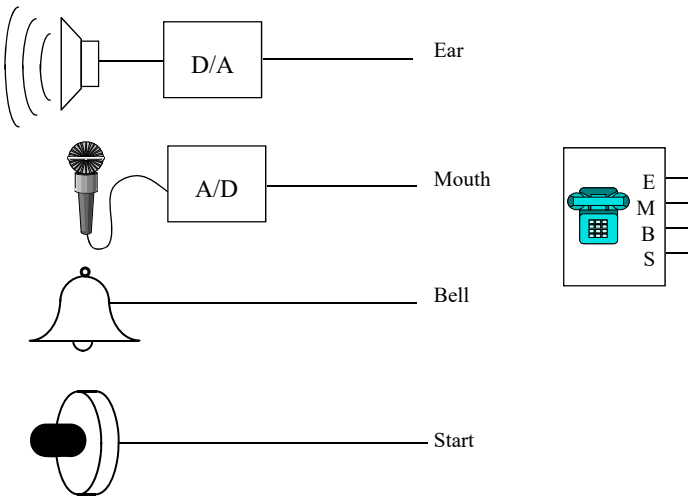
Basic Circuit Elements

- Readings: 4-4.1.1, 4.2, 4.3-4.3.2
- Standard TTL Small-Scale Integration:
  1 chip = 2-8 gates
  - Requires numerous chips to build interesting circuits
- Alternative: Complex chips for standard functions
  - Single chip that performs very complex computations
- Multiplexer/Decoder/Encoder: Standard routing elements for interconnections
- FPGAs: Programmable for arbitrary functions

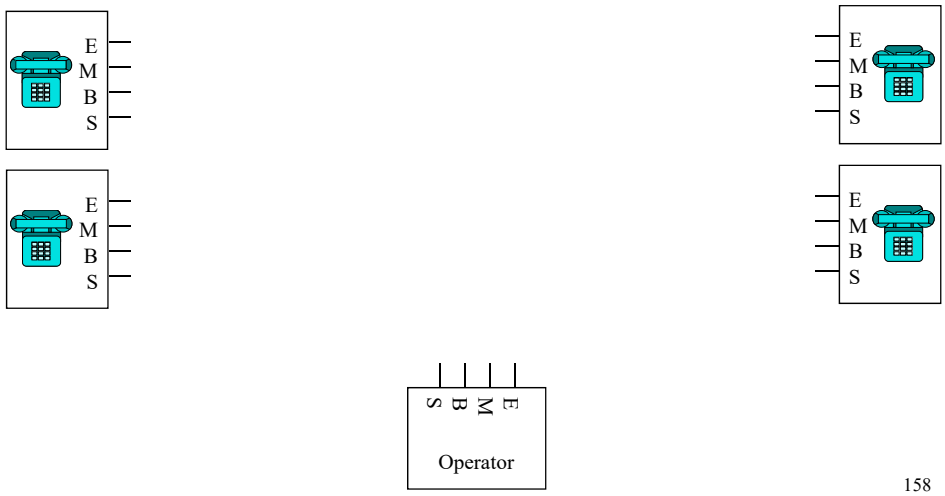
Design Example: Basic Telephone System

- Put together a simple telephone system



Basic Telephone System (cont.)

- Multiple subscribers, one operator.
  - Operator controls all connections

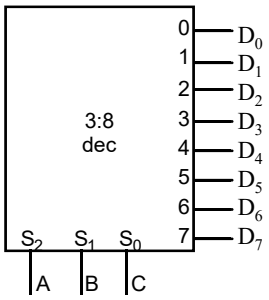


Standard Circuit Elements

- Develop implementations of important "Building Blocks"
  - Used in Networks, Computers, Stereos, etc.
- Multiplexer: Combine N sources onto 1 wire
- Encoder: Determine which input is active
- Decoder: Convert binary to one-of-N wires

Decoders

- Used to select one of 2^N outputs based on N input bits
- Input: N bits; output: 2^N outputs -- only *one* is true
- A decoder that has *n* inputs and *m* outputs is referred to as an *n* x *m*, *N:M*, or *n-to-m* decoder
- Example: 3-to-8 decoder



Decoder Implementation

S1	S0	D3	D2	D1	D0
0	0				
0	1				
1	0				
1	1				

Enabled Decoder Implementation

- Active High enable

En	S1	S0	D3	D2	D1	D0
0	0	0				
0	0	1				
0	1	0				
0	1	1				
1	0	0				
1	0	1				
1	1	0				
1	1	1				

Enabled Decoders in Verilog

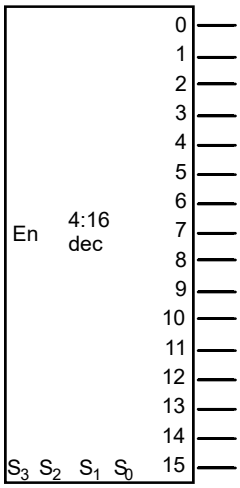
```
module enDecoder2_4 (out, in, enable);
  output  logic  [3:0]  out;
  input   logic  [1:0]  in;
  input   logic         enable;

  always_comb begin

endmodule
```

Decoder Expansion

- Construct a 4:16 decoder using 2:4 decoders

Decoders in General Logic Implementation

- Implement F = WXZ + Y$\overline{Z}$ w/4x16 Decoder



Encoders

- Performs the inverse operation of decoders
- Input: 2^N or less lines -- only 1 is asserted at any given time
- Output: N output lines
- Function: the output is the binary representation of the ID of the input line that is asserted

Encoder Implementation

- 4:2 Encoder

D3	D2	D1	D0	A1	A0
0	0	0	1	0	0
0	0	1	0	0	1
0	1	0	0	1	0
1	0	0	0	1	1

Priority Encoder

- Use priorities to resolve the problem of 2 or more input lines active at a time.
- One scheme: Highest ID active wins
- Also add an output to identify when at least 1 input active

D3	D2	D1	D0	A1	A0	Valid
0	0	0	0			
0	0	0	1			
0	0	1	X			
0	1	X	X			
1	X	X	X			

Priority Encoder Implementation

Priority Encoder Implementation (cont.)

Basic Encoders in Verilog

```
module basicEncoder4_2 (out, in);
  output logic  [1:0]  out;
  input  logic  [3:0]  in;

  always_comb begin
    assert(in == 4'b0001 || in == 4'b0010
           || in == 4'b0100 || in == 4'b1000);
  end

  always_comb begin









endmodule
```

Priority Encoders in Verilog

```
module priorityEncoder4_2 (out, in, valid);
  output logic  [1:0]  out;
  input  logic  [3:0]  in;
  output logic         valid;

  always_comb begin









endmodule
```


Multiplexer

- An element that selects data from one of many input lines and directs it to a single output line
- Input: 2^N input lines and N selection lines
- Output: the data from *one* selected input line
- Multiplexer often abbreviated as MUX

Multiplexer Implementation

- 4:1 MUX

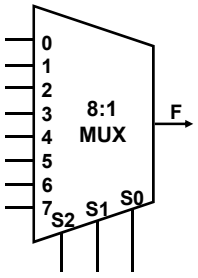
S1	S0	F
0	0	D0
0	1	D1
1	0	D2
1	1	D3

Multiplexer Expansion

- Construct a 16:1 MUX using 4:1 MUX's

Multiplexers in General Logic

- Implement $F = X\overline{Y}Z + Y\overline{Z}$ with a 8:1 MUX



Multiplexers in General Logic (cont.)

- Implement $F = X\overline{Y}Z + Y\overline{Z}$ with a 4:1 MUX