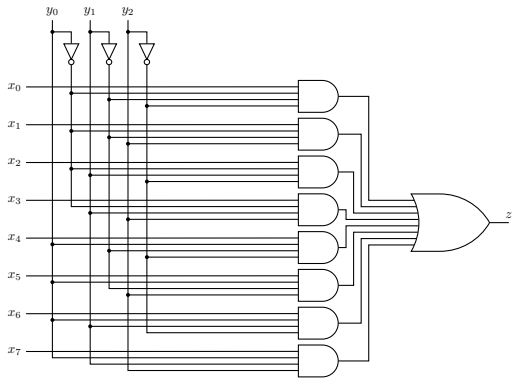
Multiplexer

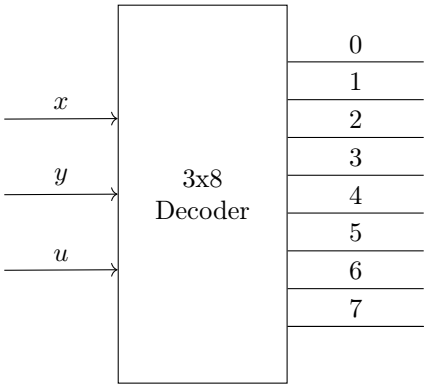
Multiplexers select one of 2^n input lines based on the binary representation of the n selection lines. For example, this 3-8 multiplexer selects the input x_i corresponding to the binary number $y_2y_1y_0$.



Demultiplexers perform the inverse, accepting 2^n input lines and selecting one for output based on the binary representation of the n selection lines.

Decoder

Decoders convert an input binary signal to an output signal.



Encoders perform the inverse, converting the index of the input signal to a binary number. Since encoders may exhibit unwanted behavior if more than one input signals are asserted at a time, *priority* encoders will output the binary representation of the highest-priority input signal for some notion of priority.

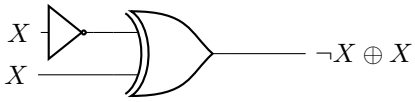
Static Hazards

Static hazards occur in combinational logic circuits when a single input change causes a momentary fluctuation in the output, even though the final output should remain constant. There are two types of static hazards:

Static-1 Occurs when the output is supposed to remain at logic 1 but momentarily drops to 0.

Static-0 Occurs when the output is supposed to remain at logic 0 but momentarily rises to 1.

For example, consider the following circuit diagram for $\neg X \oplus X$:



In this circuit, X is the input and $\neg X$ is the negated input, and $\neg X \oplus X$ is the output of the XOR gate. The output will always be 1, except for when the propogation delay from the NOT gate cause a momentary fluctuation.

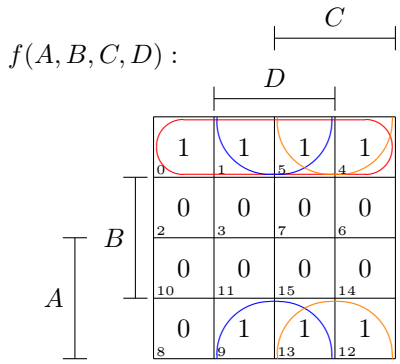
Karnaugh Map

To find the PoS and SoP representations:

SoP Group the 1s in the K-map into the largest possible groups of 1, 2, 4, 8, etc. Each group represents a product term.

PoS Group the 0s in the K-map into the largest possible groups of 1, 2, 4, 8, etc. Each group represents a sum term.

To detect static hazards from the K-map, look for adjacent cells with the same value but covered by different groups.



$$\neg A\neg B + \neg BD + \neg BC \tag{1}$$

SystemVerilog Modeling

Structural modeling describes the circuit in terms of its components and their interconnections using logic gates. This is the most concrete form of modeling.

Listing 1: Structural Modeling Example

```
module and_gate (output wire Y, input wire A, B);
    and (Y, A, B);
endmodule

module top;
    wire A, B, Y;
    and_gate u1 (.Y(Y), .A(A), .B(B));
endmodule
```

Dataflow modeling describes the circuit in terms of the flow of data using continuous assignments. It uses operators to define the relationships between inputs and outputs.

Listing 2: Dataflow Modeling Example

```
module and_gate (output wire Y, input wire A, B);
    assign Y = A & B;
endmodule

module top;
    wire A, B, Y;
    and_gate u1 (.Y(Y), .A(A), .B(B));
endmodule
```

Behavioral modeling describes the circuit in terms of its behavior using procedural blocks. It uses constructs like ‘always’, ‘if’, and ‘?’ to define how the output changes in response to changes in the input. This is the most abstract form of modeling.

Listing 3: Behavioral Modeling Example

```
module and_gate (output reg Y, input wire A, B);
    always @ (A or B) begin
        Y = A & B;
    end
endmodule

module top;
    wire A, B;
    reg Y;
    and_gate u1 (.Y(Y), .A(A), .B(B));
endmodule
```