VLSI II Homework 9

Subject: RISC-V Proccessor design notes and datasheet

Alper Akmermer - 040170230
Elif Beyza Erkan - 040106259

Contents

1. Design Process

2. Features

    a. Instruction Set Architecture

    b. Control Word

    c. Memory Structure

    d. Function Unit

    e. Pipeline Stages and Hazard Solutions

    f. Resource Utilization and Performance

Design Process

In our RISC-V design we heavily used sources given in lectures to create smaller modules such as function unit, RAM and register file. As for the greater structure, our lecture slides and textbook generally showed an example structure which was not truly compatible with RISC-V so we slightly changed our design through literature search to fit our needs better. Moving branch comparator outside of the ALU and later adding a whole new module dedicated to program flow control can be given as an example to these changes in design we made.

Generally, our first coding and RTL schematic checking was made in Vivado due to its more familiar and helpful interface especially when looking for simple errors. Later on, especially when we get to the pipelining stage, we started to use Xcelium and Genus more heavily for their speed and complexity.

Nearly all assignments of this project was separated into smaller subjects and was done separately by team members. Later on, when all parts are done, assembling, final touches and simulations were done together. Although we worked separately, there was idea exchanges and regular updates on both sides to make the design process go smoother for both sides.

Features

Instruction Set Architecture (ISA)

Instruction set is the standard instructions set that is in RISC-V official manual. [1] Functions and their code patterns are shown in Figure 1, in the right.

This instructions affect the processor via a designed control word. More detailed information about what exactly control word controls is given below.

31 27	26 25	24 20	19 15	14 12	11 7	6 0	
funct7		rs2	rs1	funct3	rd	opcode	R-type
imm[11:0]			rs1	funct3	rd	opcode	I-type
imm[11:5]		rs2	rs1	funct3	imm[4:0]	opcode	S-type
imm[12|10:5]		rs2	rs1	funct3	imm[4:1|11]	opcode	B-type
imm[31:12]					rd	opcode	U-type
imm[20|10:1|11|19:12]					rd	opcode	J-type

RV32I Base Instruction Set

imm[31:12]					rd	0110111	LUI
imm[31:12]					rd	0010111	AUIPC
imm[20|10:1|11|19:12]					rd	1101111	JAL
imm[11:0]			rs1	000	rd	1100111	JALR
imm[12|10:5]		rs2	rs1	000	imm[4:1|11]	1100011	BEQ
imm[12|10:5]		rs2	rs1	001	imm[4:1|11]	1100011	BNE
imm[12|10:5]		rs2	rs1	100	imm[4:1|11]	1100011	BLT
imm[12|10:5]		rs2	rs1	101	imm[4:1|11]	1100011	BGE
imm[12|10:5]		rs2	rs1	110	imm[4:1|11]	1100011	BLTU
imm[12|10:5]		rs2	rs1	111	imm[4:1|11]	1100011	BGEU
imm[11:0]			rs1	000	rd	0000011	LB
imm[11:0]			rs1	001	rd	0000011	LH
imm[11:0]			rs1	010	rd	0000011	LW
imm[11:0]			rs1	100	rd	0000011	LBU
imm[11:0]			rs1	101	rd	0000011	LHU
imm[11:5]		rs2	rs1	000	imm[4:0]	0100011	SB
imm[11:5]		rs2	rs1	001	imm[4:0]	0100011	SH
imm[11:5]		rs2	rs1	010	imm[4:0]	0100011	SW
imm[11:0]			rs1	000	rd	0010011	ADDI
imm[11:0]			rs1	010	rd	0010011	SLTI
imm[11:0]			rs1	011	rd	0010011	SLTIU
imm[11:0]			rs1	100	rd	0010011	XORI
imm[11:0]			rs1	110	rd	0010011	ORI
imm[11:0]			rs1	111	rd	0010011	ANDI
0000000		shamt	rs1	001	rd	0010011	SLLI
0000000		shamt	rs1	101	rd	0010011	SRLI
0100000		shamt	rs1	101	rd	0010011	SRAI
0000000		rs2	rs1	000	rd	0110011	ADD
0100000		rs2	rs1	000	rd	0110011	SUB
0000000		rs2	rs1	001	rd	0110011	SLL
0000000		rs2	rs1	010	rd	0110011	SLT
0000000		rs2	rs1	011	rd	0110011	SLTU
0000000		rs2	rs1	100	rd	0110011	XOR
0000000		rs2	rs1	101	rd	0110011	SRL
0100000		rs2	rs1	101	rd	0110011	SRA
0000000		rs2	rs1	110	rd	0110011	OR
0000000		rs2	rs1	111	rd	0110011	AND
fm	pred	succ	rs1	000	rd	0001111	FENCE
000000000000			00000	000	00000	1110011	ECALL
000000000001			00000	000	00000	1110011	EBREAK

Figure 1. 32 bit RISC-V ISA

Control Word

CONTROL WORD													
34:30	29:25	24:20	19	18	17:16	15	14:9	8	7:6	5	4	3	2:0
Datapath Control Signals[25:0]								Data Memory Control Signal[3:0]			PC Control Signal[4:0]		
25:21	20:16	15:11	10	9	8:7	6	5:0	3	2:1	0	4	3	2:0
AA[4:0]	BA[4:0]	DA[4:0]	MA	MB	MD[1:0]	RW	FS[5:0]	MW	BHW[1:0]	U	PL	JB	BC[2:0]

Table 1 Control Word

Control Word Explanation

- MA selects the A input of the Function Unit. If MA = 0, it means rs1 value, else MA=1 it means PC value is selected.
- MB selects the B input of the Function Unit. If MB = 0, it means rs2 value, else MB=1 it means immediate value, which comes from instruction, is selected.
- MD selects the data input of the Register File. MD = 0 means “data output of Data Memory” is selected. MD = 1 means output of Function Unit is selected. MD = 2 means “PC+4”, which means next instruction and comes from PC unit, is selected. MD = 3 means “immediate value”, which comes from instruction, is selected.
- RW is write enable signal for Register File.
- MW is write enable signal for Data Memory.
- BHW controls Data Memory, if BHW = 0, least significant byte of data is read or wrote. If BHW = 1, least significant two byte of data is read or wrote. If BHW = 2, all 32-bits of the data is read or wrote.
- U also controls Data Memory. If it is 1, then data loaded to the most significant bits of output.
- PL controls PC module. When PL = 1, this means there is jump or branch. If it is 0, the PC module continues its routine in which it increments the PC by 4.
- JB is a control signal for PC module. If it is 1, this means there is a jump and PC module loads the new value, which comes from the FU, to the PC.
- BC signal is used when there is a branch. It selects the condition of branch.
  - BC = 0 means BEQ instruction and branch depends on Z signal.
  - BC = 1 means BNE instruction and branch depends on $\bar{Z}$ signal.
  - BC = 4 means BLT instruction and branch depends on LT signal.
  - BC = 5 means BGE instruction and branch depends on $\overline{(LT)}$ signal.
  - BC = 6 means BLTU instruction and branch depends on ULT signal.
  - BC = 7 means BGEU instruction and branch depends on $\overline{(ULT)}$ signal.
- FS controls Function Unit, as detailed explanation in table below.

Select[2:0]	Bsel	Cin	Operation	Instruction
000	0	0	A+B	ADD/ADDI
000	1	1	A-B	SUB
010	1	1	A<B	SLT/SLTI
011	1	1	unsigned(A)<unsigned(B)	SLTU/SLTUI
100	X	X	A^B	XOR/XORI
110	X	X	A|B	OR/ORI
111	X	X	A&B	AND/ANDI

Table 2 Function Select Encoding

CONTROL WORD ENCODING

DA,BA,AA		MA		MB		MD		RW		MW		BHW		U		PL		JB		BC		FS	
Function	Code	Function	Code	Function	Code	Function	Code	Function	Code	Function	Code	Function	Code	Function	Code	Function	Code	Function	Code	Function	Code	Function	Code
R0	00000	Rs1	0	Rs2	0	Memory	00	Write to RF	0	Write to Data Memory	0	Byte	00	No Upper Load	0	No Branch or Jump	0	Branch	0	EQ	000	A+B	000000
R1	00001	PC	1	Immediate	1	FU	01	No write to RF	1	No write to Data Memory	1	Half Word	01	Upper Load	1	Branch or Jump	1	Jump	1	NE	001	A-B	000011
R2	00010					PC+4	10					Word	10							LT	100	A<B	001011
R3	00011					Immediate	11													GE	101	Unsigned A<B	001111
R4	00100																			LTU	110	A^B	0100XX
R5	00101																			GEU	111	A|B	0110XX
R6	00110																					A&B	0111XX
R7	00111																					A<<B	1X00XX
R8	01000																					A>>B	1X01XX
…	…																					Signed(A)>>B	1X11XX

Table 3 Control Word Encoding

Memory Structures

Program Memory and Data Memory are designed as separate modules. Both feature byte addressing and are designed as single port asynchronous memory. In the final synthesis, Program Memory was chosen as 1Kb (256 words) and Data Memory as 128 Bytes (contains 32 words).

Function Unit

For a faster adder block, the Carry Look Ahead structure is used. Unlike the Carry Ripple structure, this structure gives faster results because it does not expect the carry of the previous bit for the result of each bit. However, utilization also increases rapidly with the number of bits. So 8 “4-bit Carry Look Ahead” adders in series were used to obtain a 32-bit adder. As for shifter barrel shifter architecture combined with input and output mirroring to obtain both left and right shift in a single shifter block, saving more space.

Pipeline Stages and Hazard Solutions

RISC-V pipeline consists of 5 stages; instruction fetch (IF), instruction decode (ID), execute (EX), memory access (MEM), and register write-back (WB).

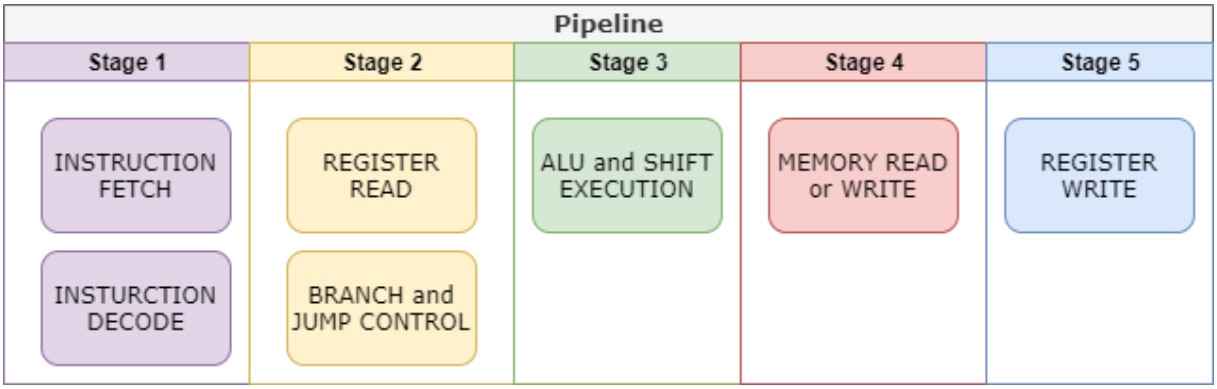

Figure 2 Pipeline Stages

Fetch (IF)

Instruction fetching from memory & decoding instructions.

Decode (ID)

Reading register and controlling jump and branch conditions.

Execute (EX)

Executing given operation or calculating new address.

Memory (MEM)

Accessing memory operand.

Write Back (WB)

Writing result back to register.

In our design, the control word is created in the IF stage. Branch and Jump decisions are made at the ID stage. Therefore, when these instructions come, 1 flush operation is performed. As a result, Branch and Jump instructions are executed with CPI = 2.

A "forward unit" was created to prevent errors that may occur when the register used in an instruction is to be used again without executing the write back stage.

These errors were avoided without affecting the CPI, unless the instruction reusing the same register was a branch instruction or came after a load instruction. Branch instructions are delayed with a bubble as they will use a value that is ready as a result of ALU or Shifter in the ID stage. Also, since the load instructions prepare the value to be loaded into the register after the MEM stage, the following instruction must be delayed with a bubble.

The possible CPI of the instructions are as follows;

Instruction	CPI
Arithmetic-Logic Instructions	1
Store	1
Load	1 or 2
Jump	2
Branch	2 or 3

Table 4 CPI of various instructions

Note: The destination register of the load instruction cannot be used as the source register of the next branch instruction, two NOPs must be inserted in between. It is assumed that the avoidance of this error will be considered by the assembler.

The control unit required for the bubble operation decides according to the register addresses of the IF stage and the ID stage. Flushes are determined by the module that also controls the Branches and Jumps in the ID stage.

Resource Utilization and Performance

- The processor synthesized with the "tcb018bcdgp2atc-110" library contains a total of 22745 cells, of which 14913 are in the control unit and 7832 are in the datapath. Overall area consumption is 1035639 nm^2.
- The entire design consumes 177.104 mW of energy.
- Biggest delay in the processor is 17.571 ns. If we give a %13.8 error margin, we get 20ns which is equal to a 50MHz processor. If we really trust our hardware, a 55.5 MHz processor can be achieved with an error margin of %2.5. Theoretically highest frequency possible is 56.91 MHz.