Comp E 475

Digital Systems

Homework 7

Student: uta buziashvili

Red ID #: 822059947

12.22.2020

Contents

[Task Description 2](#_Toc22460835)

[Solution 2](#_Toc22460836)

[Simulation & Verification 2](#_Toc22460837)

[Comparison 3](#_Toc22460838)

[Conclusion 3](#_Toc22460839)

# Task Description

*Desing ALU, that implements the hardware necessary to execute the following instructions:*

*data processing: AND, XOR, ORR, SUB, RSB, ADD, CMP*

*memory access: STR, LDR (standard addressing mode)*

*branching: B*Solution

*To carry out the task, I created four inputs (A, B, op and cmd) and 2 outputs (result and flag) and wrote down all the conditions. 4th and 3rd bits of flag represent negative and zero flags and as they are generated in the same manenr in every case, I wrote the conditions for them separately. As for carry and overflow flags (that are represented by the second and the first bits of flag) they can be generated after addition, subtraction and comparison(it is subtraction in fact too). So we get carry if the inputs are unsigned and if addition carry gets 1 if result is more than it is avalable to be written by 32 bits. If subtraction(A-B),carry gets 1 if A>B. Oveflow is generated when inputs are signed values. And if addition A[31]==B[31] &&A[31]!=result[31] this cn=ondition generates oveflow high and if subtraction -A[31]!=B[31]&&result[31] !=B[31].*

*Here is the code itself:*

*`timescale 1ns / 1ps*

*module lab7uta(*

*input[31:0] A\_in, B\_in,*

*input [5:0] cmd,*

*input [1:0] op,*

*output reg[31:0] result=0,*

*output [3:0] flag //flags for zero, negative, carry and overflow*

*);*

*reg[1:0] carry\_ovf;*

*always @(\*)*

*begin*

*case (op)*

*0: begin //data*

*case(cmd)//AND, XOR, ORR, SUB, RSB, ADD, CMP*

*0: begin //and*

*result=A\_in&B\_in;*

*carry\_ovf=0;*

*end*

*1: begin //xor*

*result=A\_in^B\_in;*

*carry\_ovf=0;*

*end*

*2: begin //sub*

*result=A\_in-B\_in;*

*if(A\_in<B\_in)*

*carry\_ovf[1]=1; //generating carry*

*else*

*carry\_ovf[1]=0;*

*if(A\_in[31]!=B\_in[31]&&result[31] !=A\_in[31])*

*carry\_ovf[0]=1; //generating overflow*

*else*

*carry\_ovf[0]=0;*

*end*

*4: begin //add*

*result=A\_in+B\_in;*

*if((A\_in>result)||(B\_in>result))*

*carry\_ovf[1]=1; //for unsigned*

*else*

*carry\_ovf[1]=0;*

*if(A\_in[31]==B\_in[31] &&A\_in[31]!=result[31])*

*carry\_ovf[0]=1;*

*else*

*carry\_ovf[0]=0; //for signed*

*end*

*3: begin //rsb*

*result=B\_in-A\_in;*

*if(A\_in>B\_in)*

*carry\_ovf[1]=1;*

*else*

*carry\_ovf[1]=0;*

*if(A\_in[31]!=B\_in[31]&&result[31] !=B\_in[31])*

*carry\_ovf[0]=1;*

*else*

*carry\_ovf[0]=0;*

*end*

*10:begin //cmp*

*result=A\_in-B\_in;*

*if(A\_in<B\_in)*

*carry\_ovf[1]=1;*

*else*

*carry\_ovf[1]=0;*

*if(A\_in[31]!=B\_in[31]&&result[31] !=A\_in[31])*

*carry\_ovf[0]=1;*

*else*

*carry\_ovf[0]=0;*

*end*

*12:begin //orr*

*result=A\_in|B\_in;*

*carry\_ovf=0;*

*end*

*default: begin result=0;*

*end*

*endcase*

*end*

*1: begin //memory*

*if(cmd[3]==1)*

*result=A\_in+B\_in;//Bis anoffset, A is a base adress*

*else*

*result=A\_in;*

*if((A\_in>result)||(B\_in>result))*

*carry\_ovf[1]=1; //for unsigned*

*else*

*carry\_ovf[1]=0;*

*if(A\_in[31]==B\_in[31] &&A\_in[31]!=result[31])*

*carry\_ovf[0]=1;*

*else*

*carry\_ovf[0]=0; //for signed*

*end*

*2: begin //branch*

*result=A\_in+B\_in; //A next instruction, B how many steps to jump over*

*if((A\_in>result)||(B\_in>result))*

*carry\_ovf[1]=1; //carry when unsigned*

*else*

*carry\_ovf[1]=0;*

*if(A\_in[31]==B\_in[31] &&A\_in[31]!=result[31])*

*carry\_ovf[0]=1;*

*else*

*carry\_ovf[0]=0; //overflow when signed*

*end*

*default: begin result=0;*

*end*

*endcase*

*end*

*assign flag[1:0] = carry\_ovf;*

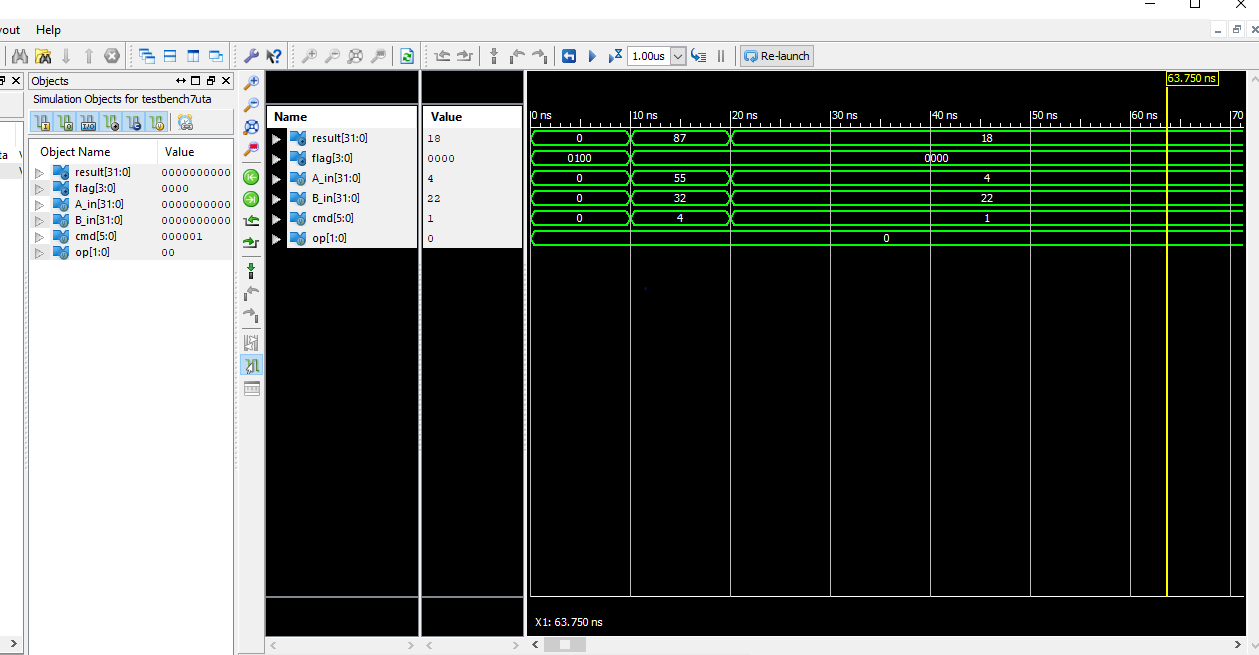
*assign flag[2]= (result==0) ? 1:0;*

*assign flag[3]= (result[31]==1) ? 1:0;*

*endmodule*

# Simulation & Verification

* *To simulate I gave different values to the inputs and checked the results*
* *Simulation of the code:*



* *Here is the link to my github repository :*

*https://github.com/NinoNonikashvili/470LabFiles/tree/master/ce475hw7*