Homework 2

(Due date: October 3rd @ 9:30 am)
Presentation and clarity are very important!

PROBLEM 1 (10 PTS)

- Complete the following table. Use the fewest number of bits for every case. Show your procedure.

Decimal	Sign-and-magnitude	1's complement	2's complement
-257			
128			
-120			
-61			
-1022			
511			

PROBLEM 2 (15 PTS)

- We need to perform the following operations, where numbers are represented in 2's complement:
  - 73 + 45
  - -35 + 64
  - -129 + 128
  - -255 - 230
  - 490 + 47
  - 990 + 113
- For each case:
  - ✓ Determine the minimum number of bits required to represent both summands. You might need to sign-extend one of the summands, since for proper summation, both summands must have the same number of bits.
  - ✓ Perform the binary addition in 2's complement arithmetic. The result must have the same number of bits as the summands.
  - ✓ Determine whether there is overflow by:
    i. Using c_n, c_{n-1} (carries).
    ii. Performing the operation in the decimal system and checking whether the result is within the allowed range for n bits, where n is the minimum number of bits for the summands.
  - ✓ If we want to avoid overflow, what is the minimum number of bits required to represent the summands and the result?

PROBLEM 3 (15 PTS)

- Sign-extension in 2's complement: Whenever we need to increase the number of bits for representing a number, we append the MSB to the left as many times as needed:

$$b_{n-1}b_{n-2}\dots b_0 \equiv b_{n-1}\dots b_{n-1}b_{n-1}b_{n-2}\dots b_0$$

Examples: $00101_2 = 0000101_2 = 2^2 + 2^0 = 5$

$10101_2 = 1110101_2 = -2^4 + 2^2 + 2^0 = -2^6 + 2^5 + 2^4 + 2^2 + 2^0 = -11$

We can think of the sign-extended number as an m-bit number, where $m > n$:

$$b_{n-1}\dots b_{n-1}b_{n-1}b_{n-2}\dots b_0 = k_{m-1}\dots k_n k_{n-1}k_{n-2}\dots k_0$$

- Demonstrate that $b_{n-1}b_{n-2}\dots b_0$ represents the same decimal number as $b_{n-1}\dots b_{n-1}b_{n-1}b_{n-2}\dots b_0$, i.e., demonstrate that sign-extension is correct for any $m > n$.

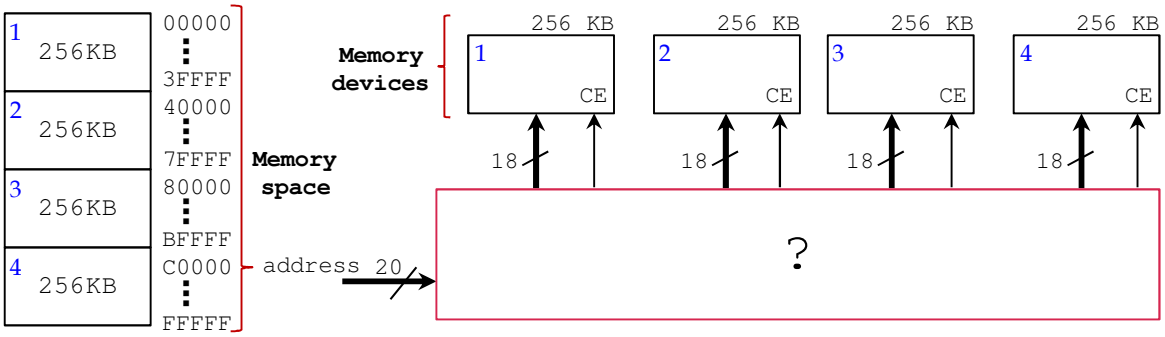
Useful formula: $\sum_{i=k}^{l} r^i = \frac{r^k - r^{l+1}}{1-r}, r \neq 1$

PROBLEM 4 (15 PTS)

- Implement the following functions using i) decoders and ii) multiplexers:
  - ✓ $F = \overline{X + Z} + ZY$
  - ✓ $F(X,Y,Z) = \prod(M_1, M_4, M_7)$
  - ✓ $F = XY + YZ + XZ$
  - ✓ $F = X \oplus Y \oplus Z$
- Using only 2-to-1 MUXs, implement the XOR and XNOR gates.
- Using only a 4-to-1 MUX, implement the following functions.
  - $F(X,Y,Z) = \sum(m_1, m_3, m_5, m_7)$
  - $F(X,Y,Z) = \sum(m_1, m_3, m_5)$
  - $F(X,Y,Z) = \sum(m_3, m_5, m_7)$
  - $F(X,Y,Z) = \sum(m_5, m_7)$

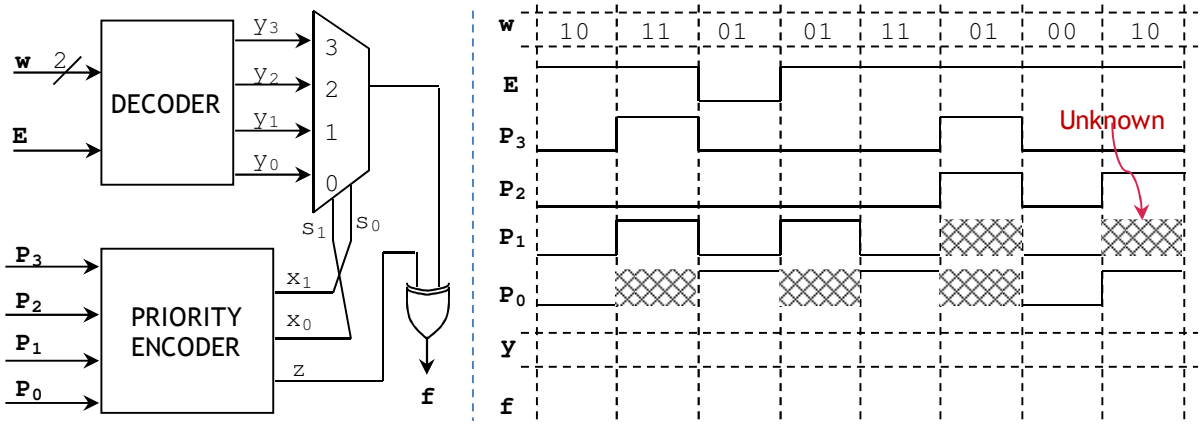
PROBLEM 5 (10 PTS)

- A 20-bit address line in a µprocessor handles up to $2^{20} = 1\,MB$ of addresses, each address containing one-byte of information. We want to connect four 256KB memory chips to the µprocessor.
- Sketch the circuit that: i) addresses the memory chips, and ii) enables only one memory chip (via CE: chip enable) when the address falls in the corresponding range. Example: if $address = 0x5FFFF$, → only memory chip 2 is enabled (CE=1). If $address = 0xD0123$, → only memory chip 4 is enabled.

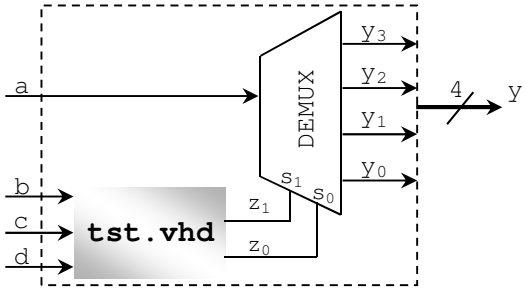


PROBLEM 6 (15 PTS)

- Complete the timing diagram of the circuit shown below:

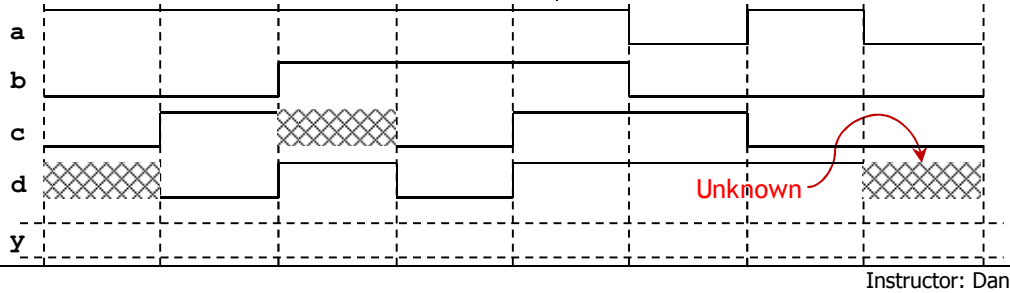


- The following VHDL code corresponds to the shaded circuit. Complete the timing diagram:



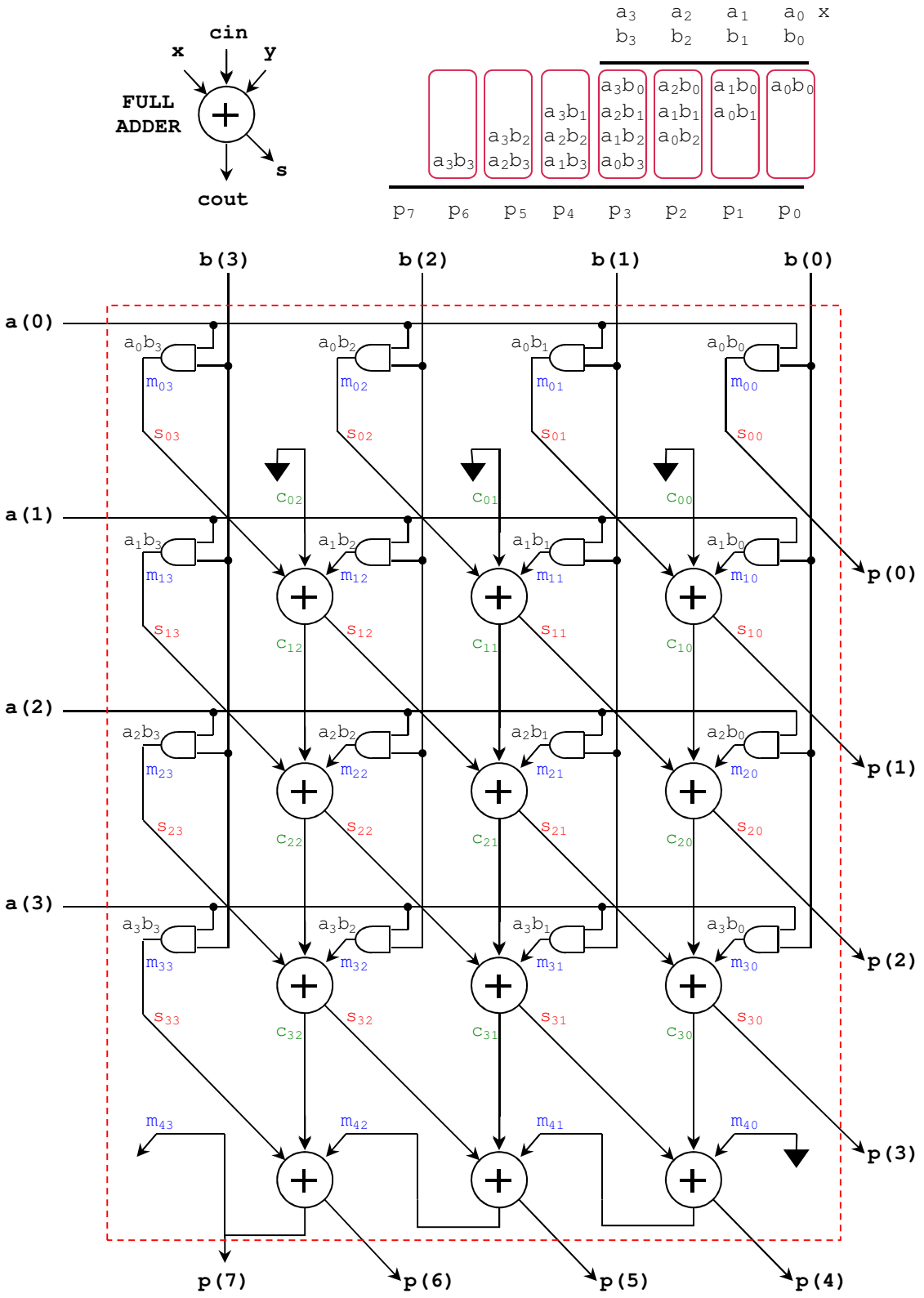

```
library ieee;
use ieee.std_logic_1164.all;

entity tst is
  port (b,c,d: in std_logic;
        z: out std_logic_vector(1 downto 0));
end tst;
```

```
architecture bhv of circ is
  signal x, y: std_logic;
begin
  process (b,c,d)
  begin
    z <= c & d;
    if b = '1' then
       case d is
          when '1' => z <= "01";
          when others => z <= "00";
       end case;
    else
      if c = '0' then z <= "11";
      end if;
    end if;
  end process;
end bhv;
```



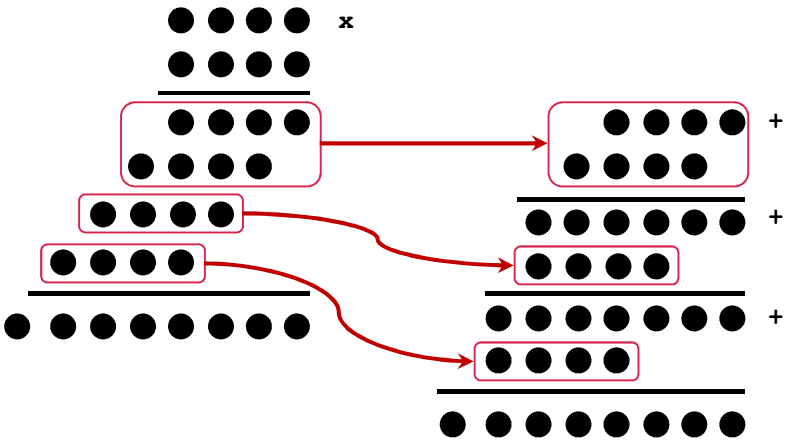
PROBLEM 7 (20 PTS)

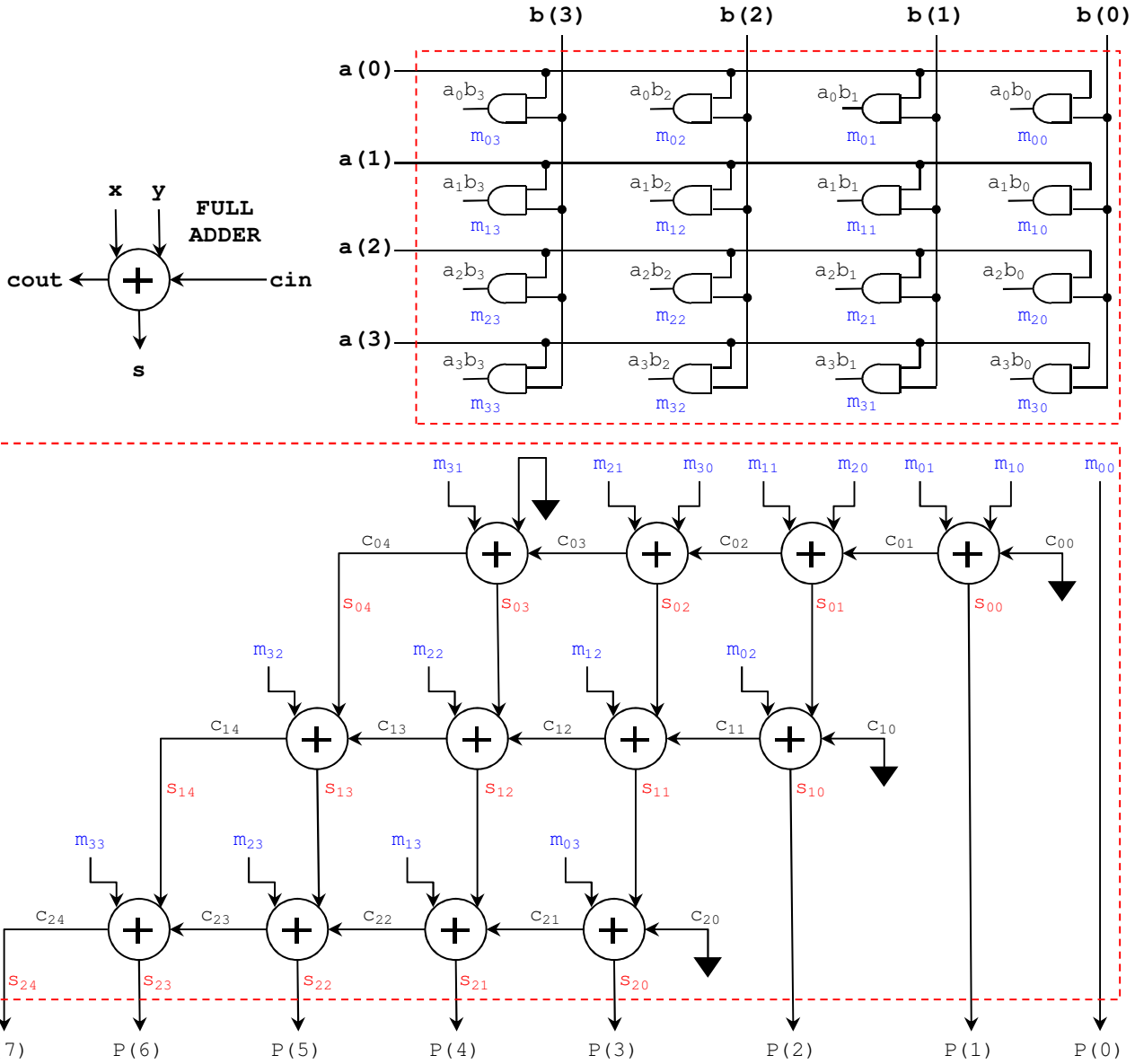
- A straightforward implementation of the multiplication operation (for positive or unsigned numbers) is called ***Array Multiplier***: the partial products are added up at every column. The figure depicts the hardware implementation for multiplying two unsigned number of 4 bits.



- One of the drawbacks of this approach is the long delay between the input and the output. In the figure, the bits p_7, p_6, p_4, p_4 require the inputs to pass through 4 adders as well as AND gates.

- An alternative implementation, called ***Wallace Multiplier***, features a shorter delay between the input and the output. The hardware implementation is more complicated though.
- The figure shows a Wallace multiplier implementation for two unsigned numbers of 4 bits. Note how the addition operation of the 4 rows has been rearranged so that only 2 rows are added up at every stage.





- ✓ Write the VHDL implementation for the multiplication operation of two unsigned numbers of 4 bits using: i) Array Multiplier, and ii) Wallace multiplier. Use the ***Structural Description***: create a separate VHDL file for the Full Adder circuit.
- ✓ Simulate both circuits and make sure that they are performing the correct operation.
- ✓ Attach your VHDL code, testbench code, and simulation results.