2-point Homework 1

Use the last 3 digits of your personal number (PN) to select which question you have to answer. Modulo divide your 3 digits with the number of questions, in this case, 2, plus 1. The resulting number is the question that you should answer.

!!! (PN mod 2) + 1 !!!

For example, if your PN ends in 730, then (730 mod 2) + 1 = 1, meaning you must answer question 1.

Make sure you comment on the top of each file you submit, your PN, name and the question you are answering.

!!! Important !!!

If you answer the wrong question, your submission will be invalid.

We can call you to explain your solution. If you cannot explain your answer, the homework will be invalid!

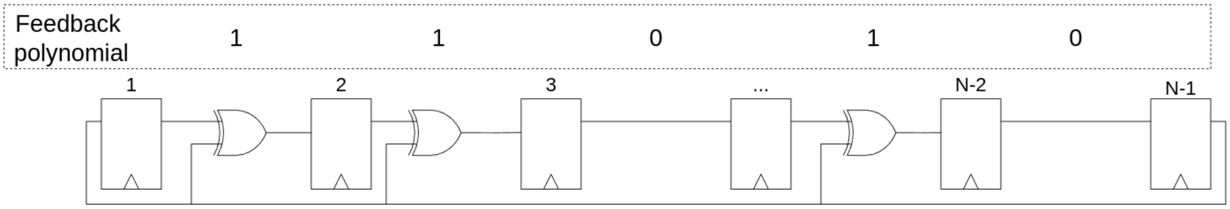
KTH has a zero-tolerance policy against cheating.

https://www.kth.se/en/student/stod/studier/fusk-1.997287

If you have questions or need clarifications, you can ask in the discussion forum in Canvas.

Questions

1.1 Question 1



Model using HDL an N bit LFSR where the feedback polynomial is specified by a parameter. Note that the Nth bit of the feedback polynomial is not in the parameter, as it is always 1. So, an N-bit LFSR will have an N-1 feedback parameter.

Instantiate the D flop modules provided and instantiate the XOR gates between them according to the parameter. Use an initialization reset value specified as a parameter.

Hint: you can define logic or bit parameters as in this example:

module <name> #(parameter logic [3:0] reset_val = 4'b0000)

Module parameters

Name	Width	Description
N	—	Length of the LFSR
feedback_poly	N-1	Coefficients of the feedback polynomial
reset_value	N	Reset value

Module pinout

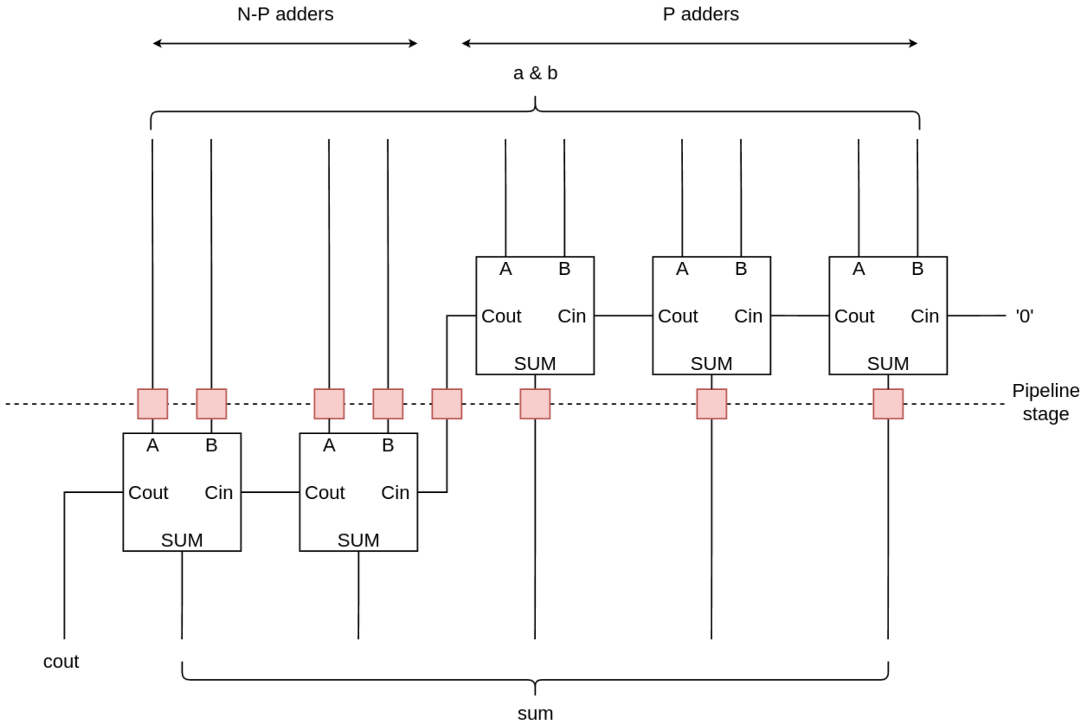
Name	Direction	Width	Description
clk	in	1	Clock signal
rstn	in	1	Active low reset
output	out	N	Value of the LFSR

Deliverables:

A) HDL implementation of the design

```
module d_flop #(parameter logic reset_val = 1'b0)(
  input logic clk, rst_n, d,
  output logic q
);
  always_ff @(posedge clk) begin
    if (!rst_n) begin
      q <= ~~rv~~reset_val;
    end else begin
      q <= d;
    end
  end
endmodule
```

1.2 QUESTION 2



Model using HDL and using full adders, a N bit adder with a pipeline where the split point of the pipeline is given by parameter P. This means that there are P full adders in the first pipeline stage and N-P in the second stage. The red boxes in the schematic represent the register needed to "cross" between pipeline stages.

Module parameters

Name	Width	Description
N	—	Width of the input operands
P	—	Splitting point of the pipeline stages

Module pinout

Name	Direction	Width	Description
clk	in	1	Clock signal
rstn	in	1	Active low reset
a, b	in	N	Input operands
sum	in	N	Output sum
cout	out	1	Carry output

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

module half_adder (
    input a,b,
    output c_out, s
  );
  assign s = a^b;
  assign c_out = a&b;
endmodule
```

Deliverables:

A. HDL implementation of the design