MSI – Medium Scale Integration

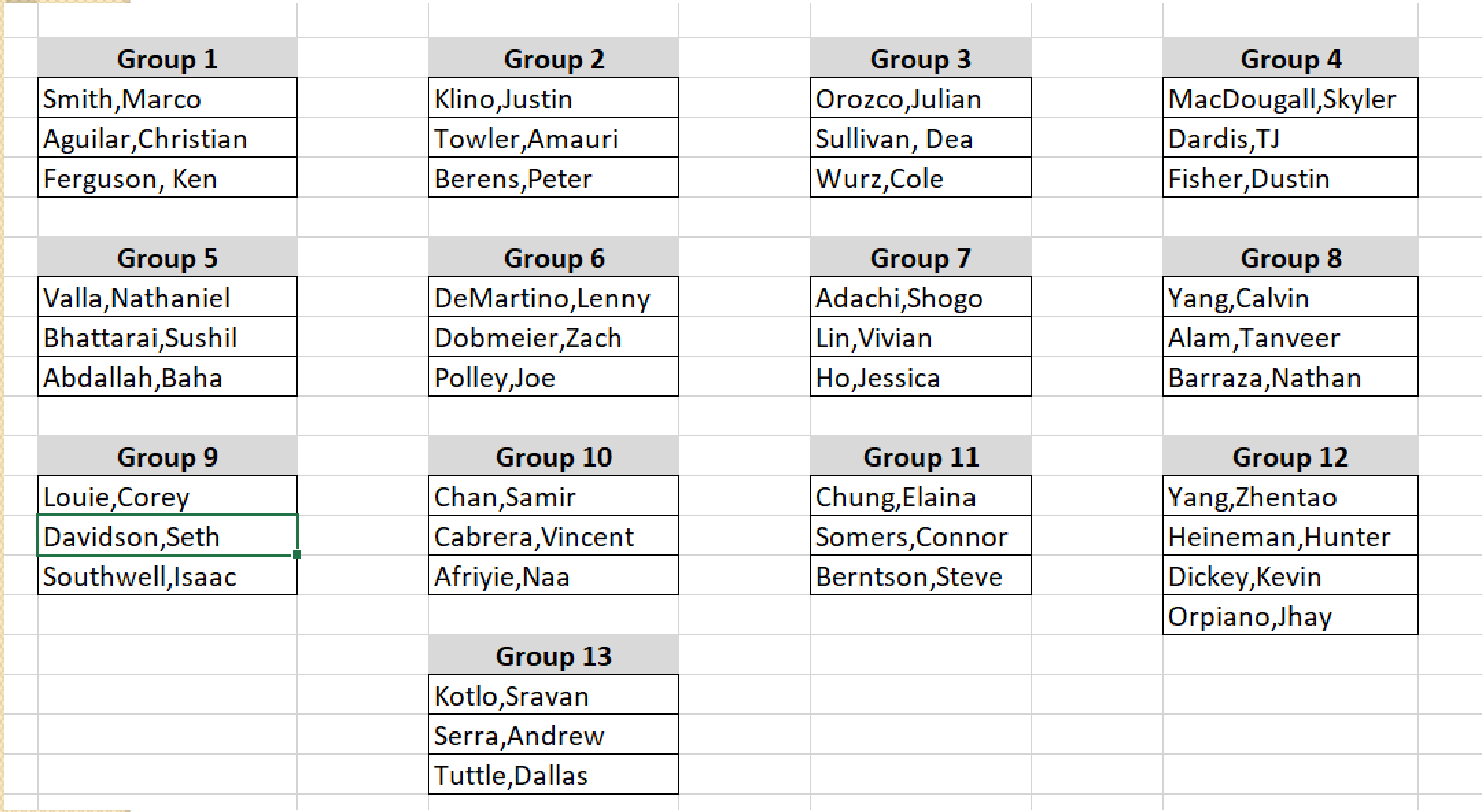
From traditional gate designs to VHDL

Announcements

- HW #6 due today
- HW #7 posted – new homework groups
    - Extended dropbox due date due to long weekend
- Oct 21st Exam #2

New Homework Groups

- Exchange contact information

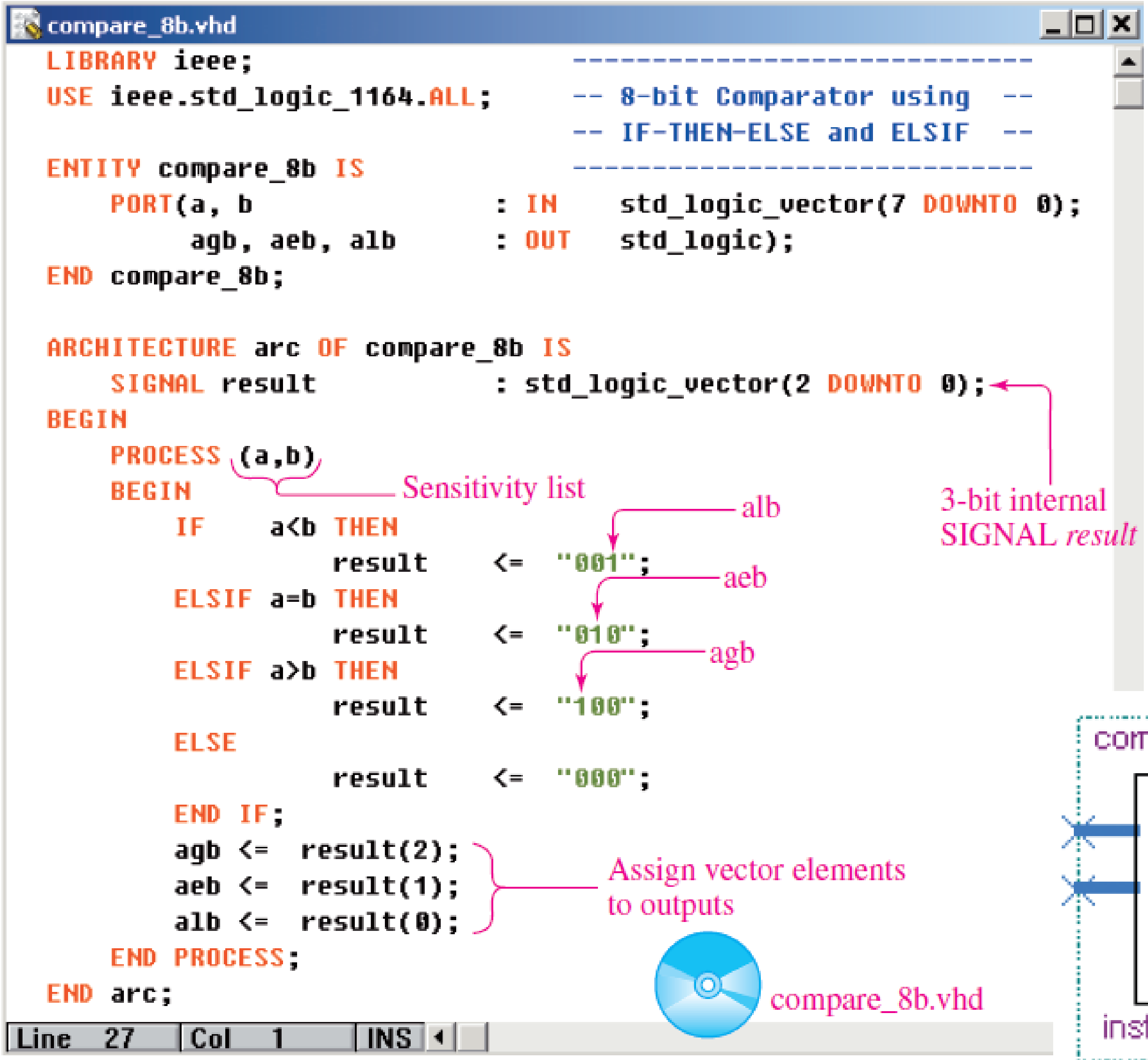


Group 1	Group 2	Group 3	Group 4
Smith,Marco	Klino,Justin	Orozco,Julian	MacDougall,Skyler
Aguilar,Christian	Towler,Amauri	Sullivan, Dea	Dardis,TJ
Ferguson, Ken	Berens,Peter	Wurz,Cole	Fisher,Dustin
Group 5	**Group 6**	**Group 7**	**Group 8**
Valla,Nathaniel	DeMartino,Lenny	Adachi,Shogo	Yang,Calvin
Bhattarai,Sushil	Dobmeier,Zach	Lin,Vivian	Alam,Tanveer
Abdallah,Baha	Polley,Joe	Ho,Jessica	Barraza,Nathan
Group 9	**Group 10**	**Group 11**	**Group 12**
Louie,Corey	Chan,Samir	Chung,Elaina	Yang,Zhentao
Davidson,Seth	Cabrera,Vincent	Somers,Connor	Heineman,Hunter
Southwell,Isaac	Afriyie,Naa	Berntson,Steve	Dickey,Kevin
			Orpiano,Jhay
	Group 13		
	Kotlo,Sravan		
	Serra,Andrew		
	Tuttle,Dallas		

Comparators

- Compare two binary strings or words
- Digital comparator
- Compare bit-by-bit

VHDL Comparator



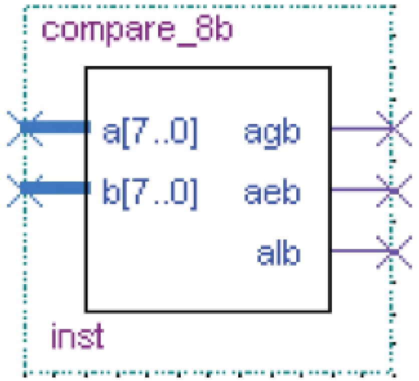
compare_8b.vhd

```
LIBRARY ieee;                       ------------------------------
USE ieee.std_logic_1164.ALL;        -- 8-bit Comparator using   --
                                    -- IF-THEN-ELSE and ELSIF   --
ENTITY compare_8b IS                ------------------------------
    PORT(a, b                : IN   std_logic_vector(7 DOWNTO 0);
         agb, aeb, alb       : OUT  std_logic);
END compare_8b;

ARCHITECTURE arc OF compare_8b IS
    SIGNAL result            : std_logic_vector(2 DOWNTO 0);
BEGIN
    PROCESS (a,b)
    BEGIN
        IF    a<b THEN
                result    <=   "001";
        ELSIF a=b THEN
                result    <=   "010";
        ELSIF a>b THEN
                result    <=   "100";
        ELSE
                result    <=   "000";
        END IF;
        agb <=  result(2);
        aeb <=  result(1);
        alb <=  result(0);
    END PROCESS;
END arc;
```

Line 27 | Col 1 | INS

Sensitivity list
alb
aeb
agb
3-bit internal SIGNAL *result*
Assign vector elements to outputs
compare_8b.vhd



Synthesized Comparators

- Every time a relational statement is included in VHDL, a comparator is synthesized.
- Examples
  - If (var1 >= 1)
  - If (var1 = 0)
  - If (var1 < var2)
- Note, less hardware is inferred with = than <, >, <=, /=, or >=
  - Use the = case if possible

Synthesized Comparators

```
IF (var1 = 0) then
    x<= '1';
ELSE
    x <= '0';
END IF;
```

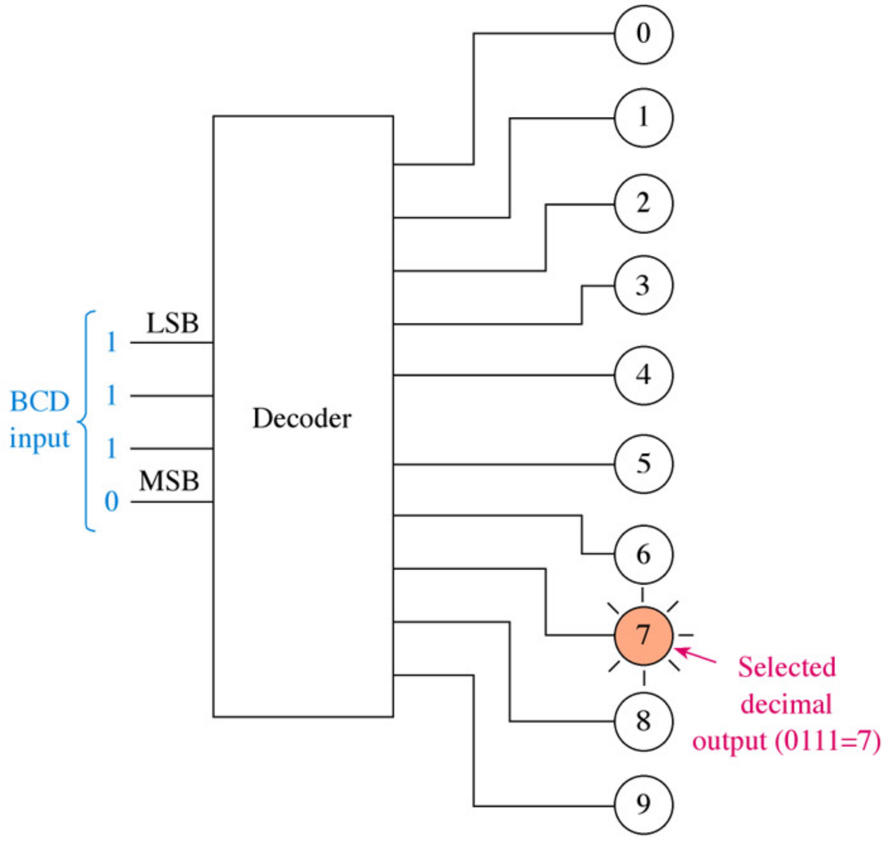
Even though the result is the same,
This method is better

```
IF (var1 >= 1) then
    x <= '0';
ELSE
    x <= '1';
END IF
```

*Assume var1 is an unsigned number

Decoding

- Process of converting some code (binary, BCD, or hex) to a single output
  - One and only one output active at a time
- 4-bit BCD decoder
  - Comprised of a combination of logic gates



Decoding

- 3 to 8 Decoder Truth Tables

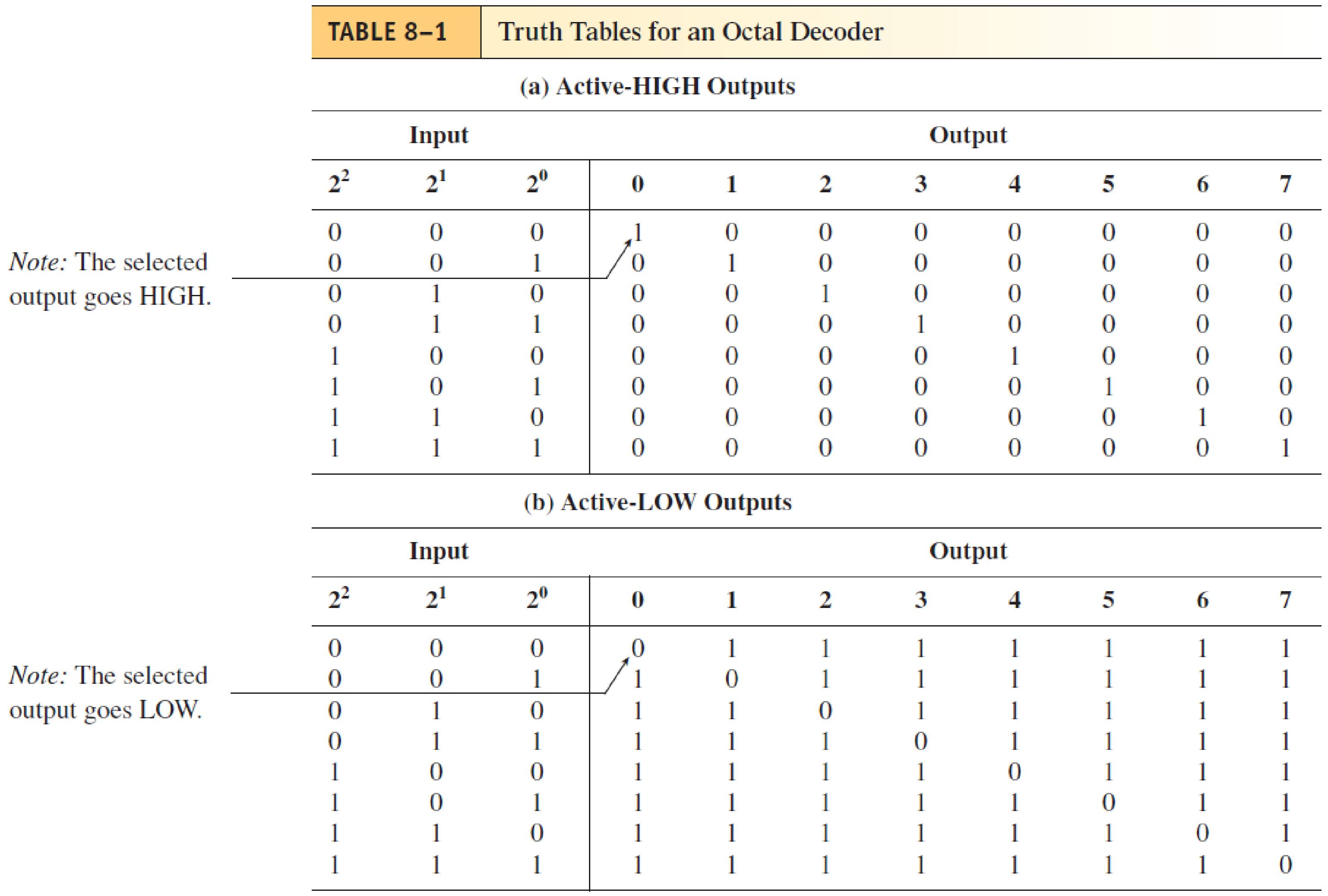


TABLE 8–1 Truth Tables for an Octal Decoder

(a) Active-HIGH Outputs

Input			Output							
2^2	2^1	2^0	0	1	2	3	4	5	6	7
0	0	0	1	0	0	0	0	0	0	0
0	0	1	0	1	0	0	0	0	0	0
0	1	0	0	0	1	0	0	0	0	0
0	1	1	0	0	0	1	0	0	0	0
1	0	0	0	0	0	0	1	0	0	0
1	0	1	0	0	0	0	0	1	0	0
1	1	0	0	0	0	0	0	0	1	0
1	1	1	0	0	0	0	0	0	0	1

Note: The selected output goes HIGH.

(b) Active-LOW Outputs

Input			Output							
2^2	2^1	2^0	0	1	2	3	4	5	6	7
0	0	0	0	1	1	1	1	1	1	1
0	0	1	1	0	1	1	1	1	1	1
0	1	0	1	1	0	1	1	1	1	1
0	1	1	1	1	1	0	1	1	1	1
1	0	0	1	1	1	1	0	1	1	1
1	0	1	1	1	1	1	1	0	1	1
1	1	0	1	1	1	1	1	1	0	1
1	1	1	1	1	1	1	1	1	1	0

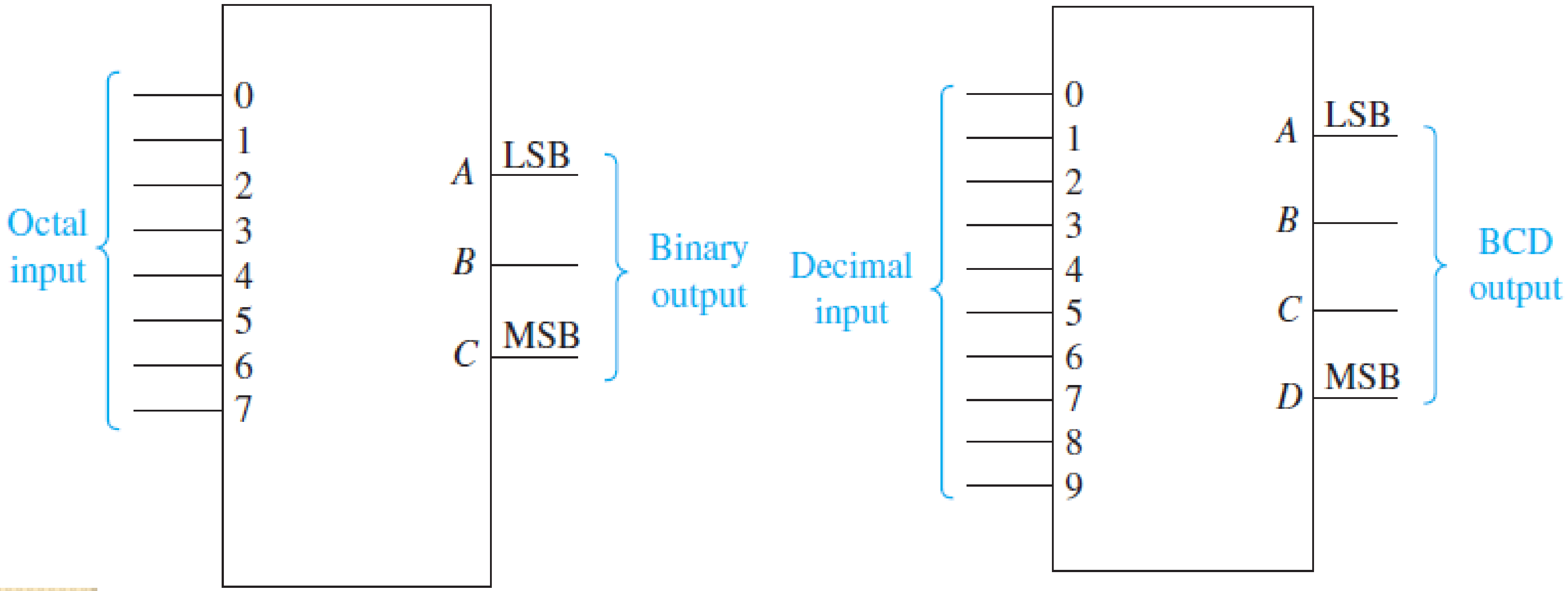
Note: The selected output goes LOW.

Encoding

- Opposite of decoding
- Used to generate a coded output from a singular active numeric input line.

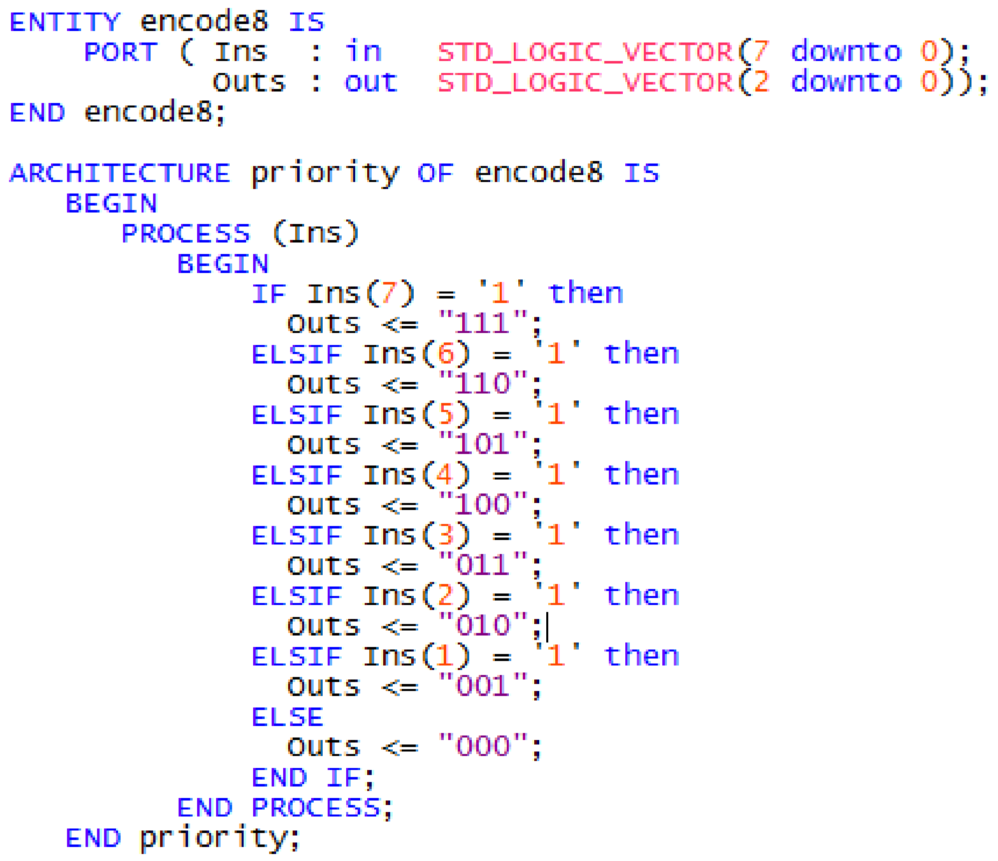
Encoding

- Octal-to-binary and decimal-to-BCD encoders



- Only 1 input active at a time

VHDL Octal Priority Encoder



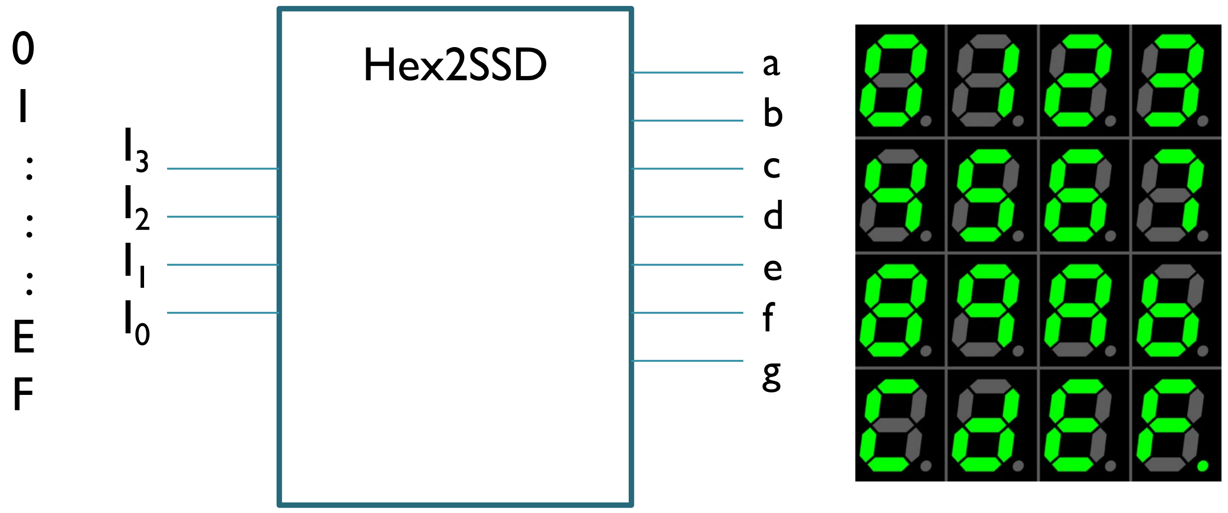
```vhdl
ENTITY encode8 IS
    PORT ( Ins   : in    STD_LOGIC_VECTOR(7 downto 0);
           Outs  : out   STD_LOGIC_VECTOR(2 downto 0));
END encode8;

ARCHITECTURE priority OF encode8 IS
   BEGIN
      PROCESS (Ins)
         BEGIN
             IF Ins(7) = '1' then
               Outs <= "111";
             ELSIF Ins(6) = '1' then
               Outs <= "110";
             ELSIF Ins(5) = '1' then
               Outs <= "101";
             ELSIF Ins(4) = '1' then
               Outs <= "100";
             ELSIF Ins(3) = '1' then
               Outs <= "011";
             ELSIF Ins(2) = '1' then
               Outs <= "010";
             ELSIF Ins(1) = '1' then
               Outs <= "001";
             ELSE
               Outs <= "000";
             END IF;
         END PROCESS;
   END priority;
```

What happens if more than one input is active?

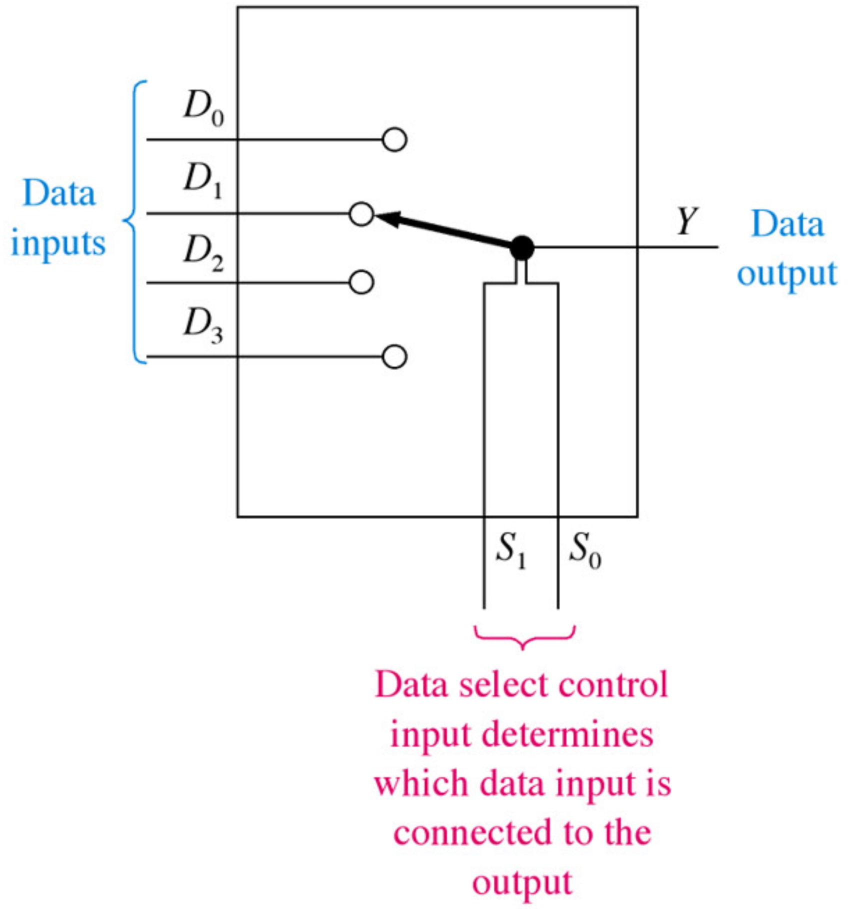
Code Converters

- Convert a coded input into another form
- Hexadecimal to seven segment display is an example you are familiar with



Multiplexers

- Funneling several data lines into a single one for transmission to another point
- Data select control determines which input is transmitted



Data inputs
D_0
D_1
D_2
D_3
Y
Data output
S_1
S_0
Data select control input determines which data input is connected to the output

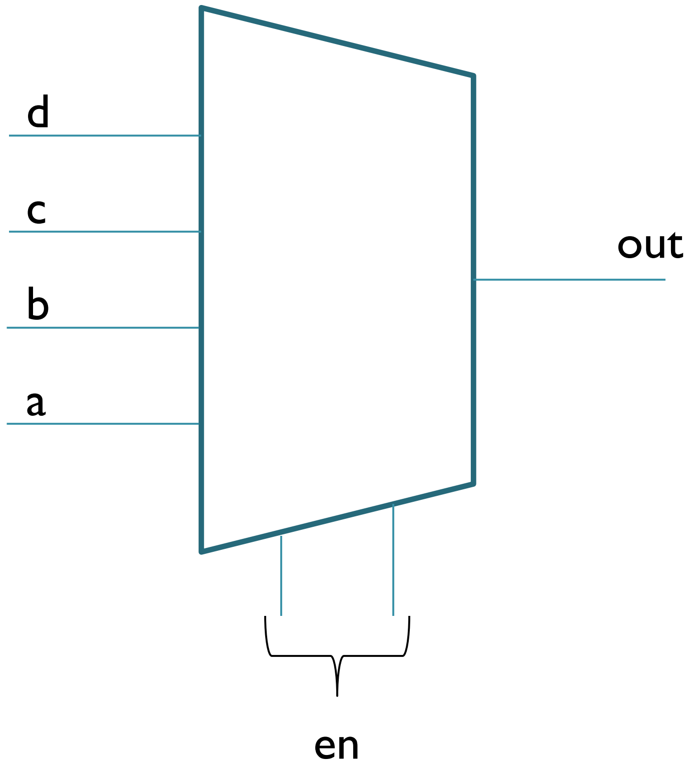
Multiplexer sizes

- Common Sizes
  - 4-to-1 (2 select lines)
  - 8-to-1 (3 select lines)
  - 16-to-1 (4 select lines)
- Pattern?
  - For n–to-1 mux, $\log_2(n)$ select lines needed

Mux / Case statement

- A case statement synthesizes to a Mux

```
Choose : process(a,b,c,d,en)
Begin
        case (en) is
          when "00" => out <= a;
          when "01" => out <= b;
          when "10" => out <= c;
          when others => out <= d;
        end case;
End process;
```

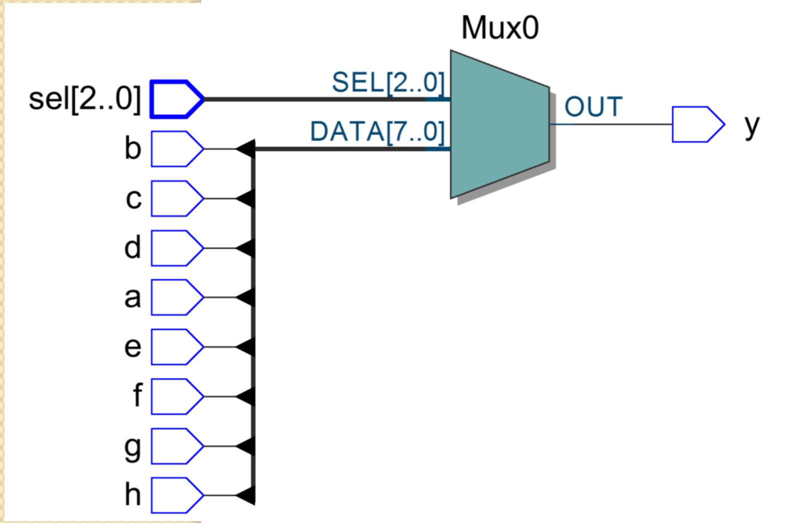


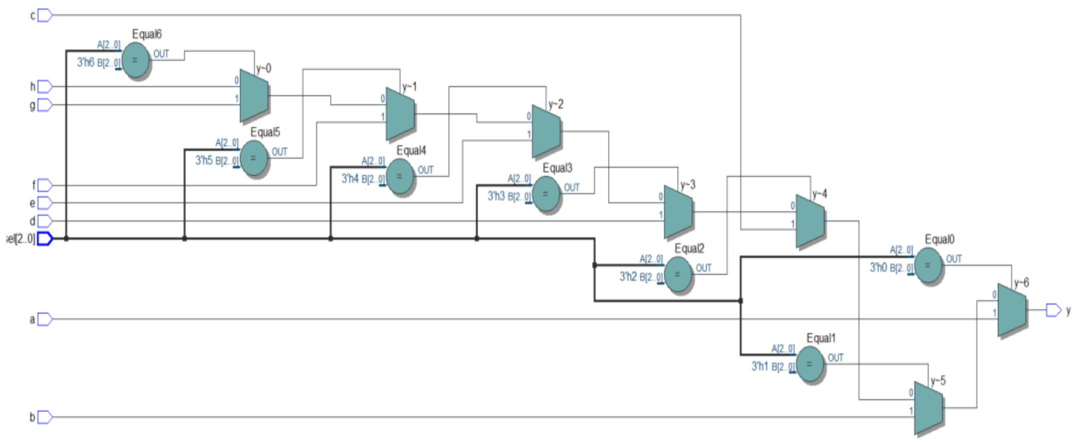
Case vs. IF-THEN-ELSE

```
architecture behave of mux8to1 is
begin

  process (a, b, c, d, e, f, g, h, sel) is
    begin
      case sel is
        when "000" => y <= a;
        when "001" => y <= b;
        when "010" => y <= c;
        when "011" => y <= d;
        when "100" => y <= e;
        when "101" => y <= f;
        when "110" => y <= g;
        when others => y <= h;
      end case;
    end process;
end behave;
```

```
  process (a, b, c, d, e, f, g, h, sel) is
    begin
        if sel = "000" then y <= a;
        elsif sel =  "001" then y <= b;
        elsif sel =  "010" then y <= c;
        elsif sel =  "011" then y <= d;
        elsif sel =  "100" then y <= e;
        elsif sel =  "101" then y <= f;
        elsif sel =  "110" then y <= g;
        else                    y <= h;
      end if;
    end process;
end behave;
```





Case vs. IF-ELSIF-ELSE

- Case creates a MUX
  - One logic level regardless of # of conditions
- IF-ELSIF-ELSE creates a *priority encoder*
  - # of logic levels = # of conditions -1
  - Why could this be a problem?