Introduction to VHDL

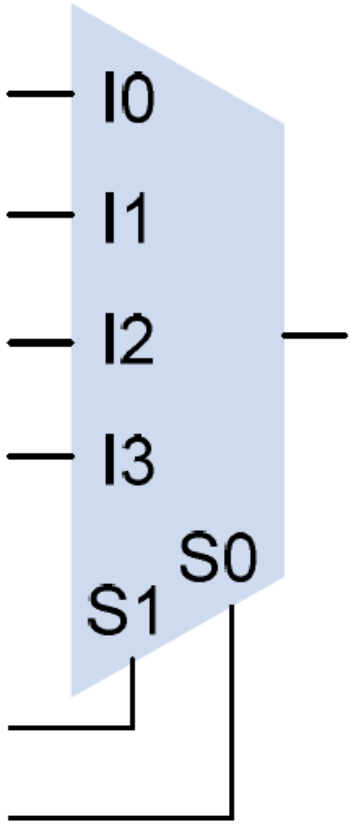
Modules #6 and #7

Digilent Inc. Course

Data Selectors / Multiplexers

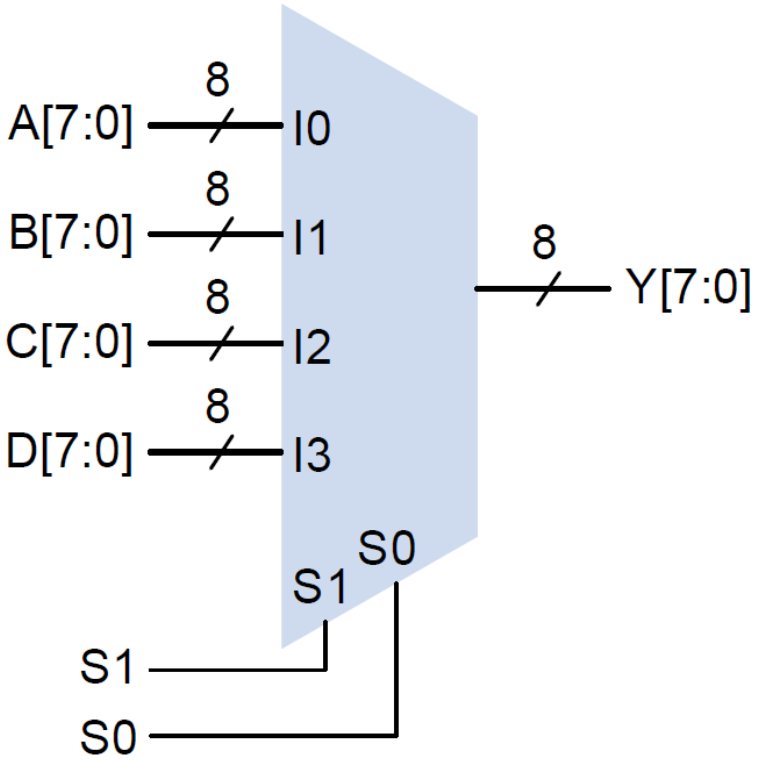
S1	S0	Y
0	0	I0
0	1	I1
1	0	I2
1	1	I3

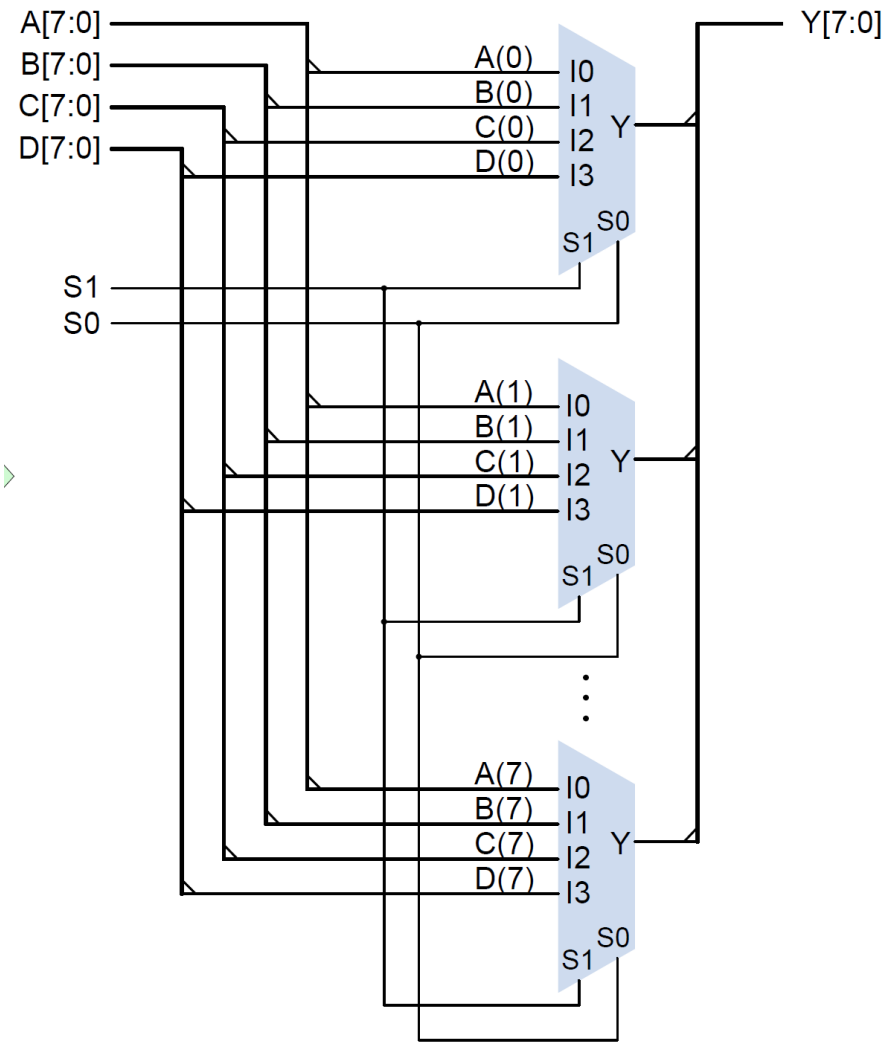
4:1 mux truth table



Mux circuit symbol

Bus Multiplexer





MUX VHDL Example: Selected Signal Assignment

```
entity mux_select is
  port ( I3, I2, I1, I0: in std_logic;
         sel   : in std_logic_vector (1 downto 0);
         Y     : out std_logic);
end mux_select;

architecture behavioral of mux_select is
Begin
 with sel select
  Y <= I0 when “00”;
       I1 when “01”;
       I2 when “10”;
       I3 when “others;
end behavioral;
```

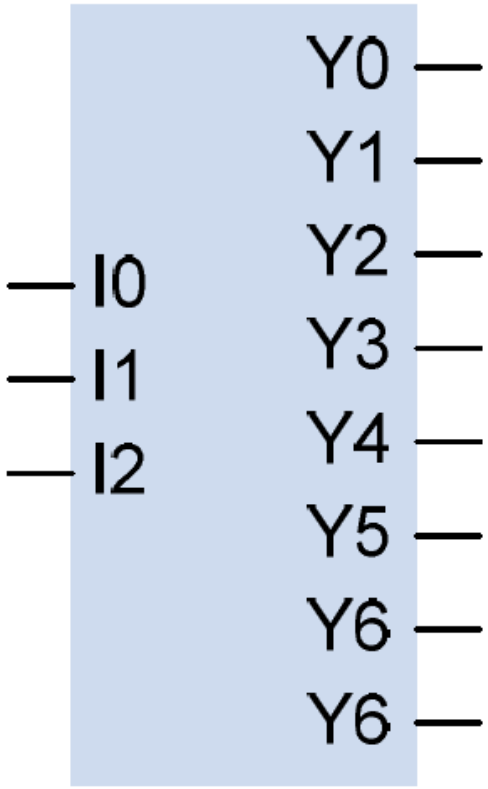
Bus MUX VHDL Example: Selected Signal Assignment

```
entity busmux_select is
  port ( I3, I2, I1, I0: in std_logic_vector (7 downto 0);
         sel       : in std_logic_vector (1 downto 0);
         Y         : outstd_logic_vector (7 downto 0));
end busmux_select;

architecture behavioral of busmux_select is
Begin
 with sel select
  Y <= I0 when “00”;
       I1 when “01”;
       I2 when “10”;
       I3 when “others;
end behavioral;
```

More Complex MUX VHDL: Conditional Assignment

```
entity mux_cond is
   port ( A, B, C  : in std_logic_vector (7 downto 0);
          Sel      : in std_logic_vector (1 downto 0);
          Y        : outstd_logic_vector (7 downto 0));
end mux_cond;

architecture behavioral of mux_cond is
Begin
  Y <=   (A or not C) when  (Sel = "00") else
         (A xor B) when     (Sel = "01") else
         not A when         (Sel = "10") else
         (B nand C);
end behavioral;
```

Decoder

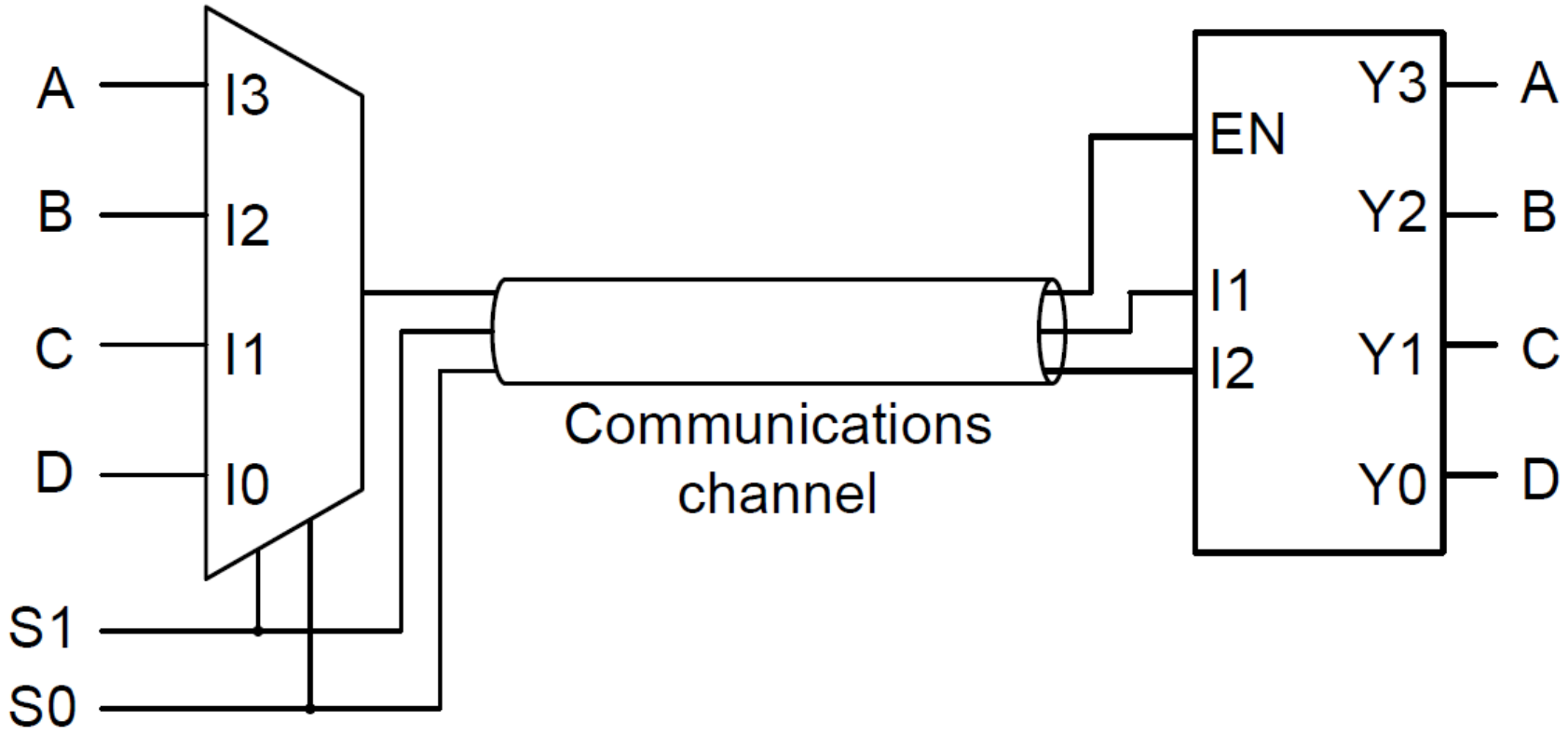


3:8 binary decoder

Decoder VHDL Example: Selected Signal Assignment

```
entity decoder is
   port ( in:        in std_logic_vector (1 downto 0);
          Y:         out std_logic_vector (7 downto 0));
end decoder;

architecture behavioral of decoder is
Begin
  with in select
  Y <=   "0001" when "00";
         "0010" when "01";
         "0100" when "10";
         "1000" when "others;
end behavioral;
```

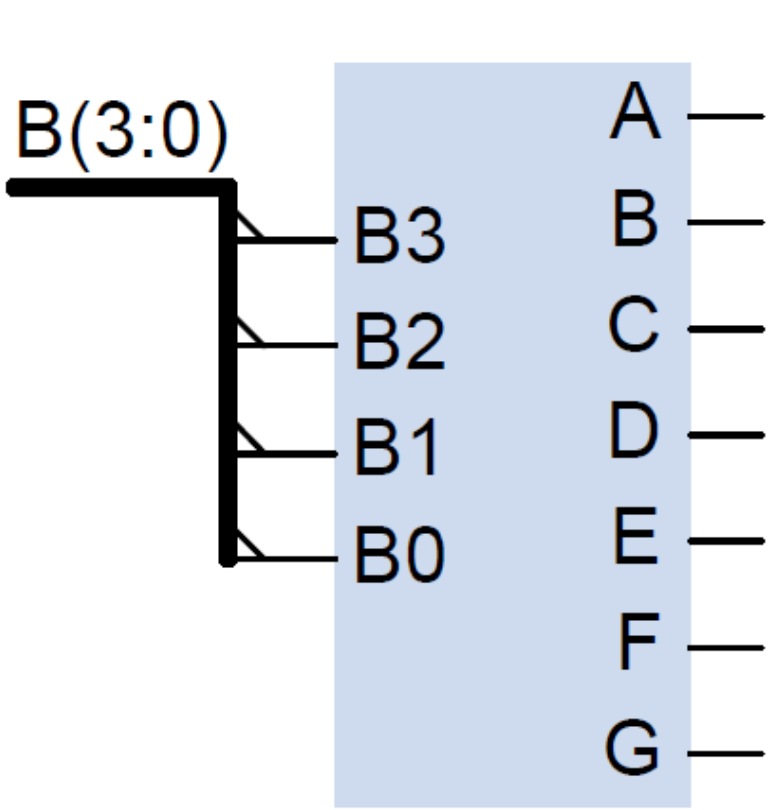
DeMultiplexer (DeMUX)

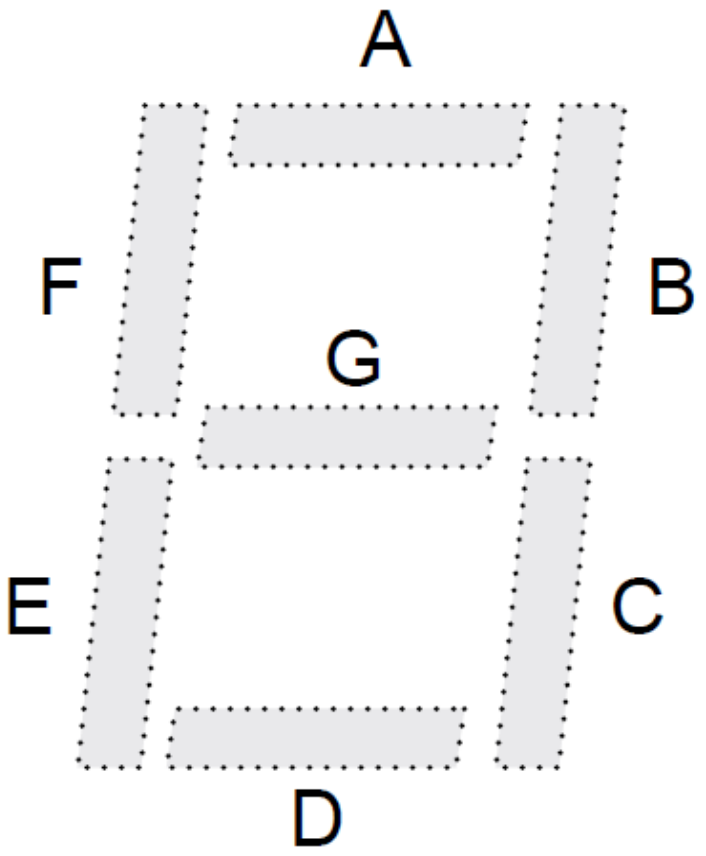


DeMUX VHDL Code

- Assignment: modify the code of a MUX to implement a DeMUX

7-Segment Decoder

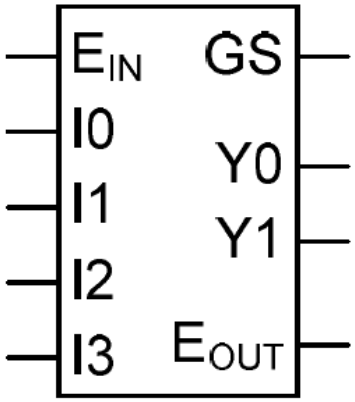




7-Segment Decoder VHDL Code

```
entity seven_seg_dec is
        port (bin: in STD_LOGIC_VECTOR (3 downto 0);
              segout : out STD_LOGIC_VECTOR (6 downto 0));
end seven_seg_dec;

architecture behavioral of seven_seg_dec is
begin
        with bin select
            segout <=   "1111110" when "0000";
                        "0110000" when "0001";
                                .
                                .
                        "0000001" when others;
end behavioral;
```

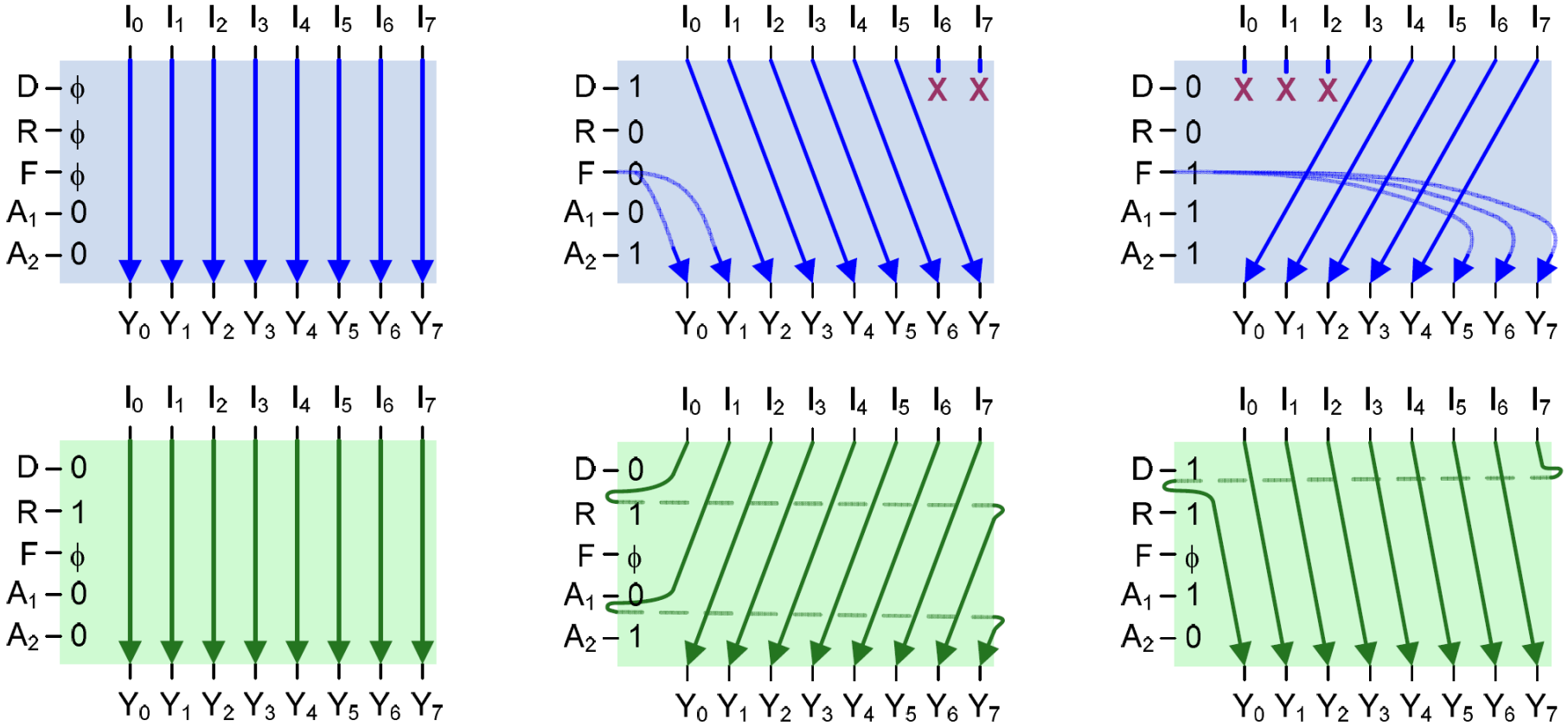
Priority Encoder



Priority Encoder

```
entity encoder is
   port (ein :          in std_logic;
         I :            in std_logic_vector (3 downto 0);
         eout, gs :     out std_logic;
         Y :            out std_logic_vector (1 downto 0));
end encoder;

architecture Behavioral of encoder is
Begin
  eout <=ein and not I(3) and not I(2) and not I(1) and not I(0);
  gs <= ein and (I(3) or I(2) or I(1) or I(0));
  Y(1) <= I(3) or I(2);
  Y(0) <= I(3) or I(1);
end Behavioral;
```

Shifters


I0 I1 I2 I3 I4 I5 I6 I7
D – ϕ
R – ϕ
F – ϕ
A1 – 0
A2 – 0
Y0 Y1 Y2 Y3 Y4 Y5 Y6 Y7
I0 I1 I2 I3 I4 I5 I6 I7
D – 1
R – 0
F – 0
A1 – 0
A2 – 1
X X
Y0 Y1 Y2 Y3 Y4 Y5 Y6 Y7
I0 I1 I2 I3 I4 I5 I6 I7
D – 0
R – 0
F – 1
A1 – 1
A2 – 1
X X X
Y0 Y1 Y2 Y3 Y4 Y5 Y6 Y7
I0 I1 I2 I3 I4 I5 I6 I7
D – 0
R – 1
F – ϕ
A1 – 0
A2 – 0
Y0 Y1 Y2 Y3 Y4 Y5 Y6 Y7
I0 I1 I2 I3 I4 I5 I6 I7
D – 0
R 1
F – ϕ
A1 – 0
A2 – 1
Y0 Y1 Y2 Y3 Y4 Y5 Y6 Y7
I0 I1 I2 I3 I4 I5 I6 I7
D – 1
R – 1
F – ϕ
A1 – 1
A2 – 0
Y0 Y1 Y2 Y3 Y4 Y5 Y6 Y7

Shifters

EN	R	D	Y_3	Y_2	Y_1	Y_0
0	ϕ	ϕ	0	0	0	0
1	0	0	I_2	I_1	I_0	0
1	0	1	0	I_3	I_2	I_1
1	1	0	I_2	I_1	I_0	I_3
1	1	1	I_0	I_3	I_2	I_1

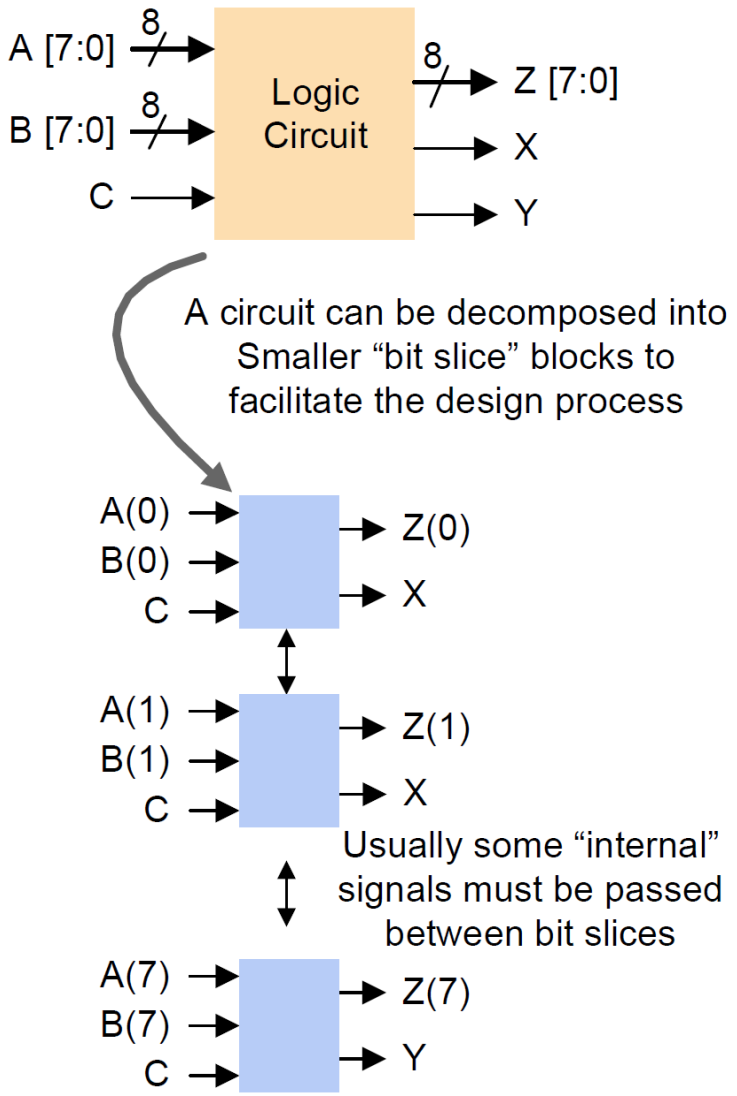
Truth table for 4-bit shifter with shift/rotate left/right functions

Shifters VHDL Example

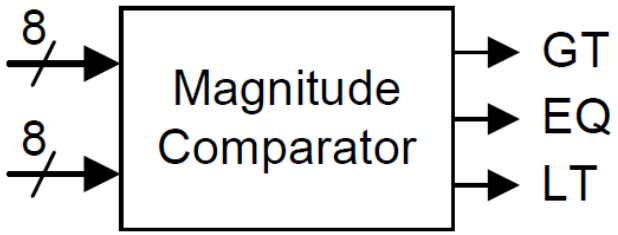
```
entity my_shift is
   port ( din:        in std_logic_vector (7 downto 0);
          r, d, en:   in std_logic;
          dout:       out std_logic_vector (7 downto 0));
end my_shift;

architecture my_shift_arch of my_shift is
begin
  dout <= “00000000” when en = '0' else
     din(6 downto 0) & din(7) when (r = '1' and d = '0') else
     din(0) & din(7 downto 1) when (r = '1' and d = '1') else
     din(6 downto 0) & '0' when (r = '0' and d = '0') else
     '0' & din(7 downto 1);
end my_shift_arch;
```

Bit-Slice Design Method

- Consider a circuit that works on a pair of bits
- Goal is to create a circuit that can simply be replicated N times
  - once for each bit
  - Some circuits defy this approach
- Information passing between adjacent bits

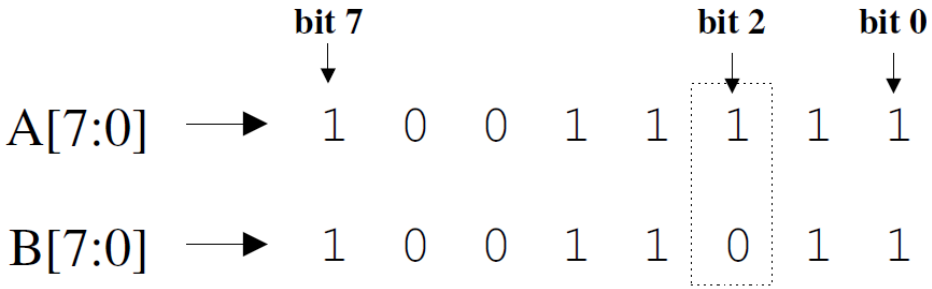


Comparators



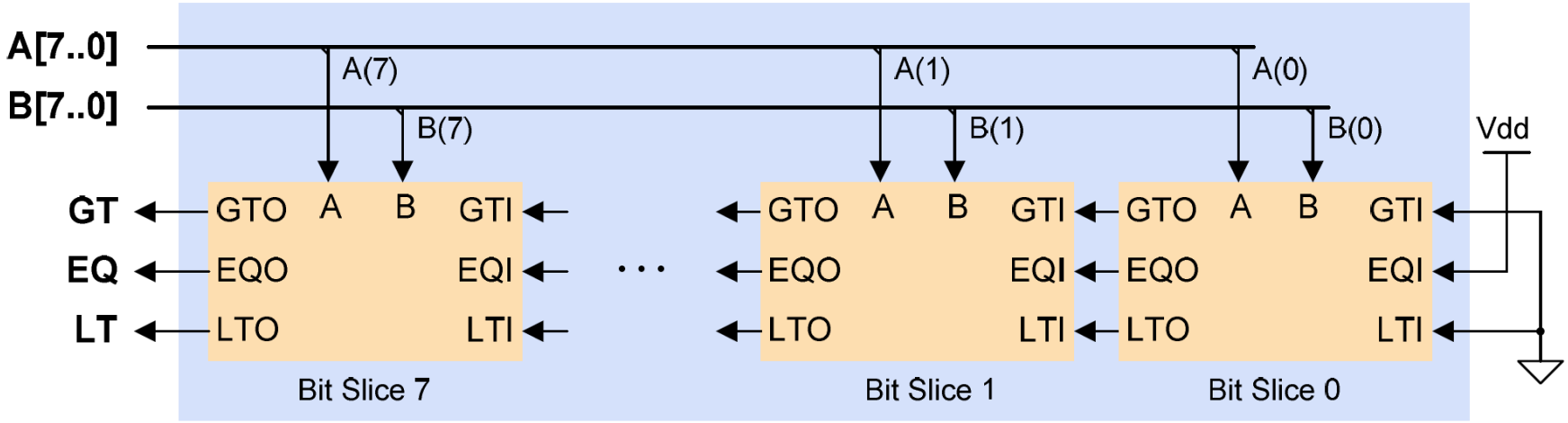

```
entity my_comp is
   port ( A, B  : in std_logic_vector (7 downto 0);
          gt, lt  : inout std_logic;
          eq     : out std_logic);
end my_comp;

architecture behavioral of my_comp is
Begin
          gt <= '1' when  A > B else '0';
          lt <= '1' when A < B else '0';
          eq <= not gt and not lt;
end behavioral;
```

Comparator Bit-Slice Design

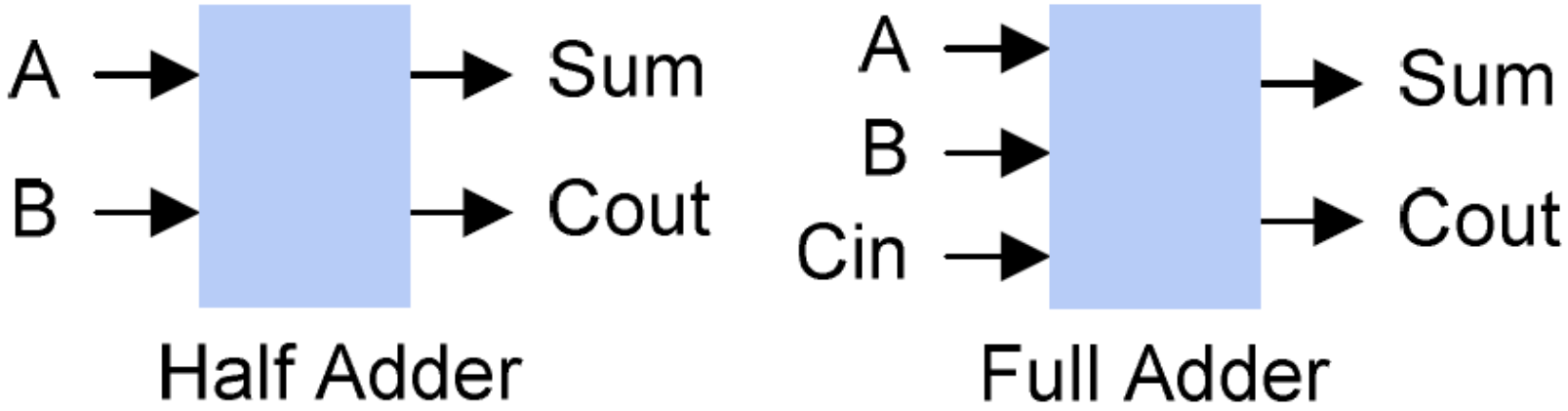


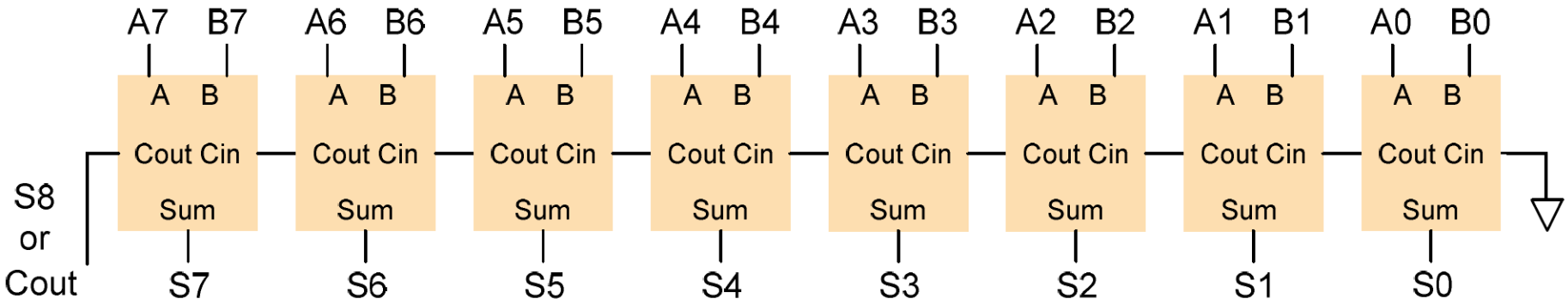
Operand inputs		Inputs from neighboring slices			Bit-slice outputs		
An	Bn	GTI	LTI	EQI	GTO	LTO	EQO
0	0	1	0	0	1	0	0
0	0	0	1	0	0	1	0
0	0	0	0	1	0	0	1
0	1	ϕ	ϕ	ϕ	0	1	0
1	0	ϕ	ϕ	ϕ	1	0	0
1	1	1	0	0	1	0	0
1	1	0	1	0	0	1	0
1	1	0	0	1	0	0	1

Comparator Bit-Slice Design

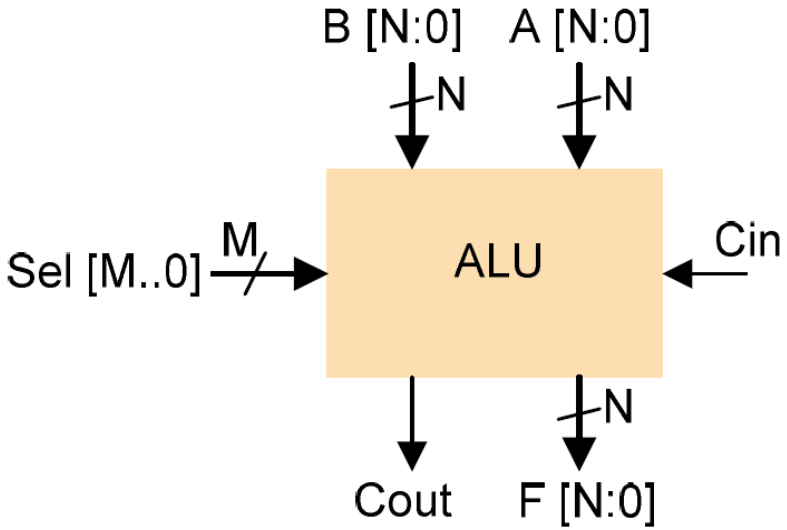


Adders


A
B
Sum
Cout
Half Adder
A
B
Cin
Sum
Cout
Full Adder


A7 B7 A6 B6 A5 B5 A4 B4 A3 B3 A2 B2 A1 B1 A0 B0
A B
Cout Cin
Sum
S8
or
Cout
S7 S6 S5 S4 S3 S2 S1 S0

Arithmetic and Logic Unit (ALU)



Op Code	Function
000	A PLUS B
001	A PLUS 1
010	A MINUS B
011	0
100	A XOR B
101	A’
110	A OR B
111	A AND B

8-Bit, 4-Function ALU VHDL

```
entity ALU is
  port ( A, B : in std_logic_vector (7 downto 0);
         Sel  : in std_logic_vector (1 downto 0);
         Y    : out std_logic_vector (7 downto 0));
end ALU;

architecture behavioral of ALU is
Begin
   With sel select
      Y <=  (A + B) when “00”,
            (A + “00000001”) when “01”,
            (A or B) when “10”,
            (A and B) when others;
end behavioral;
```

Assignment

- Lab Project P6
  - Do only on Xilinx Webpack
  - Simulate to show results
  - No Digilent board demo is necessary