BEHAVIORAL MODELING

VERILOG DESIGN STYLE

prepared by:

Gyro A. Madrona
Electronics Engineer

TOPIC OUTLINE

Logical Operators

Reduction Operators

Behavioral-Modeling

LOGICAL OPERATORS

LOGICAL EQUALITY (==)

```
reg a,b,c;

reg result;

initial begin

    a = 1;

    b = 1;

    c = 0;

    result = (a == b); // returns TRUE

    result = (b == c); // returns FALSE

end
```

```
reg [2:0] w,x,y;

reg result;

initial begin

    w = 3'b101;

    x = 3'b101;

    y = 3'b110;

    result = (w == x); // returns TRUE

    result = (x == y); // returns FALSE

end
```

LOGICAL INEQUALITY (! =)

```
reg a,b,c;

reg result;

initial begin

    a = 1;

    b = 1;

    c = 0;

    result = (a != b); // returns FALSE

    result = (b != c); // returns TRUE

end
```

```
reg [2:0] w,x,y;

reg result;

initial begin

    w = 3'b101;

    x = 3'b101;

    y = 3'b110;

    result = (w != x); // returns FALSE

    result = (x != y); // returns TRUE

end
```

LOGICAL AND (&&)

```
reg a,b,c;

reg result;

initial begin

    a = 1;

    b = 1;

    c = 0;

    result = (a && b); // returns TRUE

    result = (b && c); // returns FALSE

end
```

```
reg [2:0] w,x,y;

reg result;

initial begin

    w = 3'b000;

    x = 3'b101;

    y = 3'b110;

    result = (w && x); // returns FALSE

    result = (x && y); // returns TRUE

end
```

LOGICAL OR (||)

```
reg a,b,c;

reg result;

initial begin

    a = 1;

    b = 0;

    c = 0;

    result = (a || b); // returns TRUE

    result = (b || c); // returns FALSE

end
```

```
reg [2:0] w,x,y;

reg result;

initial begin

    w = 3'b000;

    x = 3'b101;

    y = 3'b000;

    result = (w || x); // returns TRUE

    result = (w || y); // returns FALSE

end
```

LOGICAL NOT (!)

```
reg a,b;

reg result;

initial begin

    a = 1;

    b = 0;

    result = (!a); // returns FALSE

    result = (!b); // returns TRUE

end
```

```
reg [2:0] w,x;

reg result;

initial begin

    w = 3'b000;

    x = 3'b101;

    result = (!w); // returns TRUE

    result = (!x); // returns FALSE

end
```

REDUCTION OPERATORS

REDUCTION AND (&_)

```
reg [2:0] w,x;

reg result;

initial begin

    w = 3'b110;

    x = 3'b111;

    result = (&w); // returns FALSE

    result = (&x); // returns TRUE

end
```

```
reg [2:0] w,x;

reg result;

initial begin

    w = 3'b110;

    x = 3'b111;

    result = (~&w); // returns TRUE

    result = (~&x); // returns FALSE

end
```

REDUCTION OR (|_)

```
reg [2:0] w,x;

reg result;

initial begin

    w = 3'b000;

    x = 3'b100;

    result = (|w); // returns FALSE

    result = (|x); // returns TRUE

end
```

```
reg [2:0] w,x;

reg result;

initial begin

    w = 3'b000;

    x = 3'b100;

    result = (~|w); // returns TRUE

    result = (~|x); // returns FALSE

end
```

REDUCTION XOR (^_)

```
reg [2:0] w,x;

reg result;

initial begin

    w = 3'b111;

    x = 3'b101;

    result = (^w); // returns TRUE

    result = (^x); // returns FALSE

end
```

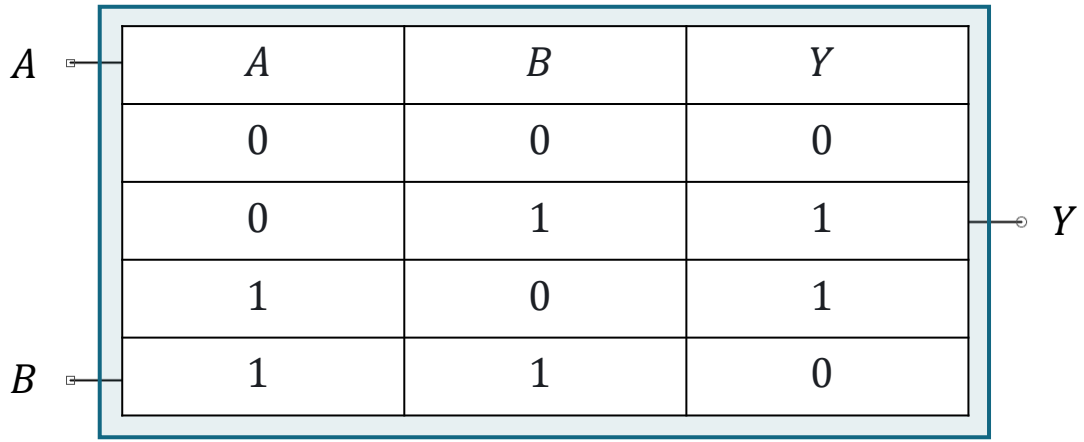
```
reg [2:0] w,x;

reg result;

initial begin

    w = 3'b111;

    x = 3'b101;

    result = (~^w); // returns FALSE

    result = (~^x); // returns TRUE

end
```

BEHAVIORAL_MODELING_

XOR GATE MODULE

module name

xor_gate



A	B	Y
0	0	0
0	1	1
1	0	1
1	1	0

```
module xor_gate(Y,A,B);
    input A,B;
    output reg Y;

    always @(A or B) begin
      if(A == B)
        Y = 0;
      else
        Y = 1;
    end
endmodule
```

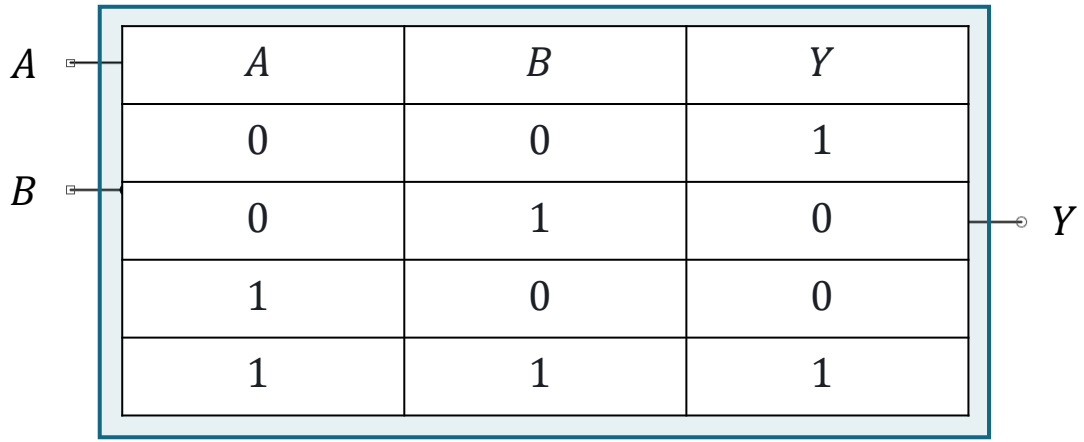
–sensitivity list

Behavioral modeling describes a circuit by specifying its behavior using **procedural statements**.

XNOR GATE MODULE

module name

xnor_gate



A	B	Y
0	0	1
0	1	0
1	0	0
1	1	1

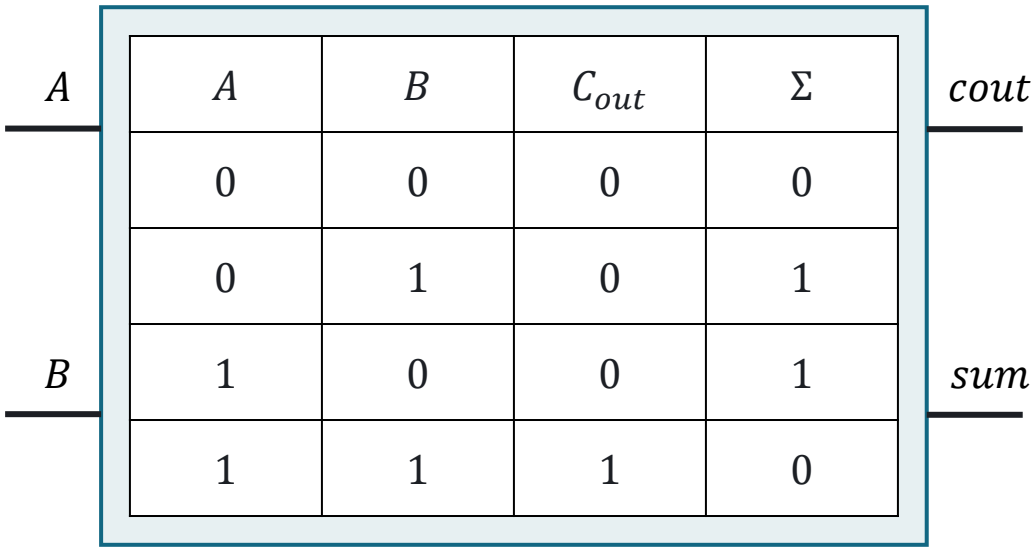
```
module xnor_gate(Y,A,B);
    input A,B;
    output reg Y;

    always @(A,B) begin
      if(A == B)
        Y = 1;
      else
        Y = 0;
    end
endmodule
```

Behavioral modeling describes a circuit by specifying its behavior using **procedural statements**.

HALF-ADDER MODULE

module name

half_adder



A	B	C_{out}	Σ
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

```
// Verilog code
```

FULL-ADDER MODULE

module name

full_adder

A	B	C_{in}	C_{out}	Σ
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

```
// Verilog code
```

LABORATORY