Decoder, Encoder, Shifter Examples

Last updated 10/31/24

Problem 1

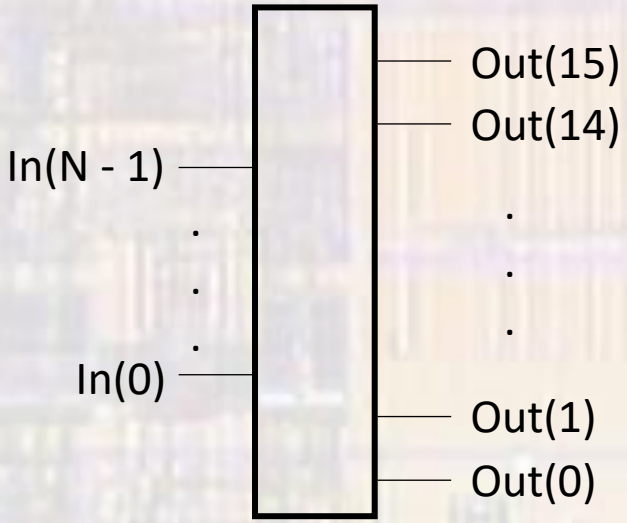
- Microcontrollers often have multiple external circuits (peripherals) that they may want to access. Each of these external circuits has an input pin to enable it to work. These are typically called Chip Select or Chip enable and are typically active low (cs_bar, ce_bar, cs or ce).

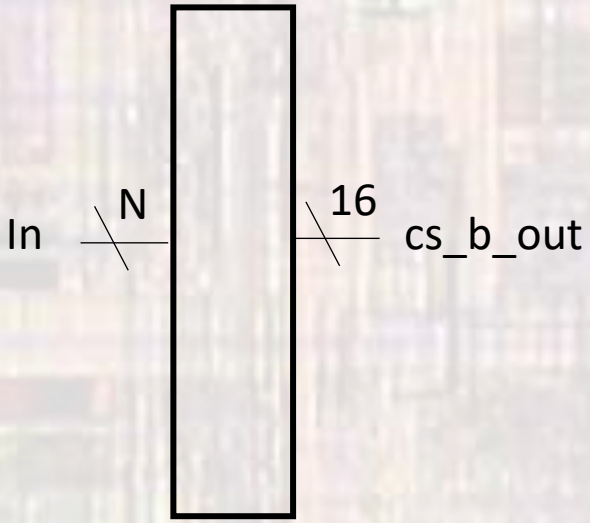
- Create a block in vhdl to allow the microcontroller to use the minimum number of bits to select one of 16 peripherals

Problem 1

- Block Diagram

N = ???





Problem 1

- First Try
  - What's wrong here!

```
--------------------------------------
--
-- decoder_cs.vhdl
--
-- created 7/5/2018
-- tj
--
-- rev 0
--------------------------------------
--
-- vhdl decoder for processor chip select
--
--------------------------------------
--
-- Inputs: in(3-0)
-- Outputs: 16 wires (active low chip select)
--
--------------------------------------
library ieee;
use ieee.std_logic_1164.all;

entity decoder_cs is
   port (
         i_in:    in std_logic_vector(3 downto 0);
         o_cs_b:  out std_logic_vector(15 downto 0)
         );
end entity;
```

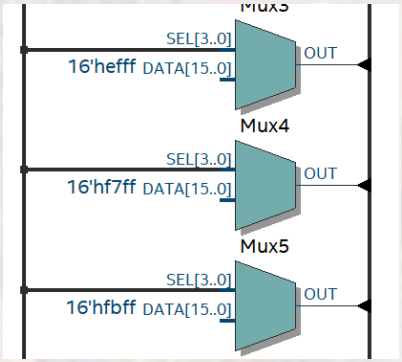
```
architecture behavioral of decoder_cs is

begin

   with i_in select o_cs_b <=
                              "1111111111111110" when "0000",
                              "1111111111111101" when "0001",
                              "1111111111111011" when "0010",
                              "1111111111110111" when "0011",
                              "1111111111101111" when "0100",
                              "1111111111011111" when "0101",
                              "1111111110111111" when "0110",
                              "1111111101111111" when "0111",
                              "1111111011111111" when "1000",
                              "1111110111111111" when "1001",
                              "1111101111111111" when "1010",
                              "1111011111111111" when "1011",
                              "1110111111111111" when "1100",
                              "1101111111111111" when "1101",
                              "1011111111111111" when "1110",
                              "0111111111111111" when "1111",
                              "1111111111111111" when others;

end behavioral;
```

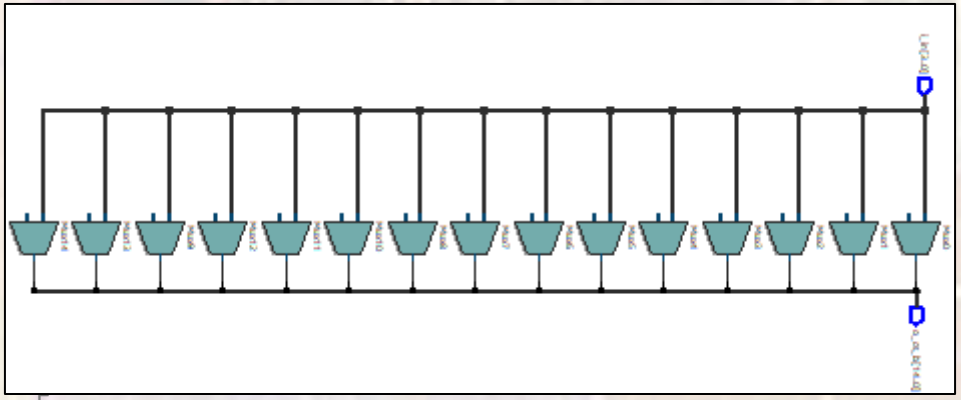

Problem 1

- Second Try

```
----------------------------------------
--
-- decoder_cs.vhdl
--
-- created 7/5/2018
-- tj
--
-- rev 0
----------------------------------------
--
-- vhdl decoder for processor chip select
--
----------------------------------------
--
-- Inputs: in(3-0)
-- Outputs: 15 wires (active low chip select)
--
----------------------------------------
library ieee;
use ieee.std_logic_1164.all;

entity decoder_cs is
   port (
         i_in:    in std_logic_vector(3 downto 0);
         o_cs_b:  out std_logic_vector(14 downto 0)
         );
end entity;
```

```
architecture behavioral of decoder_cs is

begin

   with i_in select o_cs_b <=
                              "111111111111111" when "0000",
                              "111111111111110" when "0001",
                              "111111111111101" when "0010",
                              "111111111111011" when "0011",
                              "111111111110111" when "0100",
                              "111111111101111" when "0101",
                              "111111111011111" when "0110",
                              "111111110111111" when "0111",
                              "111111101111111" when "1000",
                              "111111011111111" when "1001",
                              "111110111111111" when "1010",
                              "111101111111111" when "1011",
                              "111011111111111" when "1100",
                              "110111111111111" when "1101",
                              "101111111111111" when "1110",
                              "011111111111111" when "1111",
                              "111111111111111" when others;

end behavioral;
```



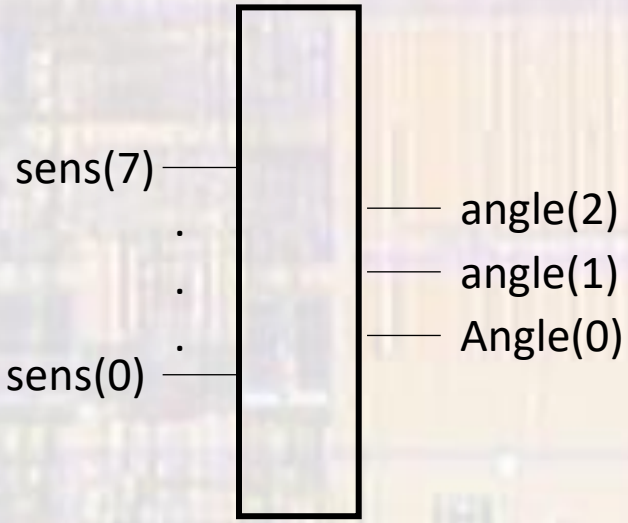


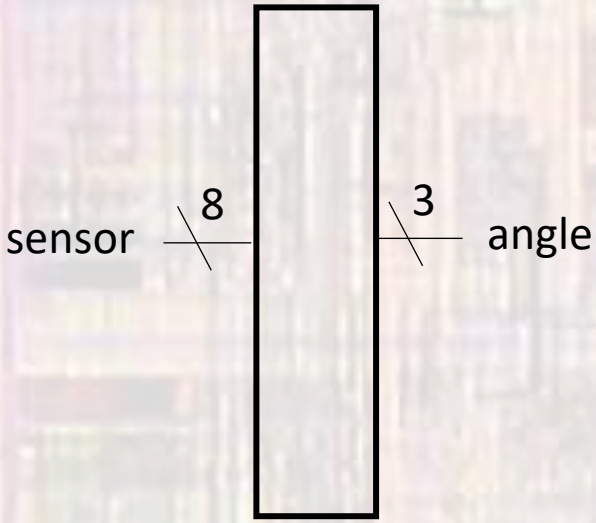
Problem 2

- A robot has 8 sensors spaced equally around a rotating joint. Your design needs to send the location of the joint back to a microcontroller in as few wires as possible (why?). Create a block to transmit the angular location to the microcontroller in as few wires as possible

Problem 2

- Block Diagram



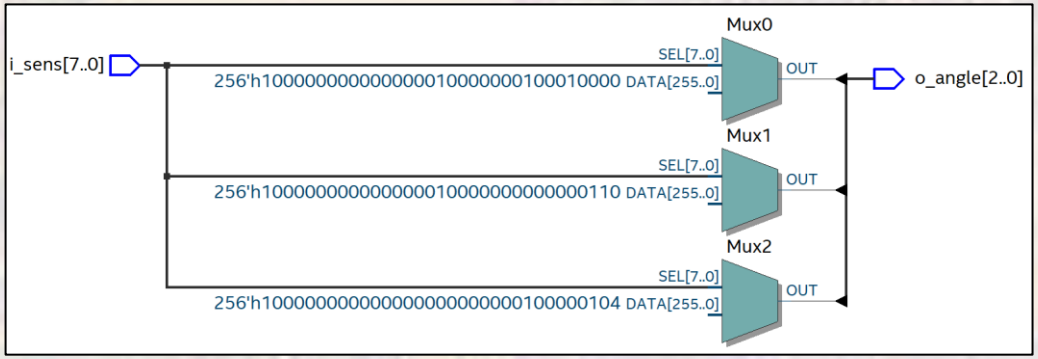


Problem 2

- First Try

```
----------------------------------------
--
-- robot_angle.vhdl
--
-- created 7/5/2018
-- tj
--
-- rev 0
----------------------------------------
--
-- vhdl robot are position - using with-select
--
----------------------------------------
--
-- Inputs: sens(7-0)
-- Outputs: angle(2-0)
--
----------------------------------------
library ieee;
use ieee.std_logic_1164.all;

entity robot_angle is
   port (
         i_sens:     in std_logic_vector(7 downto 0);
         o_angle:    out std_logic_vector(2 downto 0)
         );
end entity;
```

```
architecture behavioral of robot_angle is

begin

   with i_sens select o_angle <=
                              "000" when "00000001",
                              "001" when "00000010",
                              "010" when "00000100",
                              "011" when "00001000",
                              "100" when "00010000",
                              "101" when "00100000",
                              "110" when "01000000",
                              "111" when "10000000",
                              "000" when others;

end behavioral;
```

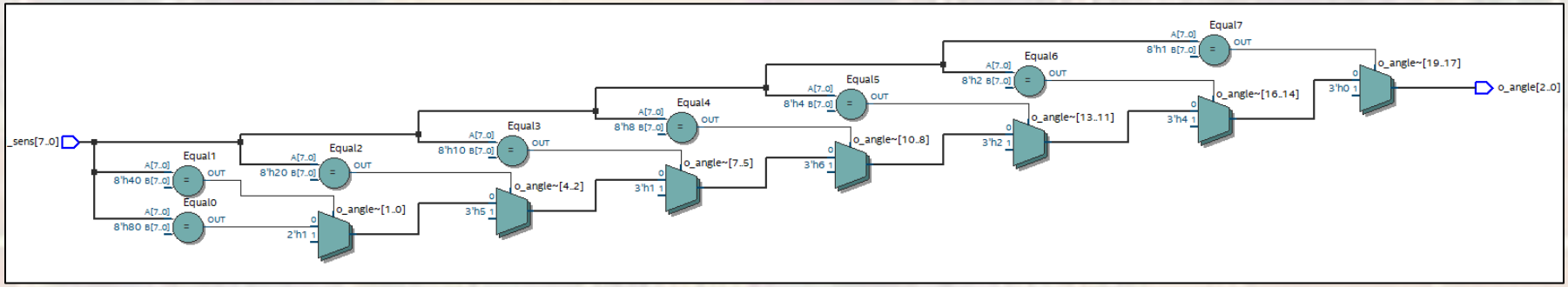


Problem 2

- Second Try

```
----------------------------------------
--
-- robot_angle2.vhdl
--
-- created 7/5/2018
-- tj
--
-- rev 0
----------------------------------------
--
-- vhdl robot arm position - using when else
--
----------------------------------------
--
-- Inputs: sens(7-0)
-- Outputs: angle(2-0)
--
----------------------------------------
library ieee;
use ieee.std_logic_1164.all;

entity robot_angle2 is
   port (
         i_sens:     in std_logic_vector(7 downto 0);
         o_angle:    out std_logic_vector(2 downto 0)
         );
end entity;
```

```
architecture behavioral of robot_angle2 is

begin

    o_angle <=
                "000" when (i_sens = "00000001") else
                "001" when (i_sens = "00000010") else
                "010" when (i_sens = "00000100") else
                "011" when (i_sens = "00001000") else
                "100" when (i_sens = "00010000") else
                "101" when (i_sens = "00100000") else
                "110" when (i_sens = "01000000") else
                "111" when (i_sens = "10000000") else
                "000";

end behavioral;
```



Problem 3

- A fancy new motor has multiple drivers. Each driver requires a 4 bit signal to make it work properly. Multiple drivers may be active at the same time. The motor rotates by shifting the driver signals in a rotating pattern. Provide a block to drive a motor with 8 drivers. The driver 4 bit signals and the current rotation location are provided.

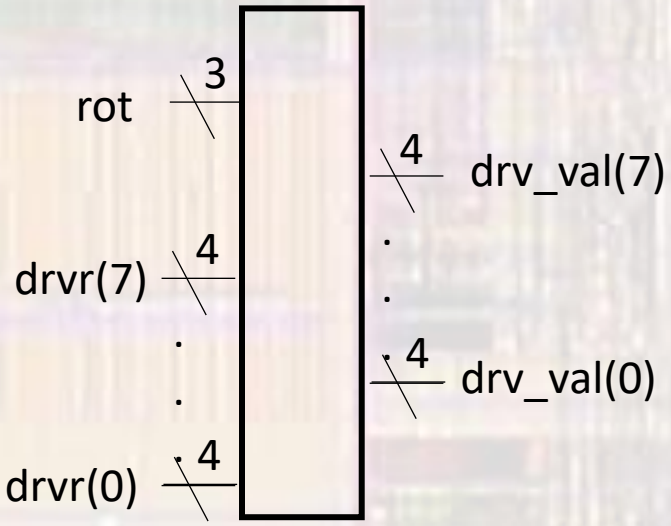
4 driver version

Rotation

A	B	C	D
3	12	5	7
7	3	12	5
5	7	3	12
12	5	7	3

Problem 3

- Block Diagram



Problem 3

```
----------------------------------------
--
-- motor_driver.vhdl
--
-- created 7/5/2018
-- tj
--
-- rev 0
----------------------------------------
--
-- motor dtiver - left
--
----------------------------------------
--
-- Inputs:  8 x drvr(3-0)
--          rotate amount (2-0)
-- Outputs: 8 x drv_val(3-0)
--
----------------------------------------
library ieee;
use ieee.std_logic_1164.all;

entity motor_driver is
   port (
         i_drvrA:     in std_logic_vector(3 downto 0);
         i_drvrB:     in std_logic_vector(3 downto 0);
         i_drvrC:     in std_logic_vector(3 downto 0);
         i_drvrD:     in std_logic_vector(3 downto 0);
         i_drvrE:     in std_logic_vector(3 downto 0);
         i_drvrF:     in std_logic_vector(3 downto 0);
         i_drvrG:     in std_logic_vector(3 downto 0);
         i_drvrH:     in std_logic_vector(3 downto 0);

         i_rot_amt:      in std_logic_vector(2 downto 0);

         o_drv_valA:     out std_logic_vector(3 downto 0);
         o_drv_valB:     out std_logic_vector(3 downto 0);
         o_drv_valC:     out std_logic_vector(3 downto 0);
         o_drv_valD:     out std_logic_vector(3 downto 0);
         o_drv_valE:     out std_logic_vector(3 downto 0);
         o_drv_valF:     out std_logic_vector(3 downto 0);
         o_drv_valG:     out std_logic_vector(3 downto 0);
         o_drv_valH:     out std_logic_vector(3 downto 0)
         );
end entity;
```

with-select and when-else don't handle this situation very well

```
architecture behavioral of motor_driver is

begin

   with i_rot_amt select o_drv_valA <=
            i_drvrA  when "000",
            i_drvrH  when "001",
            i_drvrG  when "010",
            i_drvrF  when "011",
            i_drvrE  when "100",
            i_drvrD  when "101",
            i_drvrC  when "110",
            i_drvrB  when "111",
            i_drvrA  when others;

   with i_rot_amt select o_drv_valB <=
            i_drvrB  when "000",
            i_drvrA  when "001",
            i_drvrH  when "010",
            i_drvrG  when "011",
            i_drvrF  when "100",
            i_drvrE  when "101",
            i_drvrD  when "110",
            i_drvrC  when "111",
            i_drvrB  when others;

            -- 6 more sets

end behavioral;
```