

CSCE 3301

Dr. Cherif Salama

Yasmina Mahdy

Rana Taher

Single Cycle implementation Project Report

Brief description of project

This project is an implementation of the single cycle RISC-V processor using verilog that will be supporting the instructions of the RV32I extension.

Structure

- RISC_Top.v
 - RISC V.v: Has instances of all the modules needed to complete the full datapath.
 - RCA.v: Three instances for the pc calculations.
 - Registe load rst.v: used to store the current value of the pc.
 - BranchCu.v: used to determine whether the branch instruction will be executed or not.
 - InstMem.v: Stores the instructions of the program
 - ControlUnit.v: simulates the control unit of the datapath.
 - RegFile.v: simulates the register file by storing the contents of each register.
 - rv32_ImmGen.v: generates the needed immediates by all the instructions.
 - ALUcu.v: generates the needed ALUSel needed by the ALU to figure out which instruction to compute.
 - prv32_ALU.v: simulates how the ALU computes each instruction needed by it and generates all the flags needed
 - n_bit_shifter.v: This computes the output of the shifting instructions.
 - DataMem.v: simulates the data memory and stores the content of the data memory.
 - Four_Digit_Seven_Segment_Driver.v: used to display the pc on the seven segment display.

Problems and solutions

At the beginning of the testing process a number of issues were encountered.

They were as follows:

i1: pc not working correctly

s1: fixed the default values for the registers from 8 to 12 bits (PC size to accommodate 4KB memory)

i2: sra not working correctly; not sign-extending

s2: declared the input to be shifted as signed input

i3: I-type shifting not working as expected; shamt being read incorrectly

s3: Added a multiplexor to set shamt as rs2 for R-type and instr[24:30] for I-type

i4: auipc not working correctly; pc value is too small (12 bits) to use the output of target adder as auipc resulty

s4: added separate adder and wire for auipc result

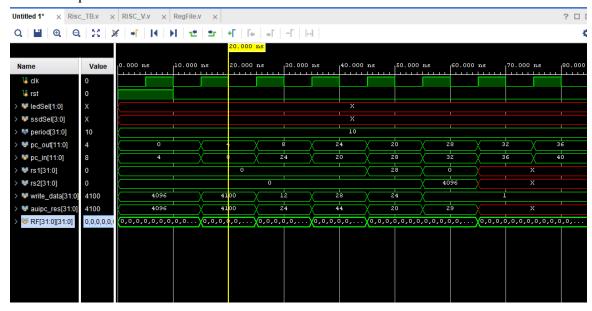
Waveforms Snippets and Test Case Descriptions

Test 1

Purposes: tests all R and I instructions except for load and jalr

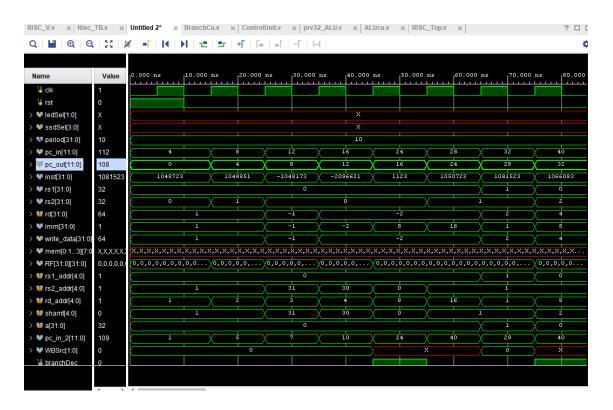
RISC_V.v × Risc_	_TB.v ×	Untitled 2* ×	Registe_load_rst	i.v x prv32_A	LU.v × n_bit_	shifter.v ×	ALUcu.v × II	nstMem.v × RC/	A.v ×	? 🗆	
Q 💾 🙉 😡	50 3	₹ +Γ	H 12 2r	+[[+ +[-[[⊷						
Name	Value	0.000 ns	15.000 ns	10.000 ns	15.000 ns	20.000 ns	25.000 ns	,30,000 ns	35.000 ns	40.00	
	• uiuo	0.000 hs	3.000 HS	10:000 HS	15.000 HS	20:000 113	20.000 HS		55.000 HS	40.00	
U dk	0										
₩ rst	0					х					
> W ledSel[1:0]	X										
> W ssdSel[3:0]	X	X									
> 15 period[31:0]	10					10					
> W pc_in[11:0]	72		4		Ĭ		_{	12	16		
> V pc_out[11:0]	68	0			↑ ———			8	12		
> W inst[31:0]	5047		3145875		108.		10	74790707	-3112813		
> 🕊 rs1[31:0]	0	(0		¥:			0	6		
> W rs2[31:0]	0		0		¥:	3		6	, °		
> 💆 rd[31:0]	4096		3		X			-6	3		
> [®] imm[31:0]	4096		3		X		_X	1025	-3		
> 🐶 write_data[31:0]	4096		3		X	,		-6	3		
> W mem[0:13][7:0	X,X,X,X,X,	x,x,x,x,x,x,x	,x,x,x,x,x,x,x,x,x,x,x,x,x,x,x,x,x,x,x	.x,x,x,x,x,x,x,	x,x,x,x,x,x,x,	,x,x,x,x,x,x	,x,x,x,x,x,	x,x,x,x,x,x,x,x,	<,x,x,x,x,x,x,x,x,	х,х.	
> W RF[31:0][31:0]	0,0,0,0,0,0	0,0,0,0,0,0,0	0,0,0,0,0,0,0,0	,0,0,0,0,	0,0,0,0,0,0,0	0,0,0,0,0,	. 0,0,0,0,0,0	,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,	0,0,0,0,0,0,0,	,0,0.	
> 👑 rs1_addr[4:0]	0		0		X		*	Ö	1		
> 👹 rs2_addr[4:0]	0		3		X		i		29		
> 👹 rd_addr[4:0]	7			1			*	2	1		
> 👹 shamt[4:0]	0			3			*	6	29		
> 👹 a[31:0]	0		0		<u> </u>	3	*	0	6		
> W pc_in_2[11:0]	68		3		<u> </u>		<u> </u>	1033	9		
> W WBSrc[1:0]	1					0					

Test 2
Purposes: tests all S and load instructions



Purpose: tests all branch instructions

Test 3

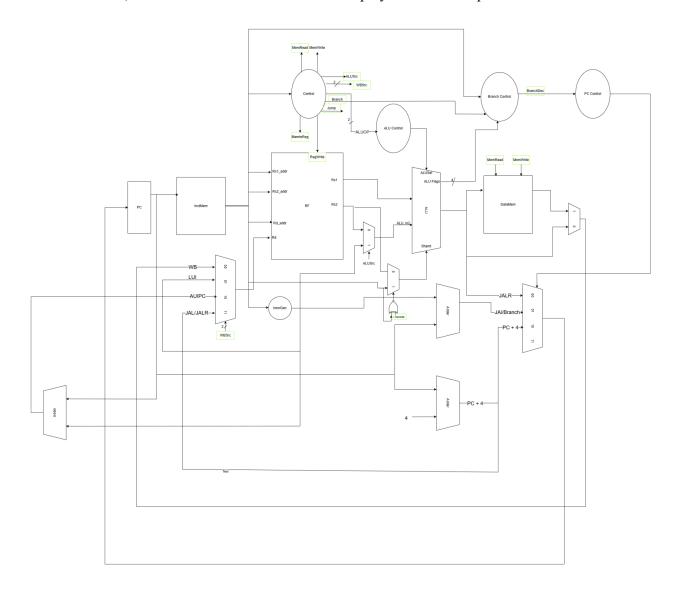


Test 4
Purpose: tests lui, auipc, jal, jalr

Untitled 1* × Risc	_TB.v ×	RISC_V.v ×	RegFile.v x													? 🗆
Q 💾 📵 Q	. 50 3	∀ + 	H 발 발	+ Γ [+	- ÷Γ	-[]-										- 1
				20.000	ns											
Name	Value	0.000 ns	10.000 ns	20.000	ns	30.000	ns	40.000	ns	50.000	ns	60.000	ns	70.000	ns	80.000
[™] dk	0															
[™] rst	0															
> W ledSel[1:0]	X		X													
> 😻 ssdSel[3:0]	X		x													
> 🐶 period[31:0]	10			10												
> W pc_out[11:0]	4	0	X	4	^	3	2		2			8	^	32		6
> W pc_in[11:0]	8	4	X		X2	4	2	0	^	8		2	X	36	^	10
> V rs1[31:0]	0	<u> </u>		0					X2	8	-		X	}		
> • rs2[31:0]	0	4096		100	0				· -	=	40	196	<u>X</u>) 1		
> write_data[31:0]		4096	^_	100	^	2	2	4	2	0		9		1	,	
> W auipc_res[31:0] > W RF[31:0][31:0]	0,0,0,0,0,0		, ο, ο χο, ο, ο,		^		<u> </u>		^			_	Va. a. a.			
7 WF[31.0][31.0]	0,0,0,0,0,0	0,0,0,0,0,0,0	70,0,0	,,,,,,,	0,0,0,	,,,,,,,,	0,0,0,	,,,,,,,,	0,0,0,	,,,,,,,,	.0,0,0,0	,,,,,,,,	No,0,0,	,0,0,0,0,	,0,0,0,0	,,,,,,,

Final datapath

Note that clock, reset and instruction bits are not displayed on the datapath



References

Computer Architecture Slides, Dr Cherif Salama, Fall2024.