PUID: ____________________

Rule	**Expression**
Commutativity	$X + Y = Y + X$ $X \cdot Y = Y \cdot X$
Associativity	$(X + Y) + Z = X + (Y + Z)$ $(X \cdot Y) \cdot Z = X \cdot (Y \cdot Z)$
Distributivity	$X \cdot Y + X \cdot Z = X \cdot (Y + Z)$ $(X + Y) \cdot (X + Z) = X + Y \cdot Z$
Covering	$X + X \cdot Y = X$ $X \cdot (X + Y) = X$
Combining	$X \cdot Y + X \cdot Y = X$ $(X + Y) \cdot (X + Y) = X$
Consensus	$X \cdot Y + X \cdot Z + Y \cdot Z = X \cdot Y + X' \cdot Z$ $(X + Y) \cdot (X + Z) \cdot (Y + Z) = (X + Y) \cdot (X + Z)$
Generalized Idempotency	$X + X + \cdots + X = X$ $X \cdot X \cdot \cdots \cdot X = X$
DeMorgan's Theorems	$(X_1 \cdot X_2 \cdot \cdots \cdot X_n)' = X_1' + X_2' + \cdots + X_n'$ $(X_1 + X_2 + \cdots + X_n)' = X_1' \cdot X_2' \cdot \cdots \cdot X_n'$
Generalized DeMorgan's	$F(X_1, X_2, \ldots, X_n, +, \cdot) = F(X_1, X_2, \ldots, X_n, \cdot, +)'$
Shannon's Expansion	$F(X_1, X_2, \ldots, X_n) = X_1 \cdot F(1, X_2, \ldots, X_n) + X_1' \cdot F(0, X_2, \ldots, X_n)$ $F(X_1, X_2, \ldots, X_n) = [X_1 + F(0, X_2, \ldots, X_n)] \cdot [X_1' + F(1, X_2, \ldots, X_n)]$
Dual	Interchange + with ·, and 0 with 1

```
//     !  : Logical NOT, ~ : Bitwise NOT
//     && : Logical AND, & : Bitwise AND
//     || : Logical OR,  | : Bitwise OR
//     ^  : Bitwise XOR
//     ~^ : Bitwise XNOR
//     == : Equality,    === : Case Equality (4-state)
//     != : Inequality,  !== : Case Inequality (4-state)
```

```
module my_module (
  input  logic clk,   // clock
  input  logic rst,   // reset
  output logic out
);
  // Module internals here
endmodule
```

```
// Sequential (synchronous) logic
always_ff @(posedge clk or posedge rst) begin
  if (rst)
    out <= 0;
  else
    out <= ~out;
end

// Combinational logic
always_comb begin
  // assignments that depend solely on input combinatorics
end
```

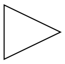
```
interface simple_if (input logic clk);
  logic data;
  modport master (output data);
  modport slave  (input  data);
endinterface
```

```
property p_reset;
  @(posedge clk) disable iff (rst) (a |-> b);
endproperty

assert property(p_reset);
```

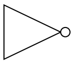
Buffer

A	Output
0	0
1	1



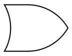
NOT

A	Output
0	1
1	0



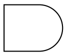
OR

A	B	Output
0	0	0
0	1	1
1	0	1
1	1	1



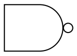
AND

A	B	Output
0	0	0
0	1	0
1	0	0
1	1	1



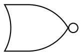
NAND

A	B	Output
0	0	1
0	1	1
1	0	1
1	1	0



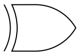
NOR

A	B	Output
0	0	1
0	1	0
1	0	0
1	1	0



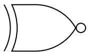
XOR

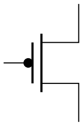
A	B	Output
0	0	0
0	1	1
1	0	1
1	1	0



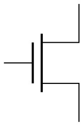
XNOR

A	B	Output
0	0	1
0	1	0
1	0	0
1	1	1





PMOS



NMOS

Parameter	**PMOS**	**NMOS**
Substrate Type	n-type	p-type
Threshold Voltage	Negative	Positive