Module #04 : Verilog HDL Operators

4.1: Arithmetic Operator:

- These perform arithmetic operations.
- The + and - can be used as either unary (-z) or binary (x-y) operators

Operators:

+	*(addition)*
-	*(subtraction)*
*	*(multiplication)*
/	*(division)*
%	*(modulus)*

Example 4.1:

```
c = a + b;
d = a – b;
count = (count + 1) % 16 ;
```

Module #04 : Verilog HDL Operators

4.2: Relational Operator:

- Relational operators compare two operands and return a single bit 1or 0.
- These operators synthesize into comparators.
- Wire and reg variables are positive. Thus (-3’b001) = = 3’b111 and (-3d001)>3’b110.
- However for integers -1< 6

Operators:

<	*(less than)*
<=	*(less than or equal to)*
>	*(greater than)*
>=	*(greater than or equal to)*
==	*(equal to)*
!=	*(not equal to)*

Example 4.2:

```
if (x = = y)
   e = 1;
else
   e = 0;
```

Equivalent Statement -

```
e = (x == y);
```

Module #04 : Verilog HDL Operators

4.3: Bit-wise Operator:

- Bit-wise operators do a bit-by-bit comparison between two operands.

Operators:

~	*(bitwise NOT)*
&	*(bitwise AND)*
|	*(bitwise OR)*
^	*(bitwise XOR)*
~^ *or* ^~	*(bitwise XNOR)*

Example 4.3:

```
module and2(a, b, c);
   input a;
   input b;
   output c;
 assign c = a & b;
endmodule
```

Module #04 : Verilog HDL Operators

4.4: Logical Operator:

- Logical operators return a single bit 1 or 0. They are the same as bit-wise operators only for single bit operands.
- They can work on expressions, integers or groups of bits, and treat all values that are nonzero as “1”.
- Logical operators are typically used in conditional (if ... else) statements since they work with expressions.

Operators:

!	*(logical NOT)*
&&	*(logical AND)*
||	*(logical OR)*

Example 4.4:

```
wire[7:0] x, y, z; // x, y and z are multibit variables.
reg a;
 if ((x == y) && (z))
       a = 1; // a = 1 if x equals y, and z is nonzero.
 else a = !x; // a =0 if x is anything but nonzero
```

Module #04 : Verilog HDL Operators

4.5: Reduction Operator:

- Reduction operators operate on all the bits of an operand vector and return a single-bit value. These are the unary (one argument) form of the bit-wise operators.

Operators:

&	*(reduction AND)*
|	*(reduction OR)*
~&	*(reduction NAND)*
~|	*(reduction NOR)*
^	*(reduction XOR)*
~^ or ^~	*(reduction XNOR)*

Example 4.5:

```
module chk_zero (a, z);
   input [2:0] a;
   output z;
   assign z = ~| a; // Reduction NOR
endmodule
```

Module #04 : Verilog HDL Operators

4.6: Shift Operator:

- Shift operators shift the first operand by the number of bits specified by the second operand. Vacated positions are filled with zeros for both left and right shifts (There is no sign extension).

Operators:

<< (shift left)
>> (shift right)

Example 4.6:

assign c = a << 2; / c = a shifted left 2 bits; vacant positions are filled with 0's */*

Module #04 : Verilog HDL Operators

4.7: Concatenation Operator:

- The concatenation operator combines two or more operands to form a larger vector.

Operators:

{ }(concatenation)

Example 4.7:

```
wire [1:0] a, b;
wire [2:0] x;
wire [3;0] y, Z;
Wire [2:0]count;
assign x = {1'b0, a}; // x[2]=0, x[1]=a[1], x[0]=a[0]
assign y = {a, b}; /* y[3]=a[1], y[2]=a[0], y[1]=b[1], y[0]=b[0] */

assign {count, y} = x + Z; // Concatenation of a result
```

Module #04 : Verilog HDL Operators

4.8: Replication Operator:

- The replication operator makes multiple copies of an item

Operators:

{n{item}} (n fold replication of an item)

Example 4.8:

wire [1:0] a, b; wire [5:0] x, y, z;
assign x = {4{1'b0}, a}; // Equivalent to x = {0,0,0,0,a }
assign y = {2{a}, {b}}; //Equivalent to y = {a, a, b}
assign z = {6{1'b1}};

Module #04 : Verilog HDL Operators

4.9: Conditional Operator: “?”

- Conditional operator is like those in C/C++. They evaluate one of the two expressions based on a condition.
- It will synthesize to a multiplexer (MUX).
- Operated on 3 operands (**Ternary Operator**)

Operators:

(condition) ? (result if condition true): (result if condition false)

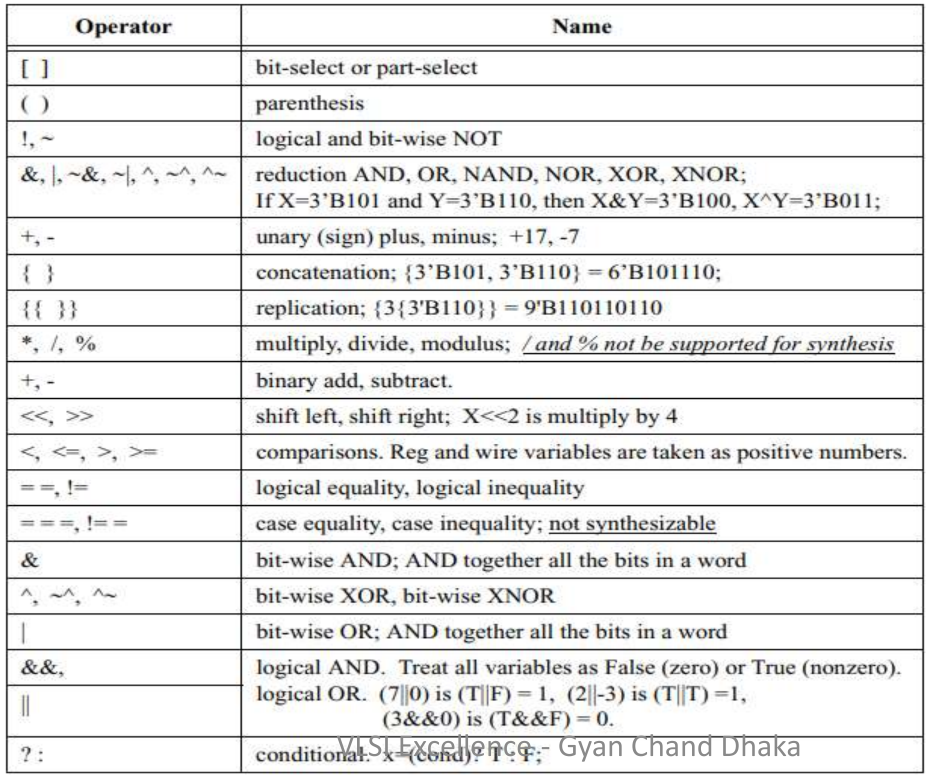
Example 4.9:

assign a = (g) ? x : y;
assign a = (inc = = 2) ? a+1 : a-1;
/ if (inc), a = a+1, else a = a-1 */*

Module #04 : Verilog HDL Operators

4.10: Operator Precedence:

- Table Below shows the precedence of operators from highest to lowest.
- Operators on the same level evaluate from left to right.
- It is strongly recommended to use parentheses to define order of precedence and improve the readability of your code.



Operator	Name
[ ]	bit-select or part-select
( )	parenthesis
!, ~	logical and bit-wise NOT
&, |, ~&, ~|, ^, ~^, ^~	reduction AND, OR, NAND, NOR, XOR, XNOR; If X=3'B101 and Y=3'B110, then X&Y=3'B100, X^Y=3'B011;
+, -	unary (sign) plus, minus; +17, -7
{ }	concatenation; {3'B101, 3'B110} = 6'B101110;
{{ }}	replication; {3{3'B110}} = 9'B110110110
*, /, %	multiply, divide, modulus; *<u>/ and % not be supported for synthesis</u>*
+, -	binary add, subtract.
<<, >>	shift left, shift right; X<<2 is multiply by 4
<, <=, >, >=	comparisons. Reg and wire variables are taken as positive numbers.
= =, !=	logical equality, logical inequality
= = =, != =	case equality, case inequality; <u>not synthesizable</u>
&	bit-wise AND; AND together all the bits in a word
^, ~^, ^~	bit-wise XOR, bit-wise XNOR
|	bit-wise OR; AND together all the bits in a word
&&,	logical AND. Treat all variables as False (zero) or True (nonzero).
||	logical OR. (7||0) is (T||F) = 1, (2||-3) is (T||T) =1, (3&&0) is (T&&F) = 0.
? :	conditional: x=(cond)? T : F;