RTL Design

Amir Reza Nekooei

Electrical and Computer Engineering
School of Engineering Colleges, Campus 2
University of Tehran, 1450 North Amirabad, 14395-515 Tehran, Iran

Amirrezank@gmail.com

Abstract**—In this assignment we design a sign integer divider in register transfer level. First step is designing a datapath which includes some components and second step is designing a controller for proper function of this module. We use VHDL to describe the hardware and verify our design.**

Keywords**— RTL, design, datapath, controller, divider, VHDL.**

I. INTRODUCTION

The purposes of this assignment is designing and verifying a sign integer divider. In implementation of this hardware we use subtraction for division. Fig. 1 shows division algorithm that implements with subtraction.

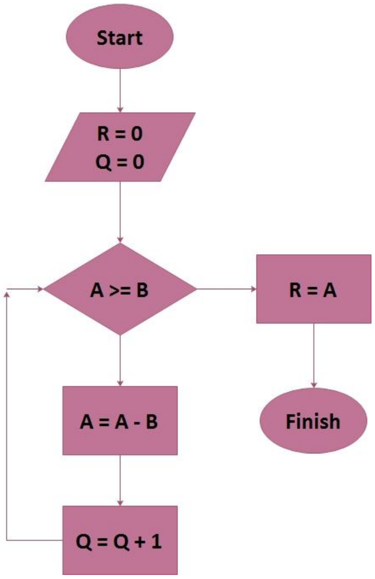

Fig. 1 Division algorithm that implement with subtraction

In this implementation, we design components of the datapath and then design a controller for this datapath.

The remainder of this report is organized as follows. Section II gives an overview of datapath. In section III, we design controller. In section IV, we verify the design. Finally, section V concludes the assignment.

II. DATAPATH

In this section we design datapath for sign integer divider. For this purpose we need some components that listed below.

A. *Simple Register*

This component is used for sampling of input for divide operation.

B. *Subtracter*

This component does subtract operation. In final step output of this module is remainder.

C. *Comparator*

This module determines the last step of divide operation, in this way, when divider is less than divisor the output of this module is zero.

D. *Counter*

This module counts the number of time that subtraction operation is executed and the output of this module in final step is quotient.

E. *Tristate*

We use this module to prevent extra switching on output to save power.

In Fig. 2 we show datapath that implements divide operation so that first *go* signal should be one and when *go* is zero *MUX* selects *Abus* and puts this on *outMux*. After one clock cycle A and B without sign bit is put on subtracter and comparator input. If B input is equal to zero (*BIsZero* = ‘1’), *ready* signal becomes one and *Qbus* and *Rbus* remain high impedence, otherwise *MUX* selects *ASubB* and puts this on *outMux*. This operation would be executing until output of *Comparator* becomes zero. In this moment *SimpleReg:A* output is remainder and *Counter* output is quotient. With finished divide operation *ready* signal is one for one clock cycle.

III. CONTROLLER

In this section we design a controller for divider. For this work we use Hoffman model. In Fig. 3 control data flow of the finite state machine module is shown. This module activates all control signals and manages data flow of our datapath.

In Fig. 4 we show state diagram of finite state machine. This is a mealy machine that its order of inputs and outputs of finite state machine is shown in (1).

(go,rsltCMP)/ (ready, selMux, loadInputA, loadinputB, parLopadR, parloadQ, rstCount, upCount) (1)

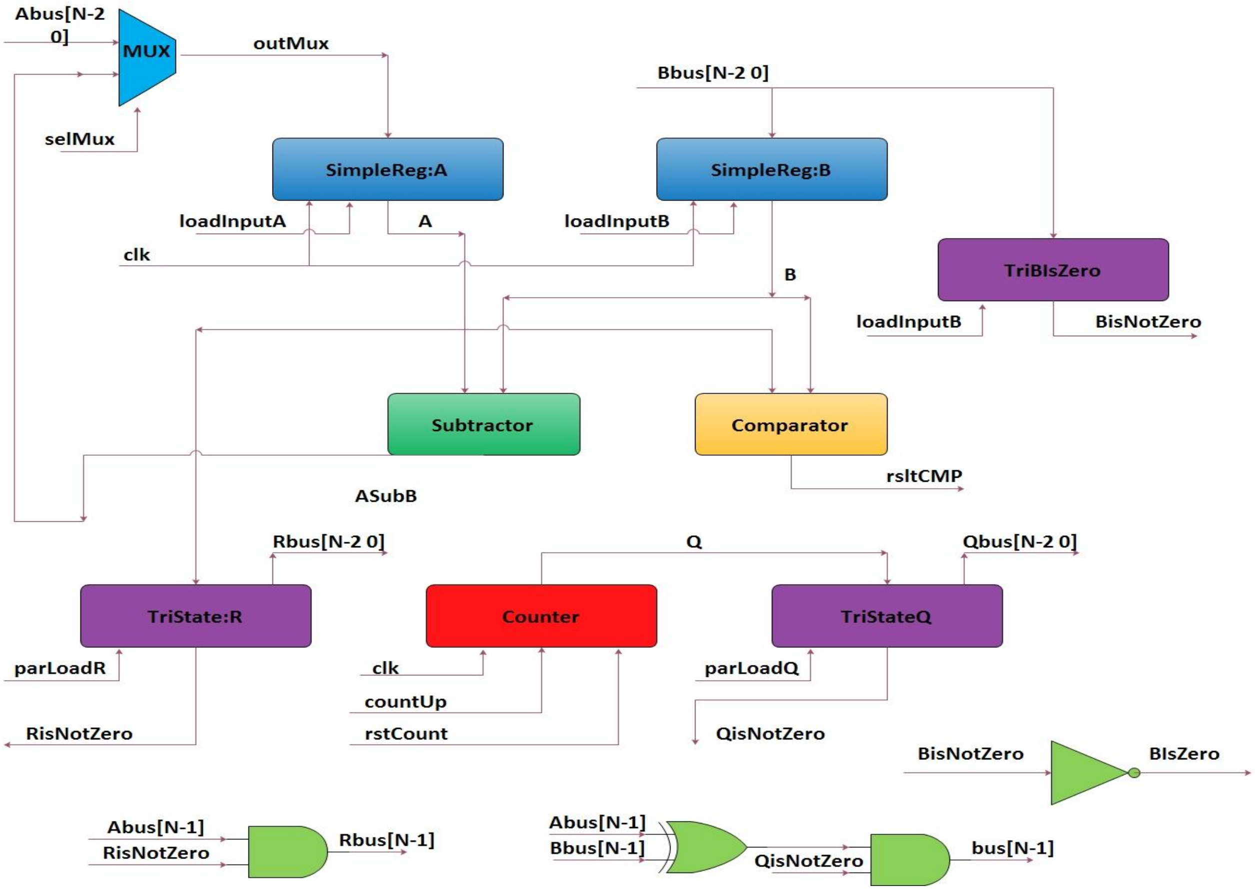

Fig. 2 Datapath

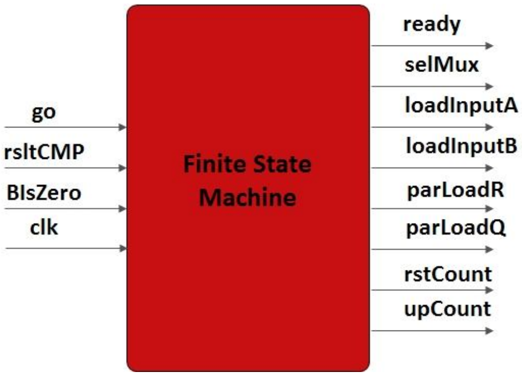

Fig. 3 Finite state machine

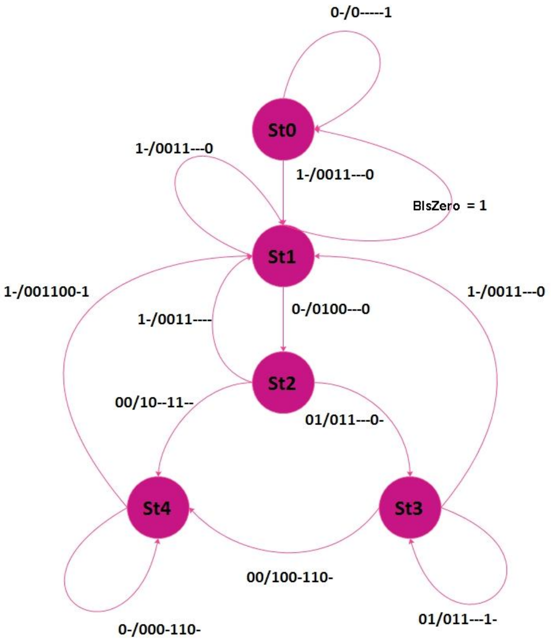

Fig. 4 state diagram of finite state machine

IV. DESIGN VERIFICATION

In this section we simulate hardware and describe functionality of sign integer divider. First, in implementing this hardware we assume that the remainder can be negative. If we had assumed remainder is always positive, a small change is needed to deal with this assumption. Second, in start of divide operation we assume that *go* signal is high; the divider starts sampling from input and when *go* signal changed to low divide operation starts. Third, we assume when divide operation is done *ready* signal would be high for one clock cycle. Also we assume the number is sign magnitude system for sign calculation and implement hardware in N-configuration. It means this divider can work in any number of bits for input data.

In simulation we write a test bench that read test data from a file, in this way, all inputs and desired outputs get from file and then inputs data are fed to the hardware and outputs are compared with desired outputs. If the result is wrong a massage will be printed in transcript.

To generate test data we write a program in MATLAB that generates proper inputs and outputs of test hardware.

All VHDL files and their functionality are described as follows:

- Trunk/ComparatorN

This module compares two N-bit numbers. If *AInput* is more than *BInput, rsltCMP* changes to one otherwise *rsltCMP* stays zero.

- Trunk/CounterN

This module is an N-bit counter. If *upCount* is one with positive edge of clock *parOut* would be increased.

- Trunk/FSM

This module is a finite state machine.

- Trunk/ Multiplexer2to1N

This module is a multiplexer 2to1 that each its line is N-bit.

- Trunk/ SimpleRegisterN

This module is a simple register. If *parLoad* is one then parOut is parIn.

- Trunk/ SubtractorN

This module is a subtracter.

- Trunk/ TriStateN

This module is a tristate buffer which its input and output is N-bit.

- Trunk/ SignIntegerDividerN

In this module all components are connected together.

V. CONCLUSIONS

In this document we design a sign integer divider in RTL. First, we design the datapath and then design the controller. To verify design we write a test bench in VHDL [1] and read test data from file and compare outputs with desired value.

REFERENCES

[1] Z.Navabi, VHDL: Modular Design and Synthesis of Cores and Systems, 3nd ed., McGraw-Hill Professional, 2007.