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

*I have made four imputs which are AB op and cmd and also I made two outputs result and flag and wrote all conditions. Firstly I want to mention that 3rd and 4th bits are representing of negative and zero flags and wrote conditions for them independently carry and overflow flags can be generated after addition subtraction and comparison, and it is clear that if addition carry gets 1 and its result is more than available it will be written in 32bits.*

*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;*

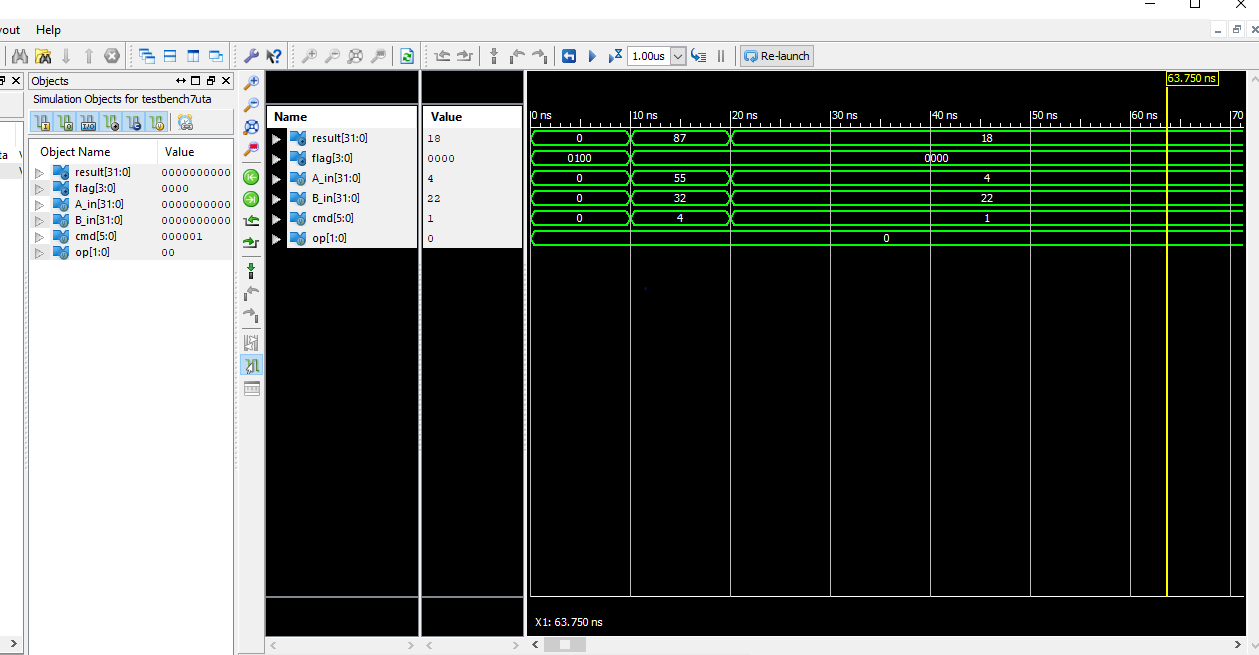
*assign flag[2]= (result==0) ? 1:0;*

*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/uta9991/labs*