UNIVERSIDAD DE COLIMA

FACULTAD DE INGENIERÍA MECÁNICA Y ELÉCTRICA

CURSO: IMPLEMENTACION DE CIRCUITOS DIGITALES

ACTIVIDAD 3: Operadores y Atributos de VHDL

Instructor: Dr. Alberto Manuel Ochoa Brust

Esta actividad está planeada con un enfoque de aprendizaje colaborativo, en el cual lo más importante es el trabajo en equipo para lograr el objetivo de aprender significativamente. Para ello se formarán grupos de 3 personas, que tendrán que leer este documento, siguiendo paso a paso las instrucciones que se indican aquí. La calificación de esta actividad va de 0 a 100 y se asignará proporcionalmente al avance logrado al final de la sesión

---

OPERADORES, OPERANDOS Y EXPRESIONES EN VHDL

Un operador es un símbolo (ejemplos: “<”, “>”, “=”) o una palabra clave (ejemplos NOT, OR, AND) que indica que un dato o varios (operandos) serán procesados para formar una expresión (ejemplos: a < b, z OR x, NOT k,), dicha expresión tendrá un valor de acuerdo al operador y el valor actual de el ó los operandos. Las expresiones en VHDL son muy similares a las de otros lenguajes de programación, por lo que se expondrán brevemente. Como es de esperarse no todos los operadores pueden aplicarse para todos los tipos de datos, por lo que se debe poner atención de cuales son los tipos de datos de los operandos definidos para cada operador, ya que si no es así, se marcará un error de sintaxis a la hora de simular o implementar el diseño.

OPERADORES LÓGICOS

Son NOT, AND, NAND, OR, NOR, XOR y XNOR. El funcionamiento es el habitual para este tipo de operadores. Los operandos deben de ser de tipo BIT, BIT_VECTOR, STD_LOGIC, STD_LOGIC_VECTOR ó BOOLEAN. En el caso de realizarse estas operaciones sobre un dato de tipo STD_LOGIC_VECTOR ó BIT_VECTOR, la operación se realiza bit a bit.

EJEMPLO

Por ejemplo si a='1', b='0', data1="0101", data2="0011", cond1=TRUE, cond2=FALSE, entonces, las siguientes expresiones tendrán los valores indicados en la tabla:

Expresión	Valor de la expresión	Expresión	Valor de la expresión	Expresión	Valor de la expresión
NOT a	'0'	NOT data1	"1010"	NOT cond1	FALSE
a AND b	'0'	dato1 AND dato2	"0001"	cond1 AND cond2	FALSE
a NAND b	'1'	dato1 NAND dato2	"1110"	cond1 NAND cond2	TRUE
a OR b	'1'	dato1 OR dato2	"0111"	cond1 OR cond2	TRUE
a NOR b	'0'	dato1 NOR dato2	"1000"	cond1 NOR cond2	FALSE
a XOR b	'1'	dato1 XOR dato2	"0110"	cond1 XOR cond2	TRUE
a XNOR b	'0'	dato1 XNOR dato2	"1001"	cond1 XNOR cond2	FALSE

Todos los operadores lógicos tienen la misma precedencia, con excepción de NOT cuya precedencia en mayor a la de los demás. Por lo tanto, cuando se usan dos o más operadores con la misma precedencia en la misma expresión, es necesario usar paréntesis para especificar correctamente las operaciones deseadas, sino se hace, se marcaría error de sintaxis, ya que no se sabe cual de las operaciones debe realizarse primero. Por ejemplo, se deseamos hacer la implementar la ecuación lógica: c=xz+x’z, no podemos expresarla como:

```
c <= x AND z OR NOT x AND z;
```

La instrucción correcta sería:

```
c <= (x AND z) OR ((NOT x) AND z);
```

OPERADOR DE CONCATENACIÓN (&)

Concatena arreglos (array) de manera que el resultado será igual a los elementos en el mismo orden en que fueron incluidos en la expresión, por ejemplo si palabra1="Esta", palabra2=" casa es ", la expresión:

```
palabra1 & palabra2 & 'r' & "oja."
```

Es igual a: "Esta casa es roja.". Este operador puede aplicarse a cualquier tipo de arreglo, STD_LOGIC_VECTOR, STRING, etc.

OPERADORES DE DESPLAZAMIENTO

Este tipo de operadores están definidos para los tipos vectoriales BIT_VECTOR y STD_LOGIC_VECTOR.

SLL (Shift Left Logic) y SRL (Shift Right Logic) son operadores de desplazamiento lógico a izquierda y a derecha, respectivamente. Estos operadores desplazan un arreglo de bits cierto número de bits a izquierda (SLL) o derecha (SRL) rellenando los elementos libres con '0'. Este tipo de operadores son muy útiles cuando se requiere multiplicar (SLL) o dividir (SRL) un número binario sin signo (positivo) por una potencia de 2. Por ejemplo, la expresión: dato SLL n, desplaza a la izquierda n bits a la cadena de bits *dato*, es decir, la multiplica por 2^n.

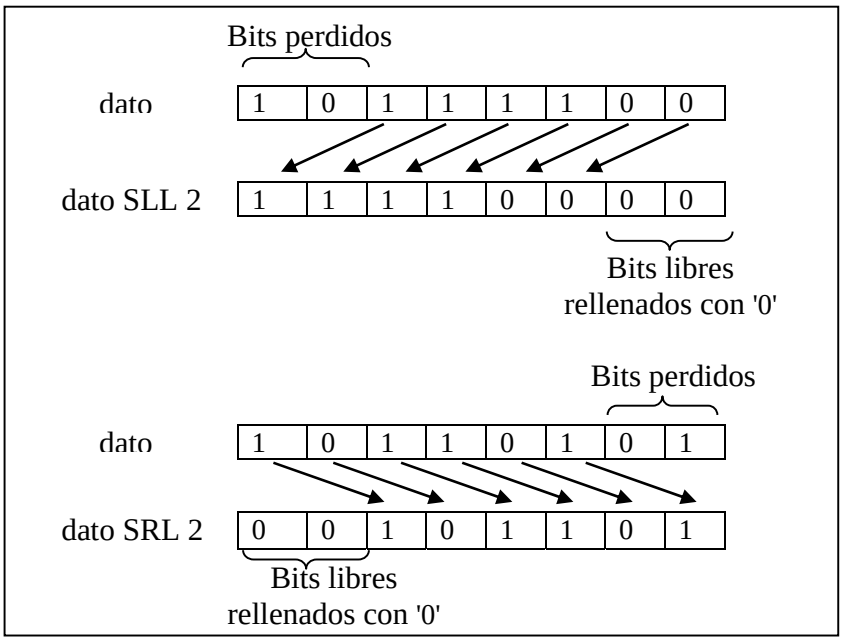


Figura 13.1: Representación de las operaciones de desplazamiento lógico SLL y SRL.

SLA (Shift Left Arithmetic) y SRA (Shift Right Arithmetic) son operadores de desplazamiento aritmético a izquierda y derecha, respectivamente. Estos operadores desplazan un vector cierto número de bits a la izquierda (SLA) o la derecha (SRA), rellenando los elementos libres con el valor el bit más significativo (MSB). Este tipo de operadores son muy útiles cuando se requiere multiplicar (SLA) o dividir (SRA) un número binario con signo (positivo ó negativo) entre una potencia de 2. Por ejemplo, la expresión: dato SRA 3, desplaza a derecha n bits a la cadena de bits *dato*, es decir, la divide entre $8=2^3$.

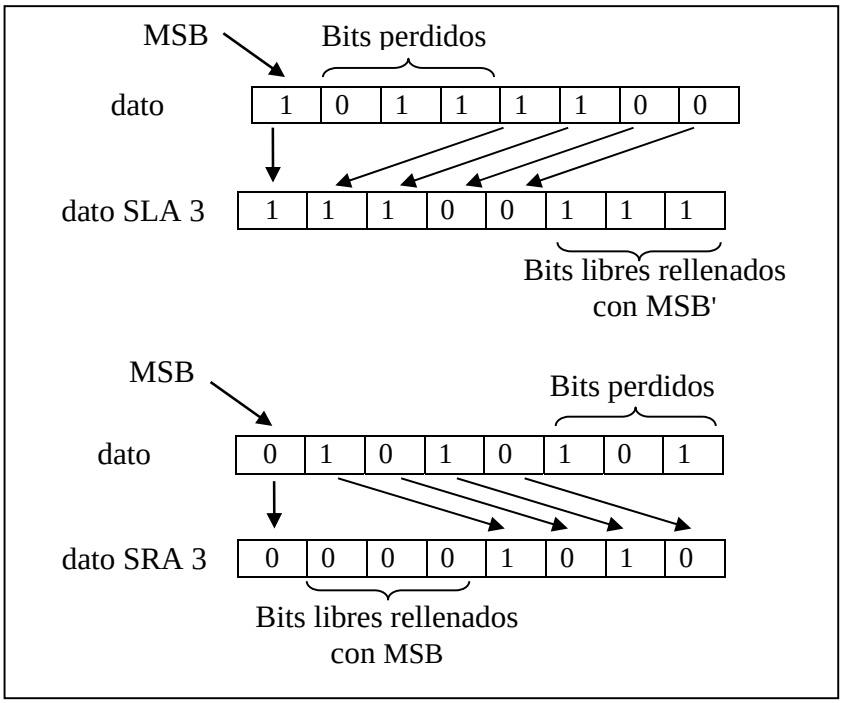


Figura 13.2: Representación de las operaciones de desplazamiento aritmético SLA y SRA.

ROL (Rotate Left), ROR (Rotate Right) son operadores de rotación a izquierda (ROL) y a derecha (ROR), respectivamente. Son como los operadores de desplazamiento lógico, pero los huecos son ocupados por los bits que van quedando fuera.

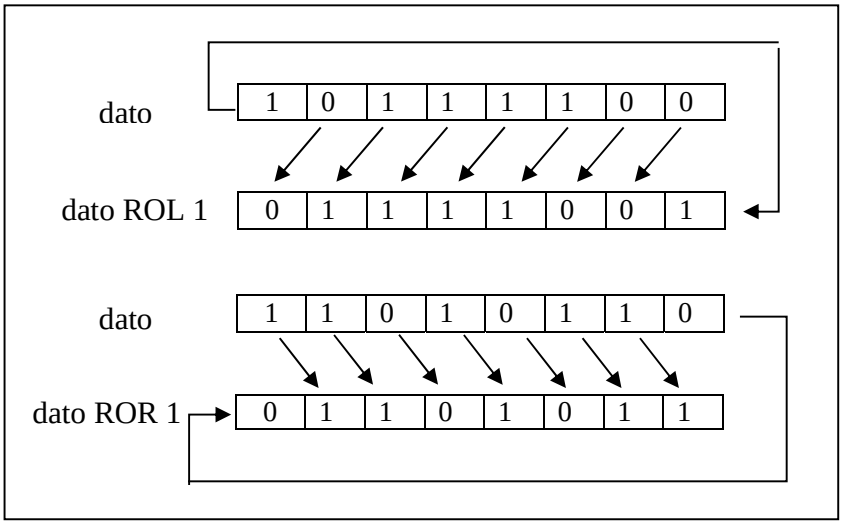


Figura 13.3: Representación de las operaciones de desplazamiento aritmético ROL y ROL.

EJEMPLO

Suponga que dato1="10111100" y dato2="01111001", la siguiente tabla puede aclarar el funcionamiento de los operadores de desplazamiento.

Expresión	Valor de la expresión	Expresión	Valor de la expresión
dato1 SLL 1	"01111000"	dato1 SLL 2	"11110000"
dato2 SLL 1	"11110010"	dato1 SLL 2	"11100100"
dato1 SRL 1	"01011110"	dato1 SRL 3	"00010111"
dato2 SRL 1	"00111100"	dato2 SRL 3	"00001111"
dato1 SLA 1	"11111001"	dato1 SLA 4	"11000000"
dato1 SLA 2	"11110011"	dato2 SLA 4	"10011111"
dato1 SRA 1	"11011110"	dato1 SRA 3	"11110111"
dato2 SRA 1	"00111100"	dato2 SRA 3	"00001111"
dato1 ROL 1	"01111001"	dato1 ROL 2	"11110010"
dato2 ROL 1	"11110010"	dato2 ROL 2	"11100101"
dato1 ROR 1	"01011110"	dato1 ROR 3	"10010111"
dato2 ROR 1	"10111100"	dato2 ROR 3	"00101111"

PROBLEMA 3.1

a) Los datos de un diseño se declaran de la manera siguiente

```
CONSTANT dato1 STD_LOGIC_VECTOR (7 DOWNTO 0):= "01101100";
CONSTANT nibble1 STD_LOGIC_VECTOR (3 DOWNTO 0):= "1001";
CONSTANT nibble2 STD_LOGIC_VECTOR (3 DOWNTO 0):= "0101";
CONSTANT cad1 STRING (1 DOWNTO 5):= "vamos";
CONSTANT cad2 STRING (1 DOWNTO 5):= " a la";
CONSTANT cad3 STRING (1 DOWNTO 6):= " playa";
CONSTANT cad4 STRING (1 DOWNTO 11):= " el domingo";
```

Complete las celdas vacías en la siguiente tabla, en la columna de expresión utilice sólo los datos mencionados en este problema:

Expresión	Valor de la expresión
nibble1 & nibble2	"10010101"
nibble2 & nibble1	
	"011011000101"
	"010110011001"
cad1 & cad2 & cad3 & cad4	"vamos a la playa el domingo"
	"vamos el domingo"
cad1 & cad4 & cad2 & cad3	
cad4 & cad1 & cad2 & cad3	

b) Si byte1= "01101100" y byte2= "11110001", complete la columna de valores para cada expresión en la tabla siguiente:

Expresión	Valor de la expresión
byte1 SLL 1	
byte2 SLL 1	
byte1 SRL 2	
byte2 SRL 2	
byte1 SLA 3	
byte2 SLA 3	
byte1 SRA 4	
byte2 SRA 4	
byte1 ROL 3	
byte2 ROL 3	
byte1 ROR 1	
byte2 ROR 1	

c) Suponga que x='1', z='0', byte1="01010111", byte2="11000010", error=TRUE, inicio=FALSE, complete la columna “Valor de la expresión” en la siguiente tabla:

Expresión	Valor de la expresión
x XOR (NOT z)	
byte1 AND byte2	
byte2 NOR byte1	
(NOT error) OR inicio	
(x AND z) OR ((NOT x) AND z)	

Llame al instructor para registrar el avance. (30 puntos).

OPERADORES ARITMETICOS

Estos aritméticos sólo están definidos para los tipos numéricos (REAL, INTEGER). Los tipos BIT o BIT_VECTOR no son considerados como numéricos, aunque se les use para representar números binarios. En VHDL existen los siguientes operadores aritméticos:

**** (exponencial)**: Sirve para elevar un número a una potencia dada: 4**2 es 4^2=16. El operador de la izquierda puede ser entero o real, pero el de la derecha sólo puede ser entero.

ABS() (valor absoluto): Devuelve el valor absoluto de su argumento que puede ser de cualquier tipo numérico.

*** (multiplicación):** Sirve para multiplicar dos números de cualquier tipo.

/ (división): Divide un dato de tipo numérico entre otro de igual tipo.

MOD (módulo): Calcula en módulo de dos números. Exactamente se define el módulo como la operación que cumple: (x MOD z)=z – *int*(x/z), donde *int*() representa el valor entero (sin decimales). Los operandos sólo pueden ser enteros. El resultado toma el signo de z. Nota: *int*() no pertenece a VHDL, aquí se usa sólo para darle claridad matemática a la definición del operador.

REM (residuo): Calcula el residuo de la división entera y se define como el operador que cumple: x=*int*(x/z)*z+(x REM z). Los operandos sólo pueden ser enteros. El resultado toma el signo de z.

+ (suma y signo positivo): Este operador sirve para indicar suma, si va entre dos operandos (z + x), o signo, si va al principio de una expresión (+z). La precedencia es diferente en cada caso. Opera sobre valores numéricos de cualquier tipo.

– (resta y signo negativo): Cuando va entre dos operandos se realiza la operación de sustracción (z – x), y si va delante de una expresión le cambia el signo (–z). Los operandos pueden ser numéricos de cualquier tipo

EJEMPLO

Suponga que en1=10, en2=3, en3= –2, re1=4.5, re2=1.5, re3= –3.5, la siguiente tabla muestra el funcionamiento de los operadores aritméticos.

Operador	Expresión con enteros	Valor de la expresión	Expresión con reales	Valor de la expresión
**	en1 ** en2	1000	re1 ** en2	91.125
ABS	ABS(en1)	10	ABS(re3)	3.5
*	en1 * en3	–20	re1 * re2	6.75
/	en1 / en2	3	re1 / re3	–1.28571
MOD	en1 MOD en2	1	No aplica	No aplica
REM	en1 REM en2	1	No aplica	No aplica
+ (suma)	en1 + en2	13	re1 + re3	1.0
+ (signo)	+en1;	10	+re3	–3.5
– (resta)	en1 – en2	7	re1 – re2	3.0
– (signo)	–en1;	–10	–re3;	3.5

OPERADORES RELACIONALES

Devuelven siempre un valor de tipo booleano (TRUE o FALSE). Los tipos con los que pueden operar dependen de la operación:

=, /= (igualdad, desigualdad): El primero devuelve TRUE si los operandos son iguales o FALSE en caso contrario. El segundo indica desigualdad, así que funciona de manera inversa. Los operandos pueden ser de cualquier tipo, siempre y cuando ambos sean del mismo tipo y dimensión.

<, <=, >, >= (menor que, menor o igual que, mayor que, mayor o igual que): Tienen el significado normal que en los lenguajes de programación convencionales. La diferencia con los anteriores es que los operandos deben ser de tipo numérico (enteros y reales). Nótese que el símbolo "<=" también se usa para denotar la asignación concurrente, así que para diferenciar las operaciones hay que tomar el contexto en que se usa este símbolo.

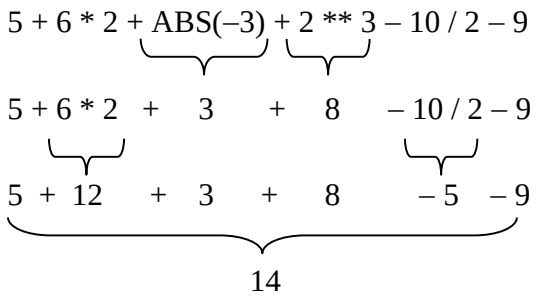
PRECEDENCIA DE OPERADORES

La precedencia de los operadores es el orden en que se ejecutarán, cuando existen varios en una misma expresión, a pesar del orden en que estén escritos; si un operador tiene mayor precedencia que otro, se ejecutará primero, aunque aparezca después del operador de menor precedencia. La precedencia de operadores en VHDL se presenta en la siguiente tabla:

Máxima precedencia	( )					
↓	**	ABS	NOT			
	*	/	MOD	REM		
	+(signo)	–(signo)				
	+(suma)	–(resta)	&			
	=	/=	<	<=	>	>=
Mínima precedencia	AND	OR	NAND	NOR	XOR	XNOR

EJEMPLO

En la siguiente expresión hay varios operadores con diferente precedencia.



Como los operadores ABS() y ** tienen la mayor precedencia entre los demás operadores, éstos se ejecutan primero, después los operadores que siguen en precedencia, la multiplicación y la división, y por último los 3 operadores de de suma y los 2 de resta, que tienen la misma precedencia, dando como resultado de la expresión el número entero 14.

PROBLEMA 3.2:

a) Relacione con una línea los elementos de la columna de expresiones con la columna de valores de las expresiones, tomando en cuenta que valor1=4, valor2=2, valor3=–3, valor4=2.1, valor5=0.1, valor6=–0.2.

EXPRESIONES	**VALORES DE EXPRESIONES**
valor1 ** valor2	7
ABS(valor3)	3
valor2 * valor3	0.7
valor4 / valor3	–6
valor4 + valor5	2.2
valor1 – valor3	16
–valor6;	0.2

b) Suponga que x='1', z='0', dato1=3, dato2=4 y dato3=1, complete la columna “Valor de la expresión” en la siguiente tabla:

Expresión	**Valor de la expresión**
z = (NOT x)	
x = dato3	
dato2 /= (dato1 + 1)	
dato2 < dato1	
(dato2 – 1) <= dato1	
dato2 > dato1	

c) Suponga que cond1=TRUE, cond2=TRUE, cond3=TRUE, dato1=3, dato2=2 y dato3=1, y tomando en cuenta la precedencia de los operadores, complete la columna “Valor de la expresión” en la siguiente tabla:

Expresión	**Valor de la expresión**
dato1 * dato2 + 2	
dato3 + dato1 * dato2	
(dato1 + 1) * 4	
– dato1 * 5 + 2	
6 * (– dato3 – 1)	
dato1 = dato2 OR dato1 = 3	

Llame al instructor para registrar el avance. (30 puntos).

ATRIBUTOS

Los elementos en VHDL, como señales, variables, etc, pueden tener información adicional llamada atributos. Estos atributos están asociados a estos elementos del lenguaje y se manejan en VHDL mediante la comilla simple ('). Hay algunos de estos atributos que están predefinidos en el lenguaje, a continuación se muestran los más utilizados.

dato'LEFT: Límite izquierdo del tipo de *dato*.
dato'RIGHT: Límite derecho del tipo de *dato*.
dato'LOW: Límite inferior del tipo de *dato*.
dato'HIGH: Límite superior del tipo de *dato*.
dato'RANGE: Rango del tipo de *dato*.
dato'LENGTH: Longitud de la dimensión del tipo de *dato*.

Por ejemplo, considere el siguiente código:

```
SIGNAL byte: BIT_VECTOR (7 DOWNTO 0);
SIGNAL cad: STRING (1 TO 11);
```

La siguiente tabla muestra el funcionamiento de los atributos.

Atributo	Valor de la expresión	Atributo	Valor de la expresión
byte'LEFT	7	cad'LEFT	1
byte'RIGHT	0	cad'RIGHT	11
byte'LOW	0	cad'LOW	1
byte'HIGH	7	cad'HIGH	11
byte'RANGE	7 DOWNTO 0	cad'RANGE	1 TO 11
byte'LENGTH	8	cad'LENGTH	11

PROBLEMA 3.3
Considere el siguiente código:

```
SIGNAL frase: STRING (1 TO 11);
SIGNAL reg: STD_LOGIC_VECTOR (0 DOWNTO –15);
```

Complete las columnas “Valor de la expresión” en la siguiente tabla.

Atributo	Valor de la expresión	Atributo	Valor de la expresión
frase'LEFT		reg'LEFT	
frase'RIGHT		reg'RIGHT	
frase'LOW		reg'LOW	
frase'HIGH		reg'HIGH	
frase'RANGE		reg'RANGE	
frase'LENGTH		reg'LENGTH	

Llame al instructor para registrar el avance. (10 puntos).