2-point Homework 3

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, 3, 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 3) + 1 = 2, meaning you must answer question 2.

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

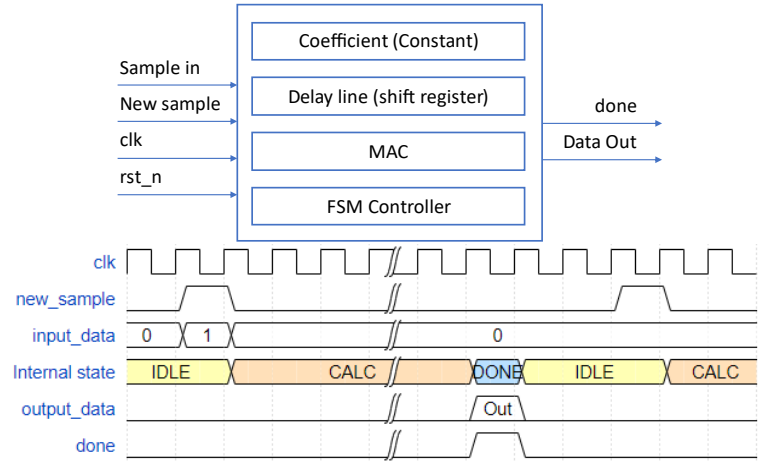
1.1 QUESTION 1

Design a parametric FIR filter in using SystemVerilog with the following considerations.

- The filter should have 7 taps (i.e. 7 coefficients)
- Sample data width of W bits
- Coefficient data with of C bits
- Data represented in two's complement
- The filter uses a **single** MAC (Multiply-ACcumulate) unit
- The input data should be sampled when the new_sample flag is high and stored in a shift register.
- The shift register has an input port and an output port to implement the delay line.
- The coefficients can be stored in a ROM and hard-coded as constants in the RTL code (i.e. it is not necessary to load them during runtime). You can use 1,3,7,15,31,15,7,3,1 as coefficients to simplify the testing.

```
if(new_sample):
  //shift the new sample in the first location of the delay line
  shift(delay_line, new_sample)
  output = 0
  For (i=0; i ≤ 6; j++)::
    output = output + delay_line[i]*coeff[i]
  end
  done = 1
end
```

Below you can find an example of the structure and a timing diagram of a computation.



Name	Width	Description
W	--	With of the input data
C	--	Width of the coefficients

Module pinout

Name	Direction	Width	Description
clk	in	1	Clock signal
rstn	in	1	Active low reset
new_sample	in	1	1 when a new sample is present
sample	in	W	Input sample
output_data	out	*	Result of the FIR
done	out	1	1 when the output is available

*Note: calculate the required size of the output considering the number of taps so that there is no overflow or lost information

1.2 QUESTION 2

Implement an FSMD using SystemVerilog that can generate an address pattern in the form of 2 nested loops afine functions.

Take the following example that is commonly found in applications such as linear algebra, signal processing or machine learning. The FSMD should implement the following algorithm.

```
For (i=0; i ≤ C; i++):
  For (j=0; j ≤ B; j++):
     address = A + D*j + E*i;
  end
end
```

Implement an FSMD that can perform this computation autonomously.

If there is an overflow in the address, set a warning flag for 1 clock cycle, and allow the computation to continue. Consider all the A, B, C, D, E, and address to be unsigned.

Module parameters:

Name	Description
A	Initial value of 1st level loop
B	Range of 1st level loop
C	Range of 2nd level loop
D	Step of 1st level iteration
E	Step of 2nd level iteration
ADDR_WIDTH	Width of the addresses and parameters

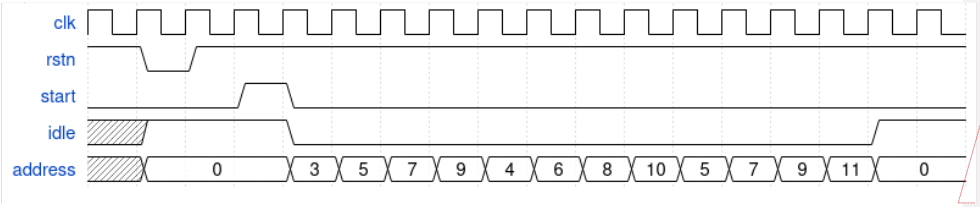
Module pinout

Name	Direction	Width	Description
clk	in	1	Clock signal
rstn	in	1	Active low reset
start	in	1	When driven to 1, the operations should start
idle	out	1	1 when no addresses are computed, 0 when operating
address	out	ADDR_WIDTH	Value of the generated address
overflow	out	1	Overflow warning flag

You can find an example of a timing diagram for a computation of the addresses where the parameters were as follows:

- `A = 3`
- `B = 4`
- `C = 3`
- `D = 2`

- E = 1



Commented [JG1]: {signal: [
{name: 'clk', wave: 'p.....'},
{name: 'rstn', wave: '101.....'},
{name: 'start', wave: '0..10.....'},
{name: 'idle', wave: 'x1..0.....1.'},
{name: 'address', wave: 'x2..2222222222222.',
data: [0,3,5,7,9,4,6,8,10,5,7,9,11,0]},

]}

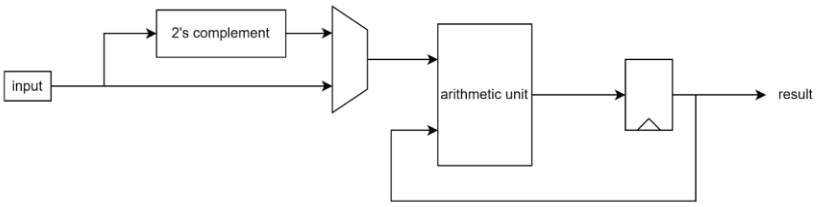
Python code:

```
A = 3
B = 4
C = 3
D = 2
E = 1
for i in range(C):
  for j in range(B):
    address = A + D*j + E*i
    print(address);
```

1.3 Question 3

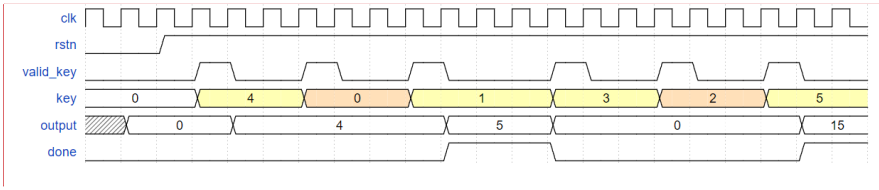
Design an FSM+D that implements a simple calculator that can do addition or substruction. The user inputs a number, followed by an operation (0: addition, 1: subtraction, 2:multiplication), then followed by another number. After the operation is completed and the result generated, the user can continue with the next sequence. Hint: the FSM should control the 2's complement mux and the ALU.

Note: you can use the arithmetic operators that are provided by the language to implement the addition.



Name	Direction	Width	Description
clk	in	1	Clock signal
rstn	in	1	Active low reset
key	in	4	The "value" of the pressed key
valid_key	in	1	1 when a key is pressed
Operation	in	2	0 for addition and 1 for subtraction 2 for multiplication
result	out	5	Result of the operation
Valid results	out	1	1 if the output is valid, 0 if the output is not valid

Below you can see a timing diagram to perform the operation 4 + 1 * 3. The yellow inputs are numbers and the orange are the operators



Commented [JG2]: {signal: [
{name: 'clk', wave: 'p.....'},
{name: 'rstn', wave: '101.....'},
{name: 'valid_key', wave: '0..10.10.10..10.10.'},
{name: 'key', wave: '2..3..4..3...4..3..',
data: [0,4,0,1,2,3]},
{name: 'output', wave: 'x2..2.....2.....2.',
data: [0,4,5,15]}

]}