Homework 1

To pass this assignment, you need to complete the following tasks:

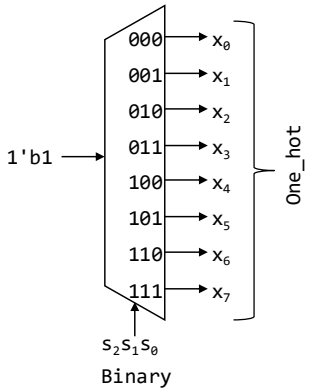
1) Solve at least one of the following problems.
2) Submit your solution on GitHub. (HDL and schematic when required)
3) Review and give feedback on your colleague's solutions that have been assigned to you in Canvas.

QUESTIONS

1.1 QUESTION 1

Model using HDL a look-up table **using a 1-to-16 de-multiplexer** that can act as a 4-bit one-hot (active-high outputs) decoder. The design should take as input a 4-bit binary value and output the one-hot equivalent. The definition of the module is given below.

```
module decoder (
  input  logic [3:0] binary,
  output logic [15:0] one_hot
);
  // …
  // Add your description here
  // …
endmodule
```



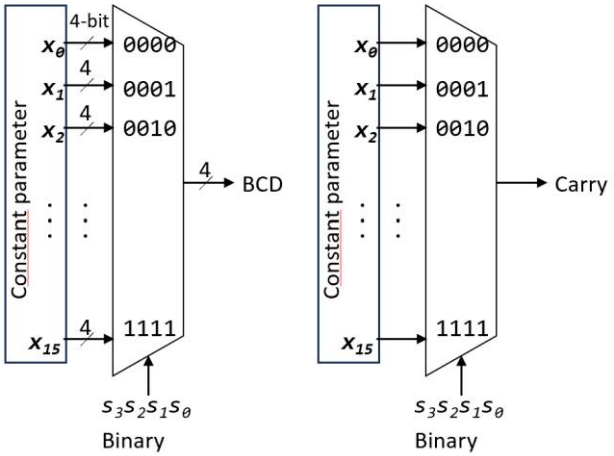
1.2 QUESTION 2

Model using HDL a design that uses **16-to-1 multiplexers** and implements a 4-bit binary to BCD (Binary-coded decimal) encoder. The BCD encoding can be seen in the following table. The module definition is also seen below.

Decimal	Binary	BCD				Carry
		8	**4**	**2**	**1**	
0	0000	0	0	0	0	0
1	0001	0	0	0	1	0
2	0010	0	0	1	0	0
3	0011	0	0	1	1	0
4	0100	0	1	0	0	0
5	0101	0	1	0	1	0
6	0110	0	1	1	0	0
7	0111	0	1	1	1	0
8	1000	1	0	0	0	0
9	1001	1	0	0	1	0
10	1010	0	0	0	0	1

11	1011	0	0	0	1	1
12	1100	0	0	1	0	1
13	1101	0	0	1	1	1
14	1110	0	1	0	0	1
15	1111	0	1	0	1	1

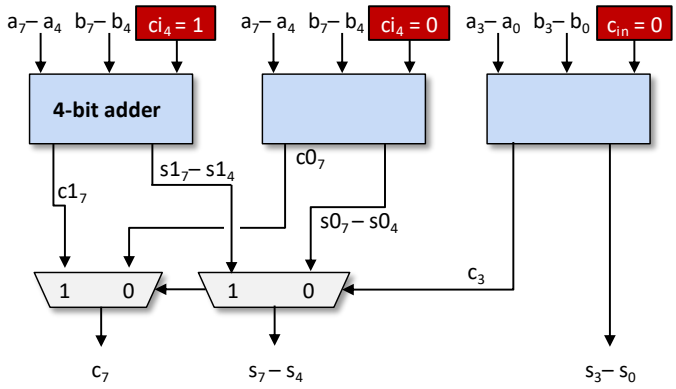
```
module bin2bcd (
  input  logic [3:0] binary,
  output logic [3:0] bcd,
  output logic carry
);
  // …
  // Add your description here
  // …
endmodule
```



1.3 QUESTION 3

Model a design using HDL that implements an 8-bit unsigned carry select adder (CSA). The CSA should be built using 4-bit adders. The design of the adder is given to you. The definition of the module can be seen below.

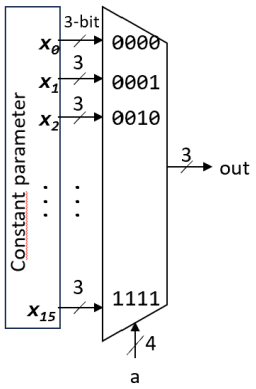
```
module CSA_8 (
  input  logic [7:0] A, B,
  output logic [7:0] sum,
  output logic carry
);
  // …
  // Add your description here
  // …
endmodule
module adder_4 (
    input logic [3:0] A, B,
    output logic [3:0] sum,
    output logic carry
  );
  assign {carry, sum} = A+B;
endmodule
```



1.4 Question 4

Model a design using HDL that can count the number of 1s in an 4-bit binary number. For example, the `4'b0101` should output `3'b010`. The definition of the module is given below.

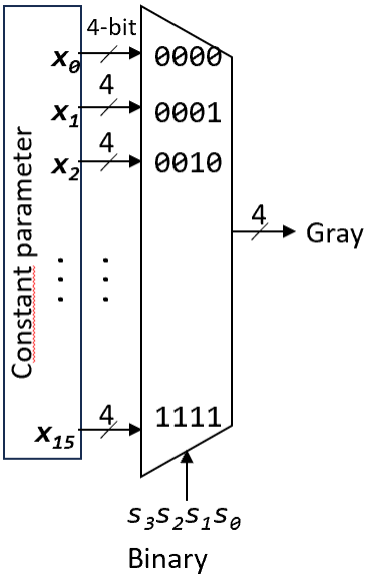
```
module count_1 (
  input  logic [3:0] a,
  output logic [2:0] out
);
  // …
  // Add your description here
  // …
Endmodule
```



1.5 Question 5

Model using HDL a design that uses a **16-to-1 multiplexer** and implements a 4-bit binary to gray encoder. The gray encoding can be seen in the following table. The module definition is also seen below.

Decimal	Binary	Gray
0	0000	0000
1	0001	0001
2	0010	0011
3	0011	0010
4	0100	0110
5	0101	0111
6	0110	0101
7	0111	0100
8	1000	1100
9	1001	1101
10	1010	1111
11	1011	1110
12	1100	1010
13	1101	1011
14	1110	1001
15	1111	1000




```
module bin2gray (
  input  logic [3:0] binary,
  output logic [3:0] gray
);

  // …
  // Add your description here
  // …
Endmodule
```

1.6 Question 6

A) Model using HDL a design that can be used to implement an arithmetic right shift. Your design has to be parametric and have N-bit input, where N ≥ 4. It should also allow to shift up to 3 positions. Do **not** use the '>>>, <<<' operators.

```
module ArithmeticRightShifter #(parameter N) (
  input  logic [N-1:0] input_data,
  input  logic [1:0] control,
  output logic [N-1:0] shifted_result
);
```

B) Draw a schematic of your design. Assume a value for the parameter N=5.

1.7 Question 7

A) Model using HDL a signed N-bit multiplier. The design has to be parametric with 2 inputs of N-bit, where N>3. Your design should use half adders and/or full adders (given below).

B) Draw the schematic of your design. Assume a value for the parameter N=5.

```
module full_adder (
    Input logic a,b,c_in,
    Output logic c_out, s
  );
  logic s1,c1,c2;
  half_adder ha1(a,b,s1,c1);
  half_adder ha2(s1,c_in,c2,s);
  assign c_out = c1|c2;
endmodule
```

```
module half_adder (
    Input logic a,b,
    Output logic c_out, s
  );
  assign s = a^b;
  assign c_out = a&b;
endmodule
```

```
module multiplier #(parameter N) (
    Input logic [N-1:0]a,b,
    output logic [2*N-1:0] product
  );
```

1.8 Question 8

A) Model using HDL a design that can multiply 6 N-bit numbers and add the results. The design should be parametric with parameter N. You must ensure that your design does not overflow or underflow and assign the correct bit width for all intermediate and output signals. The design should implement the following equation:

$$out = \sum_{k\in\{0,2,4\}} X_k \cdot X_{k+1}$$

B) Draw a schematic of your design. Assume a value for the N=4 parameter.

```
module sum_prod #(parameter N) (
    input logic [N-1:0] X [5:0],
    output logic [2N+2:0] result
  );
```

1.9 Question 9

A) Model using HDL an unsigned multiplier that can be used to multiply two 16-bit numbers. The multiplier should be built using components of multiple 4-bit multipliers. The 4-bit multiplier can be modelled behaviorally by using the (*) operator.

B) Draw a schematic of your design.