LELEC 2531 - Evaluation 2 - Project
November 2022

Recommendations

During this evaluation, you may only use the reference book, the supports of this courses (slides and tutorials available on Moodle) and your personal homework. You may not access Internet.

See INGInious for the exact definition of the modules. When you submit your code on INGInious, don't forget to include the module definition !

Your grade will be based on the number of attempts on INGInious. You do not have to provide a PDF report. Because of the limited time available, you may choose not to simulate your circuit on ModelSim. It's your choice. Nevertheless, we strongly advise you to compile it to check that there is no syntax error before submitting to INGInious.

2A - ALU

You start from your homework 4 and add some new functionalities : XOR and bitwise Clear operations (the Clear operation sets to 0 specific bits of a - the bits to be cleared are equal to 1 in b)

```
module Homework_4(
      input  logic [31:0] a, b,
      input  logic [2:0]  ALUControl,
      output logic [31:0] Result,
      output logic [3:0]  ALUFlags);

The ALUControl is defined as following :
      000 : a + b        (Addition)
      001 : a - b        (Substraction)
      100 : a AND b      (Bitwise AND)
      101 : a OR b       (Bitwise OR)
      110 : a XOR b      (Bitwise XOR)
      111 : a AND NOT(b) (Bitwise Clear)

The ALUFlags are not modified :
      3 : N=1 if the result is negative
      2 : Z=1 if the result is 0
      1 : C=1 if the adder produces a carry out
      0 : O=1 if the adder results in overflow

C and O must be set to 0 for operations other than addition and substraction
```

2B - ARM single-cycle processor

You start from your homework 6 where you added to the ARM sigle-cycle processor the `EOR` and `LDRB` instrucitons. You will have now to add the `BIC` instruction (bitwise Clear). See Table B.1 for details.

Table B.1 Data-processing instructions—Cont'd

cmd	Name	Description	Operation
$I = 0$ AND ($sh = 10$)	`ASR Rd, Rm, Rs/shamt5`	Arithmetic Shift Right	`Rd ← Rm>>>Src2`
$I = 0$ AND ($sh = 11$; $\text{instr}_{11:7,\ 4} = 0$)	`RRX Rd, Rm, Rs/shamt5`	Rotate Right Extend	`{Rd, C} ← {C, Rd}`
$I = 0$ AND ($sh = 11$; $\text{instr}_{11:7} \neq 0$)	`ROR Rd, Rm, Rs/shamt5`	Rotate Right	`Rd ← Rn ror Src2`
1110	`BIC Rd, Rn, Src2`	Bitwise Clear	`Rd ← Rn & ~Src2`
1111	`MVN Rd, Rn, Src2`	Bitwise NOT	`Rd ← ~Rn`

To test your implementation, we provide the .s and .hex files with BIC instructions. You do not need to use the ARM toolchain.

2C - ARM multicycle processor

You start from your last homework 7 where you implemented the `STR` and `B` instructions. You must now add the `TEQ` instruction. See Table B.1 for details.

Use the MyTestbench.sv, the .s and .hex files to test your design. Ensure that you have the most efficient microarchitecture in terms of number of cycles.

Table B.1 Data-processing instructions

cmd	Name	Description	Operation
0000	`AND Rd, Rn, Src2`	Bitwise AND	`Rd ← Rn & Src2`
0001	`EOR Rd, Rn, Src2`	Bitwise XOR	`Rd ← Rn ^ Src2`
0010	`SUB Rd, Rn, Src2`	Subtract	`Rd ← Rn - Src2`
0011	`RSB Rd, Rn, Src2`	Reverse Subtract	`Rd ← Src2 - Rn`
0100	`ADD Rd, Rn, Src2`	Add	`Rd ← Rn+Src2`
0101	`ADC Rd, Rn, Src2`	Add with Carry	`Rd ← Rn+Src2+C`
0110	`SBC Rd, Rn, Src2`	Subtract with Carry	`Rd ← Rn - Src2 -` $\overline{\text{C}}$
0111	`RSC Rd, Rn, Src2`	Reverse Sub w/ Carry	`Rd ← Src2 - Rn -` $\overline{\text{C}}$
1000 ($S = 1$)	`TST Rd, Rn, Src2`	Test	`Set flags based on Rn & Src2`
1001 ($S = 1$)	`TEQ Rd, Rn, Src2`	Test Equivalence	`Set flags based on Rn ^ Src2`
1010 ($S = 1$)	`CMP Rn, Src2`	Compare	`Set flags based on Rn - Src2`
1011 ($S = 1$)	`CMN Rn, Src2`	Compare Negative	`Set flags based on Rn+Src2`
1100	`ORR Rd, Rn, Src2`	Bitwise OR	`Rd ← Rn | Src2`

I remind you that you are not obliged to simulate your circuit if time is limited. You can simply compile it to check that there are no syntax errors.

What to Turn In

Validate your design on INGInious (the evaluation will be done through INGInious and will be based on the number of attempts)

Upload on Moodle 1 file:

- A ZIP file untitled "Evaluation-2-Simu-StudentLastName-StudentFirstName.zip" that includes all the simulation files (Define 3 different projects in Quartus or ModelSim – Put the files of each project in a specific folder)