DATAFLOW MODELING

VERILOG DESIGN STYLE

prepared by:

Gyro A. Madrona

Electronics Engineer

TOPIC OUTLINE

Bitwise Operator

Dataflow Modeling

BITWISE OPERATORS

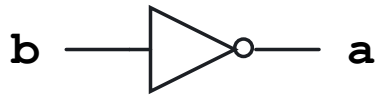
BITWISE NOT

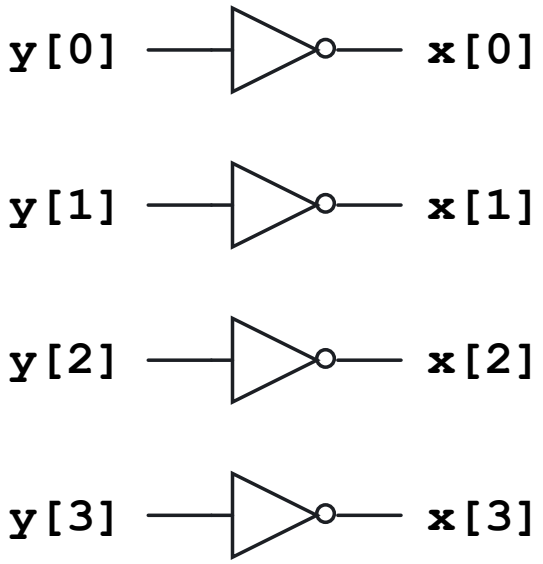
```
wire a,b;

wire [3:0] x,y;
```

not (~)

```
    a = ~b;

    x = ~y;
```

implementation





BITWISE OR

```
wire a,b,c;

wire [3:0] w,x,y;
```

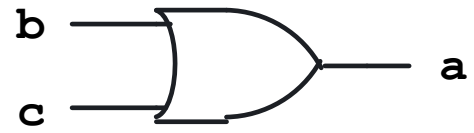
or (|)

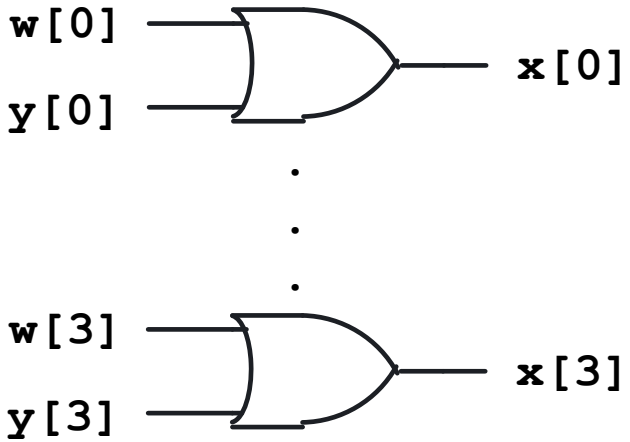
```
a = b | c;

x = w | y;
```

Set bits

```
        w 0010
(mask)  y 0101      1-set
        ----
        x 0111
```

implementation





BITWISE AND

```
wire a,b,c;

wire [3:0] w,x,y;
```

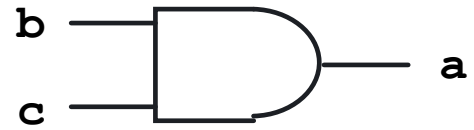
and (&)

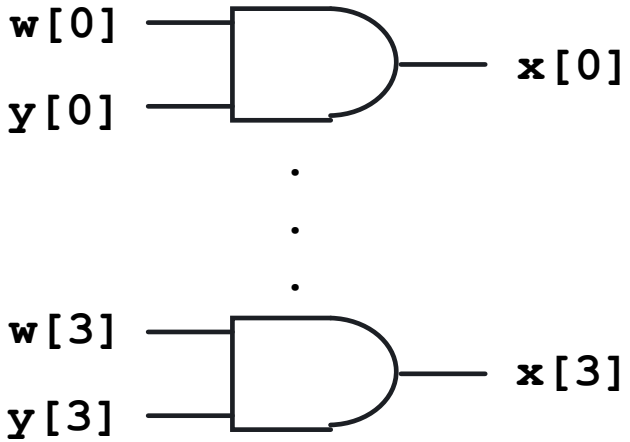
```
a = b & c;

x = w & y;
```

Reset bits

```
      w 1101
(mask) y 0110      0-reset
      ------
      x 0100
```

implementation





BITWISE XOR

```
wire a,b,c;

wire [3:0] w,x,y;
```

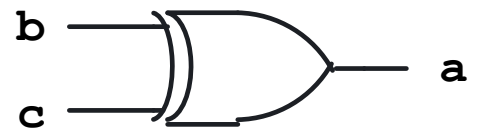
xor (^)

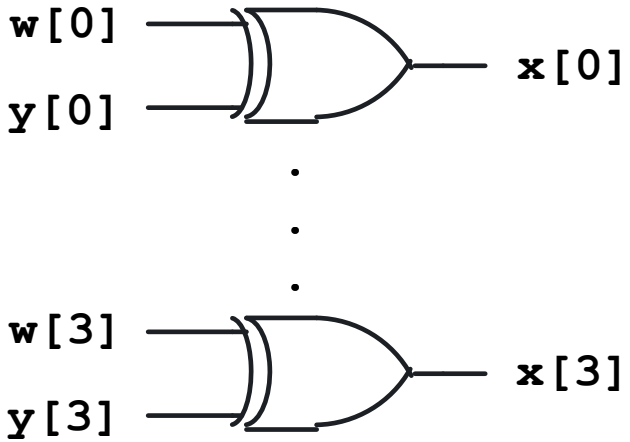
```
a = b ^ c;

x = w ^ y;
```

Flip bits

```
      w 0101
(mask) y 0110    1-flip
      ------
      x 0011
```

implementation





BITWISE NOR

```
wire a,b,c;

wire [3:0] w,x,y;
```

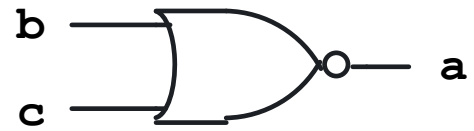
nor (~|)

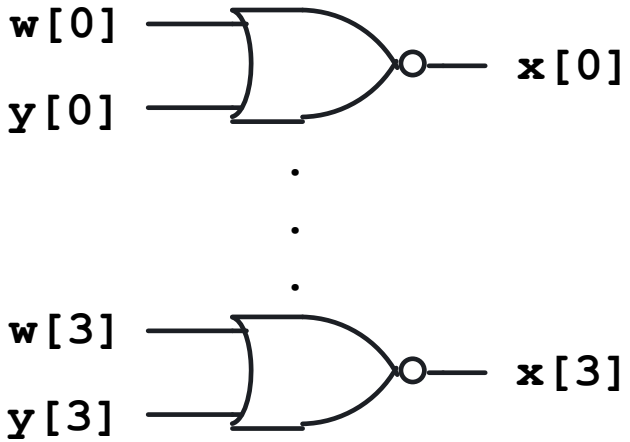
```
a = b ~| c;

x = w ~| y;
```

Reset and flip bits

```
      w 0101      0-flip
(mask) y 0110     1-reset
      ---------
      x 1001
```

implementation





BITWISE NAND

```
wire a,b,c;

wire [3:0] w,x,y;
```

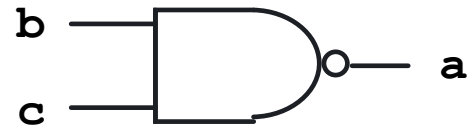
nand (~&)

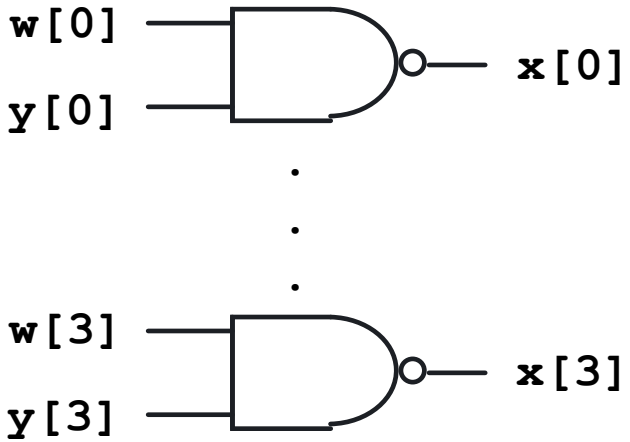
```
a = b ~& c;

x = w ~& y;
```

Set and flip bits

```
        w 0101      0-set
(mask)  y 1001      1-flip
        ----
        x 1110
```

implementation





BITWISE XNOR

```
wire a,b,c;

wire [3:0] w,x,y;
```

xnor (~^)

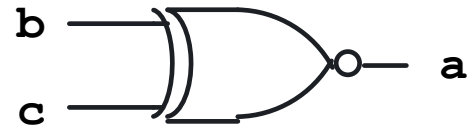
```
a = b ~^ c;

x = w ~^ y;
```

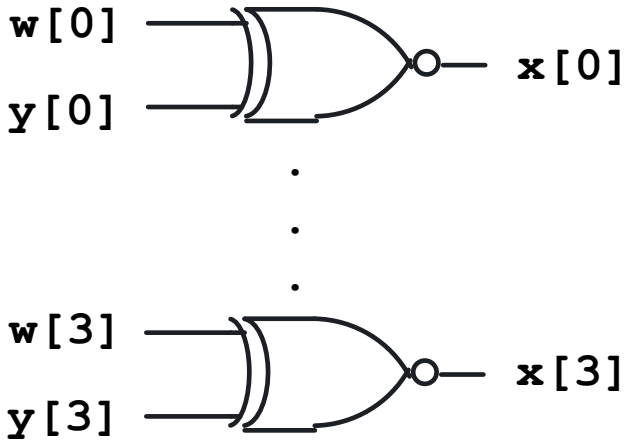
Equality check

```
      w 0101
(mask) y 0100
-------------
      x 1110
```

1-equal

implementation



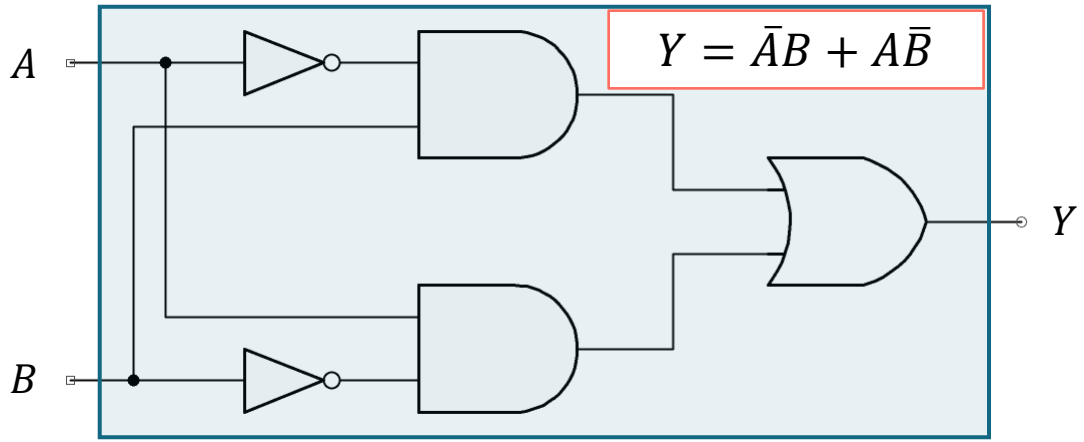


DATAFLOW MODELING

XOR MODULE

module name

xor_gate



Dataflow modeling describes a circuit using continuous assignments and requires knowing the **logic expression** of the design.

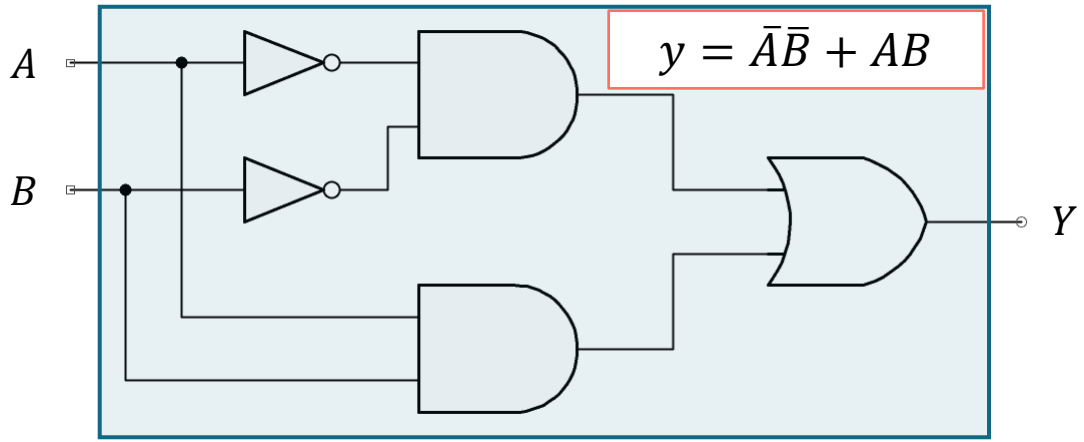
```
module xor_gate(Y,A,B);
    input A,B;
    output Y;

    assign Y = (~A & B)|(A & ~B);
endmodule
```

The **assign** keyword is used for continuous assignments – it continuously drives a value onto a net based on an expression.

XNOR MODULE

module name

xnor_gate




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

    assign Y = (~A & ~B)|(A & B);
endmodule
```

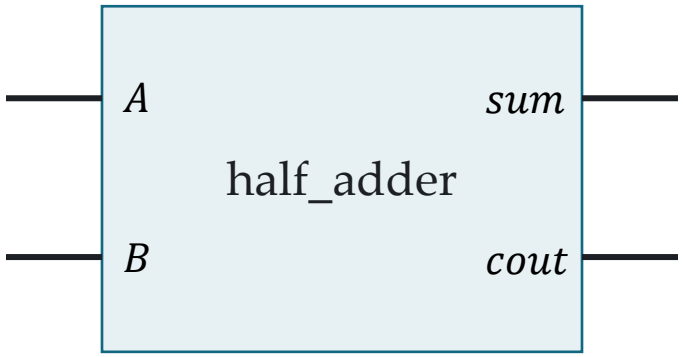
Dataflow modeling describes a circuit using continuous assignments and requires knowing the **logic expression** of the design.

HALF-ADDER MODULE

module name

half_adder

```
// Verilog code
```



$$cout = AB$$

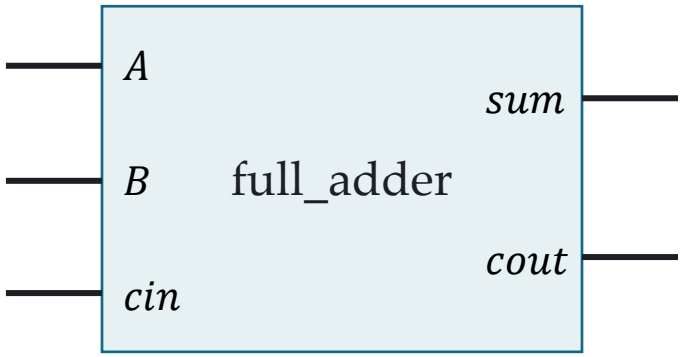
$$sum = A \oplus B$$

FULL-ADDER MODULE

module name

half_adder

```
// Verilog code
```



$$cout = cin(A \oplus B) + AB$$

$$sum = cin \oplus A \oplus B$$

LABORATORY