria X.
Xnor A,X	$A \leftarrow \sim(A \wedge M[X])$	Realiza la operación lógica XNOR entre el acumulador y el contenido de la dirección de memoria X.
Nori A,I	$A \leftarrow \sim(A \mid I)$	Realiza la operación lógica NOR entre el acumulador y un valor inmediato.
Nandi A,I	$A \leftarrow \sim(A \& I)$	Realiza la operación lógica NAND entre el acumulador y un valor inmediato.
Xori A,I	$A \leftarrow A \wedge I$	Realiza la operación lógica XOR entre el acumulador y un valor inmediato.
Xnori A,I	$A \leftarrow \sim(A \wedge I)$	Realiza la operación lógica XNOR entre el acumulador y un valor inmediato.

- Instrucciones de Salto**

Nemónico	Operación	Descripción
Jump X	$PC \leftarrow PC + X$	Salto incondicional relativo al program counter.
Jz X	if (Z==1) then {PC $\leftarrow$ PC + X} else {PC $\leftarrow$ PC + 2}	Salto condicional relativo al program counter si el acumulador fue cero en la instrucción previa.
Jc X	if (C==1) then {PC $\leftarrow$ PC + X} else {PC $\leftarrow$ PC + 2}	Salto condicional relativo al program counter si hubo acarreo en la instrucción previa.
Jn X	if (N==1) then {PC $\leftarrow$ PC + X} else {PC $\leftarrow$ PC + 2}	Salto condicional relativo al program counter si MSB del acumulador fue '1' en la instrucción previa.

### Formato de instrucciones

Cada instrucción constará de dos palabras consecutivas de 8 bits. La primera palabra se posicionará en una dirección par de memoria, mientras que la segunda lo hará en una dirección impar. La primera palabra definirá el código de instrucción, mientras que la segunda definirá el valor inmediato o la dirección de memoria según corresponda.

- Instrucciones de Movimiento de Datos**

		I <sub>7</sub>	I <sub>6</sub>	I <sub>5</sub>	I <sub>4</sub>	I <sub>3</sub>	I <sub>2</sub>	I <sub>1</sub>	I <sub>0</sub>
Load	A,X	0	0	0	0	0	0	0	0
		X <sub>7</sub>	X <sub>6</sub>	X <sub>5</sub>	X <sub>4</sub>	X <sub>3</sub>	X <sub>2</sub>	X <sub>1</sub>	X <sub>0</sub>
Store	A,X	0	0	0	0	0	0	0	1
		X <sub>7</sub>	X <sub>6</sub>	X <sub>5</sub>	X <sub>4</sub>	X <sub>3</sub>	X <sub>2</sub>	X <sub>1</sub>	X <sub>0</sub>
Loadi	A,I	0	0	0	0	0	0	1	0
		I <sub>7</sub>	I <sub>6</sub>	I <sub>5</sub>	I <sub>4</sub>	I <sub>3</sub>	I <sub>2</sub>	I <sub>1</sub>	I <sub>0</sub>
Storei	A,I	0	0	0	0	0	0	1	1
		I <sub>7</sub>	I <sub>6</sub>	I <sub>5</sub>	I <sub>4</sub>	I <sub>3</sub>	I <sub>2</sub>	I <sub>1</sub>	I <sub>0</sub>

- Instrucciones Aritméticas**

		I <sub>7</sub>	I <sub>6</sub>	I <sub>5</sub>	I <sub>4</sub>	I <sub>3</sub>	I <sub>2</sub>	I <sub>1</sub>	I <sub>0</sub>
Add	A,X	0	1	0	0	0	0	0	0
		X <sub>7</sub>	X <sub>6</sub>	X <sub>5</sub>	X <sub>4</sub>	X <sub>3</sub>	X <sub>2</sub>	X <sub>1</sub>	X <sub>0</sub>
Sub	A,X	0	1	0	0	0	0	0	1
		X <sub>7</sub>	X <sub>6</sub>	X <sub>5</sub>	X <sub>4</sub>	X <sub>3</sub>	X <sub>2</sub>	X <sub>1</sub>	X <sub>0</sub>
Addc	A,I	0	1	0	0	0	0	1	0
		I <sub>7</sub>	I <sub>6</sub>	I <sub>5</sub>	I <sub>4</sub>	I <sub>3</sub>	I <sub>2</sub>	I <sub>1</sub>	I <sub>0</sub>
Subc	A,I	0	1	0	0	0	0	1	1
		I <sub>7</sub>	I <sub>6</sub>	I <sub>5</sub>	I <sub>4</sub>	I <sub>3</sub>	I <sub>2</sub>	I <sub>1</sub>	I <sub>0</sub>

		$I_7$	$I_6$	$I_5$	$I_4$	$I_3$	$I_2$	$I_1$	$I_0$
Addi	A,I	0	1	0	0	0	1	0	0
		$I_7$	$I_6$	$I_5$	$I_4$	$I_3$	$I_2$	$I_1$	$I_0$
Subi	A,I	0	1	0	0	0	1	0	1
		$I_7$	$I_6$	$I_5$	$I_4$	$I_3$	$I_2$	$I_1$	$I_0$
Addic	A,I	0	1	0	0	0	1	1	0
		$I_7$	$I_6$	$I_5$	$I_4$	$I_3$	$I_2$	$I_1$	$I_0$
Subic	A,I	0	1	0	0	0	1	1	1
		$I_7$	$I_6$	$I_5$	$I_4$	$I_3$	$I_2$	$I_1$	$I_0$

- Instrucciones Lógicas

		$I_7$	$I_6$	$I_5$	$I_4$	$I_3$	$I_2$	$I_1$	$I_0$
Nor	A,X	1	0	0	0	0	0	0	0
		$X_7$	$X_6$	$X_5$	$X_4$	$X_3$	$X_2$	$X_1$	$X_0$
Nand	A,X	1	0	0	0	0	0	0	1
		$X_7$	$X_6$	$X_5$	$X_4$	$X_3$	$X_2$	$X_1$	$X_0$
Xor	A,I	1	0	0	0	0	0	1	0
		$X_7$	$X_6$	$X_5$	$X_4$	$X_3$	$X_2$	$X_1$	$X_0$
Xnor	A,I	1	0	0	0	0	0	1	1
		$X_7$	$X_6$	$X_5$	$X_4$	$X_3$	$X_2$	$X_1$	$X_0$
Nori	A,X	1	0	0	0	0	1	0	0
		$I_7$	$I_6$	$I_5$	$I_4$	$I_3$	$I_2$	$I_1$	$I_0$
Nandi	A,X	1	0	0	0	0	1	0	1
		$I_7$	$I_6$	$I_5$	$I_4$	$I_3$	$I_2$	$I_1$	$I_0$
Xori	A,I	1	0	0	0	0	1	1	0
		$I_7$	$I_6$	$I_5$	$I_4$	$I_3$	$I_2$	$I_1$	$I_0$
Xnori	A,I	1	0	0	0	0	1	1	1
		$I_7$	$I_6$	$I_5$	$I_4$	$I_3$	$I_2$	$I_1$	$I_0$

- Instrucciones de Salto

		$I_7$	$I_6$	$I_5$	$I_4$	$I_3$	$I_2$	$I_1$	$I_0$
Jump	X	1	1	0	0	0	0	0	0
		$X_7$	$X_6$	$X_5$	$X_4$	$X_3$	$X_2$	$X_1$	$X_0$
Jz	X	1	1	0	0	0	0	0	1
		$X_7$	$X_6$	$X_5$	$X_4$	$X_3$	$X_2$	$X_1$	$X_0$
Jc	X	1	1	0	0	0	0	1	0
		$X_7$	$X_6$	$X_5$	$X_4$	$X_3$	$X_2$	$X_1$	$X_0$

$J_n$	$X$	1	1	0	0	0	0	1	1
		$X_7$	$X_6$	$X_5$	$X_4$	$X_3$	$X_2$	$X_1$	$X_0$

## Desarrollo del trabajo práctico

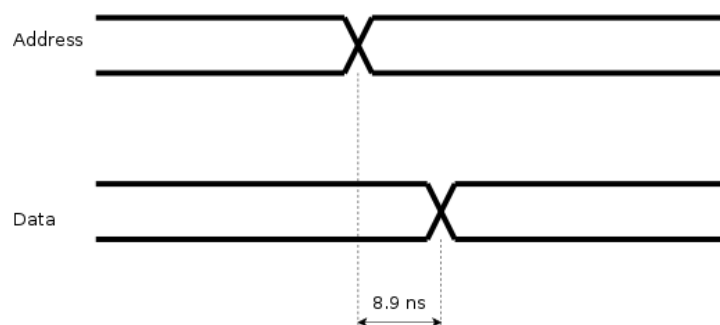
El lenguaje de descripción HDL será Verilog 2001. La verificación del correcto funcionamiento del circuito se realizará mediante la implementación de un programa en código ensamblador que realiza la multiplicación de dos números enteros positivos de 8 bits. Todas las combinaciones deberán ser probadas, es decir se debe verificar el correcto funcionamiento de  $256 \times 256 = 65536$  combinaciones. Las memoria de datos y programa que se conectan al procesador deberán ser inicializada al principio de la simulación con el programa de multiplicación y los multiplicandos en las posiciones 0x00 y 0x01. Asimismo, la correcta operación de todas las instrucciones restantes deberá realizarse mediante *direct case tests*.

Sólo podrá existir una señal de *clock* activa por flanco ascendente y una señal de *power-on-reset* asincrónica. La frecuencia de la señal de reloj será de 100MHz. Todas las entradas y salidas de datos (todas las entradas y salidas excluyendo *clock* y *power-on-reset*) deberán ser registradas.

Las especificaciones de la memoria son descriptas posteriormente. La descripciones de las memorias deberán ser parte integrante del *testbench*.

### Especificaciones de la memoria de programa

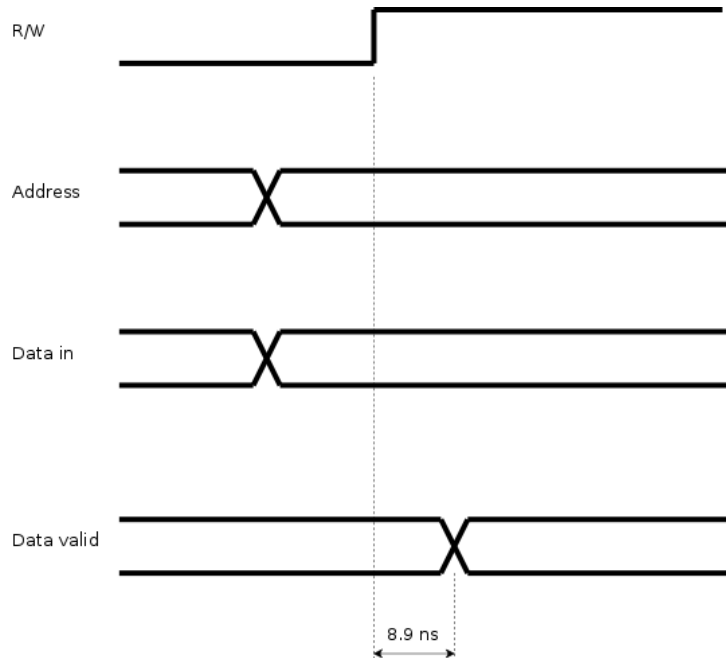
- La memoria de programa tendrá un bus de direcciones de 8 bits y un bus de datos de salida también de 8 bits.
- El tiempo de acceso para la lectura será de 8.9 ns. Es decir si el *address* se modifica, la memoria tomará 8.9 ns en entregar el contenido de la posición de memoria correspondiente a dicho *address*.



### Especificaciones de la memoria de datos

- La memoria de programa será una *dual port* RAM la cual tendrá un bus de direcciones de 8 bits, un bus de datos de entrada de 8 bits y un bus de salida de 8 bits.
- Asimismo, poseerá una señal de lectura/escritura  $\overline{R/W}$  que cuando está en '1' indicará que la memoria debe escribir el dato del bus de entrada en la posición determinada por el bus de direcciones.
- El tiempo de acceso para la lectura y la escritura será de 8.9 ns. Es decir, en lectura, si el *address* se modifica, la memoria tomará 8.9 ns en entregar el contenido de la posición de

memoria correspondiente a dicho *address*. Mientras que en escritura, una vez que el *address* se modifique o la señal *R/W* sea '1', la memoria tomará 8.9 ns en guardar el dato.



### Condiciones mínimas para la aprobación

1. **SÓLO** podrá existir una señal de reloj con flanco ascendente. **NO** debe haber más de una señal de reloj ni puede haber más de un flanco activo.
2. **SÓLO** puede haber una señal de *power-on-reset* que resetee el sistema cuando está a nivel alto '1'.
3. **TODAS** las entradas y salidas del procesador en ambas descripciones deben ser registradas (excepto *clock* y *power-on-reset*).
4. El informe **DEBERÁ** contener el diagrama de estado y el diagrama de la implementación. Asimismo, **DEBERÁ** contener un diagrama de flujo del test de verificación.
5. El trabajo práctico **DEBERÁ** ser presentado en la fecha indicada por la planificación de la asignatura o estará automáticamente desaprobado. Toda la estructura de archivos **DEBERÁ** ser subida y *taggeada* al repositorio de la cátedra para antes de dicha fecha.