DSLab. 04 Canonical and Standard Forms

Lab. 04 Canonical and Standard Forms

- Design and Verify the following circuits using Verilog HDL
- Schematic
- Verilog HDL
  - Behavioral level modeling
  - Dataflow modeling
  - Structural level modeling
- Please write and upload the lab report (Lab04) -- Due on 2022/09/30 23:59

Operators in Verilog

Table 8.1
Verilog 2001 HDL Operators

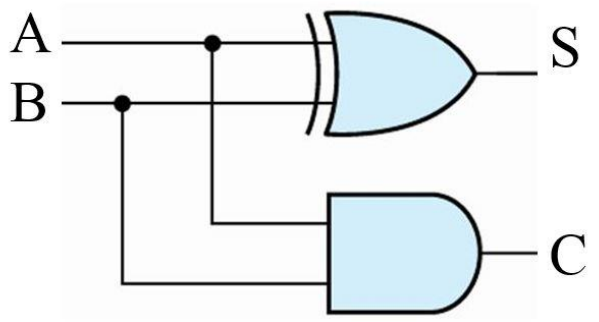
Operator Type	Symbol	Operation Performed
Arithmetic	+	addition
	–	subtraction
	*	multiplication
	/	division
	%	modulus
	**	exponentiation
Bitwise or Reduction	~	negation (complement)
	&	AND
	|	OR
	^	exclusive-OR (XOR)
Logical	!	negation
	&&	AND
	||	OR
Shift	>>	logical right shift
	<<	logical left shift
	>>>	arithmetic right shift
	<<<	arithmetic left shift
	{ , }	concatenation
Relational	>	greater than
	<	less than
	==	equality
	!=	inequality
	===	case equality
	!==	case inequality
	>=	greater than or equal
	<=	less than or equal

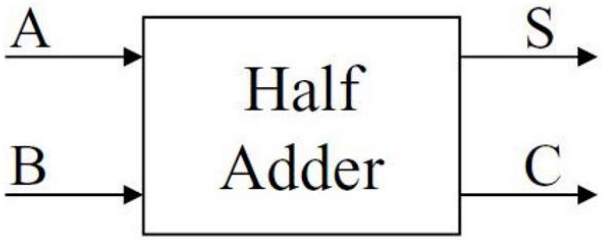
Table 8.2
Verilog Operator Precedence

+ – ! ~ & ~& | ~ | ^ ~^ ^~ (unary)	Highest precedence
**	
* / %	
+ – (binary)	
<< >> <<< >>>	
< < = > >=	
== != === !==	
& (binary)	
^ ^~ ~^ (binary)	
| (binary)	
&&	
||	
?: (conditional operator)	
{ } { { } }	Lowest precedence

Use parentheses to enforce your priority

Example 1: Half Adder (Dataflow modeling)



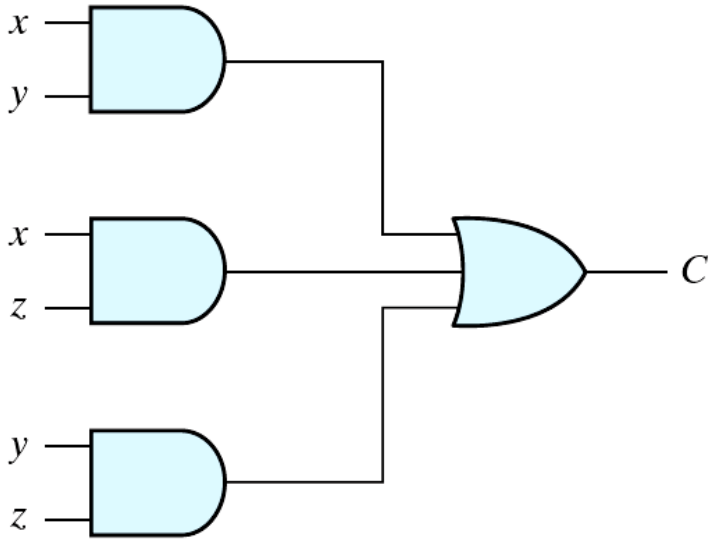


$S = A \oplus B$

$C = AB$

```
module half_adder (S, C, A, B);
    output S, C;
    input A, B;

    assign S = A ^ B;
    assign C = A & B;

endmodule
```

Example 2: Carry of Full Adder (Dataflow modeling)



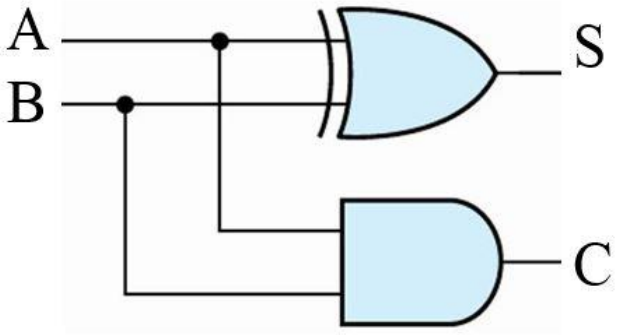
$C = xy + xz + yz$

```
module C_FA (C, x, y ,z);
    output C;
    input x, y, z;

    assign C = x & y | x & z | y & z;

endmodule
```

Verilog Structural Model (Gate Level)

- Built-in gate primitives:
  - **and, nand, nor, or, xor, xnor, buf, not, bufif0, bufif1, notif0, notif1**
- Usage:
  - nand (out, in1, in2); 2-input NAND without delay
  - and #2 (out, in1, in2, in3); 3-input AND with 2 t.u. delay
  - not #1 N1(out, in); NOT with 1 t.u. delay and instance name
  - xor X1(out, in1, in2); 2-input XOR with instance name
- Write them inside module, outside procedures

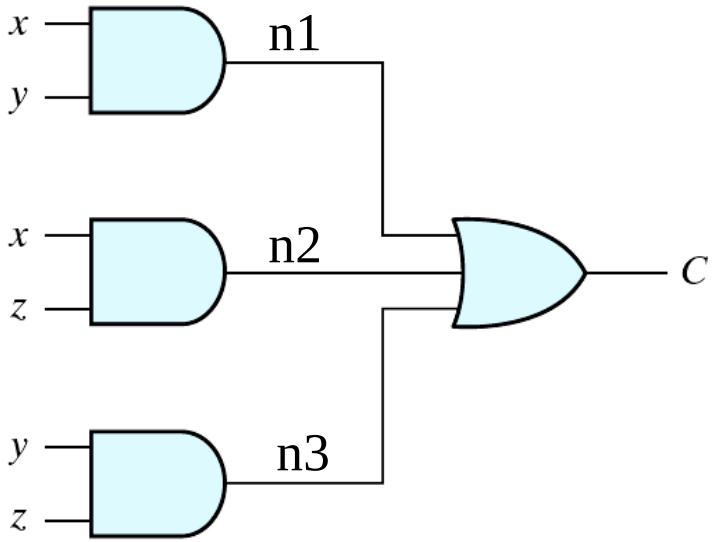
Example 3: Half Adder (Structural level modeling)



Assuming:
- XOR: 2 t.u. delay
- AND: 1 t.u. delay

```
module half_adder (S, C, A, B);
    output S, C;
    input A, B;

    xor #2 (S, A, B);
    and #1 (C, A, B);

endmodule
```

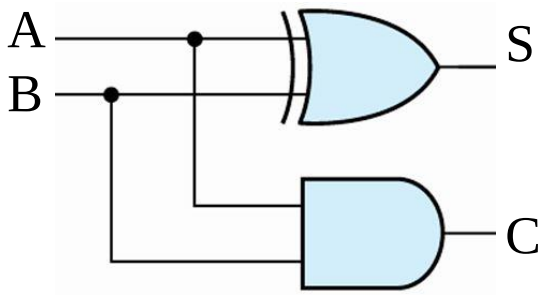
Example 4: Carry of Full Adder (Structural level modeling)



$C = xy + xz + yz$

```
module C_FA(C, x, y ,z);
    output C;
    input x, y, z;
    wire n1, n2, n3;

    and (n1, x, y);
    and (n2, x, z);
    and (n3, y, z);
    or  (C, n1, n2, n3);

endmodule
```

Testbench




```
module t_half_adder;
  reg A;
  reg B;
  wire S;
  wire C;

  half_adder UUT ( .A(A), .B(B), .S(S), .C(C) );

  initial begin
     A=1'b0; B=1'b0;
     #10  A=1'b0; B=1'b1;
     #10  A=1'b1; B=1'b0;
     #10  A=1'b1; B=1'b1;
     #10  $finish;
  end
endmodule
```

Exercise 1

- *F1* = (*x*+*z*)(*y'z'*+*yz*)
- Find and show the complement of *F1* (*F2*) using DeMorgan's theorem
- Verify *F1'* = *F2* (using Dataflow modeling)

Exercise 2

- Find and show the sum of minterms (*F3*) of the following truth table
- Find and show the product of maxterms (*F4*) of the following truth table
- Verify *F3* = *F4*

  (using Structural level modeling)

w	*x*	*y*	*z*	*F*
0	0	0	0	0
0	0	0	1	1
0	0	1	0	1
0	0	1	1	1
0	1	0	0	0
0	1	0	1	0
0	1	1	0	1
0	1	1	1	0
1	0	0	0	0
1	0	0	1	0
1	0	1	0	0
1	0	1	1	1
1	1	0	0	1
1	1	0	1	0
1	1	1	0	1
1	1	1	1	1

Exercise 3

- $F5(x, y, z) = x+z'$
- Find and show the canonical forms of *F5*, i.e., the sum of minterms (*F6*) and the product of maxterms (*F7*)
- Verify *F5* = *F6* = *F7*

  (using Structural level modeling)