Jumpabug

MIPS64 Processor Design
External Architecture Specification

John Dawson
256 37th street
Pittsburgh PA, 15201

This page is intentionally blank.

Table of Contents

1. Introduction

This document details the design of the Jumpabug custom 64 bit MIPS soft core processor for use within Altera FPGA devices.

1.1 References

This section outlines the references used for this document.

MIPS64Instruction Set	*http://equipe.nce.ufrj.br/gabriel/arqcomp2/MIPS64_Instruction_Set_v0.95.pdf*
Computer Organization and Design	
See Mips Run Linux	

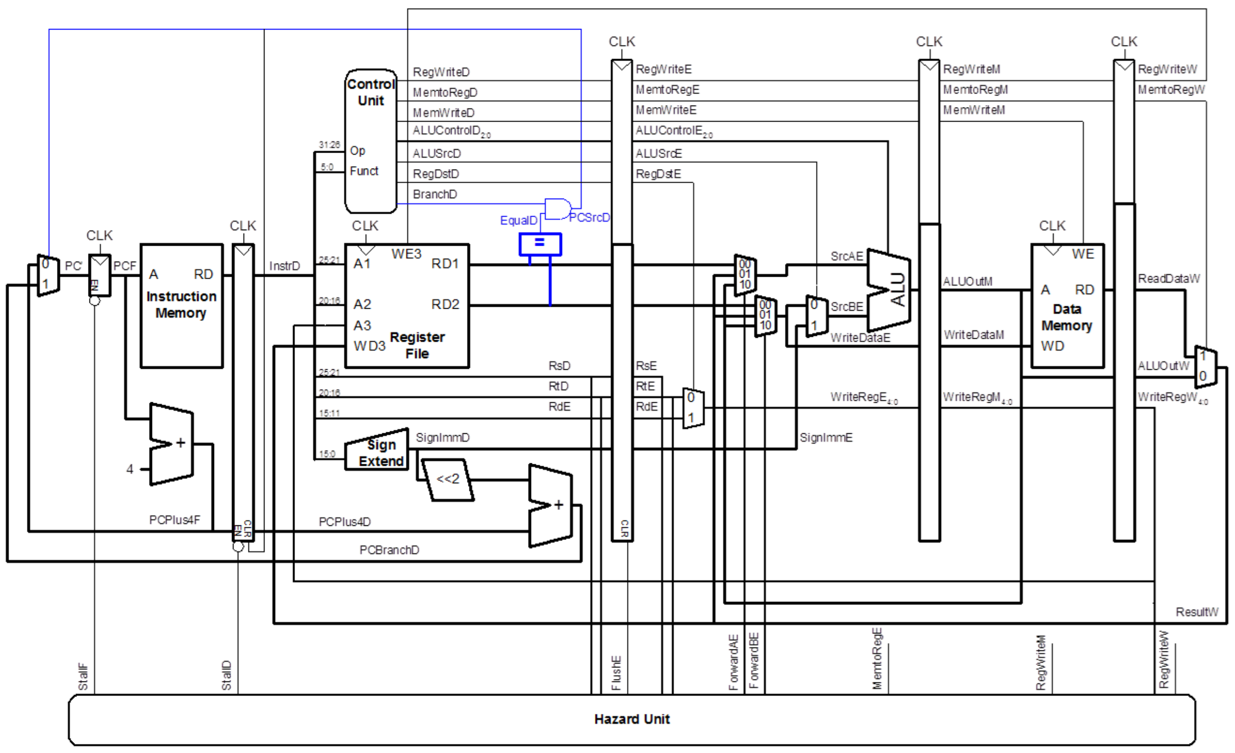
1.2 Glossary of Terms

1.3 Revision History

Version	Date	Changed By	Modifications
0.1	9Sept02	J. Dawson	Initial draft.

2. Overview

The Jumpabug MIPS64 IP consists of a single core 64 bit MIPS processor designed for use within Altera brand FPGA devices.



2.1 Main Features

- Five stage pipeline
- Full compliance to MIPS64 ISA V0.95
- Core operating frequency of TBD MHz
- Integrated Coprocessor 0 and Coprocessor 1 support

3. Pipeline Detailed Design

3.1 Instruction Fetch (IF)

3.1.1 Overview

The instruction fetch stage of the Jumpabug core is responsible for requesting the next instruction from memory and passing it to the decode stage. The block continually requests instructions from the instruction memory interface in increments of 4 bytes. On each clock cycle the address programmed into the program counter (PC) is updated from one of the following sources:

- Previous PC Value (Normal Operation)
- Branch/Jump Address
- Exception/Interrupt Address

In the case of a memory access wait condition, the IF block will initiate an IF/ID Pipeline stage stall which will allow the remaining stages to complete while waiting on the next instruction.

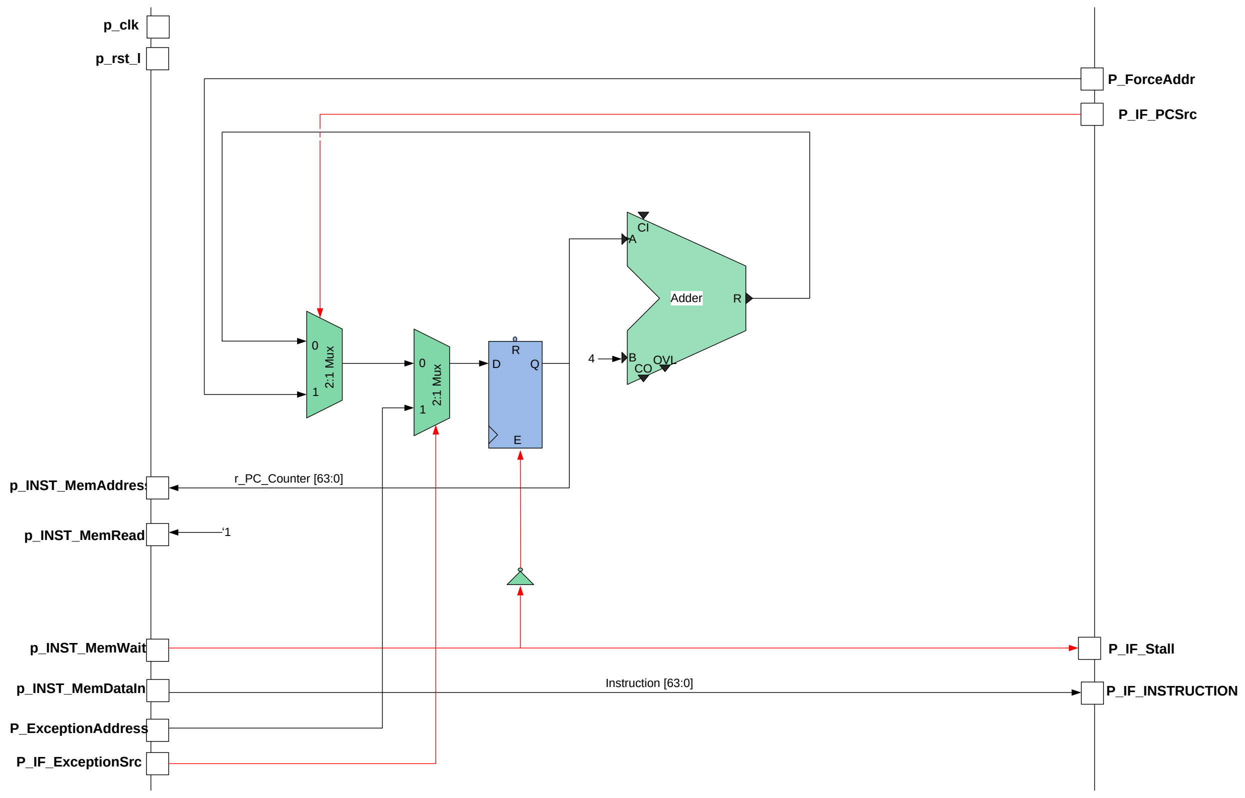


Figure 1: IF Block Diagram (Black - Data, Red - Control)

3.1.2 Port Descriptions

The IF block design is contained in a single Verilog file: **IF_Block.v**. The block contains the following I/O ports.

I/O Name	Direction	Size	Description
P_clk	IN	1	Core clock
P_rst_l	IN	1	Core reset
P_IF_INST_MemDataIn	IN	32	Instruction data from memory
P_IF_INST_MemWait	IN	1	Memory access wait from memory
P_IF_INST_MemAddress	OUT	64	Instruction address to memory
P_IF_INST_MemRead	OUT	1	Instruction memory read request
P_IF_BranchAddress	IN	64	Branch address (from MEM stage)
P_IF_ExceptionAddress	IN	64	Exception address (From COPROC 0)
P_IF_PCSrc	IN	1	Branch address select
P_IF_ExceptionSrc	IN	1	Exception address select
P_IF_Instruction	OUT	32	Instruction to ID stage
P_IF_PC_Counter	OUT	64	Current instruction PC value to ID stage
P_IF_Stall	OUT	1	IF stage stall signal

Table 1: IF Block I/O

3.2 Instruction Decode (ID)

3.2.1 Overview

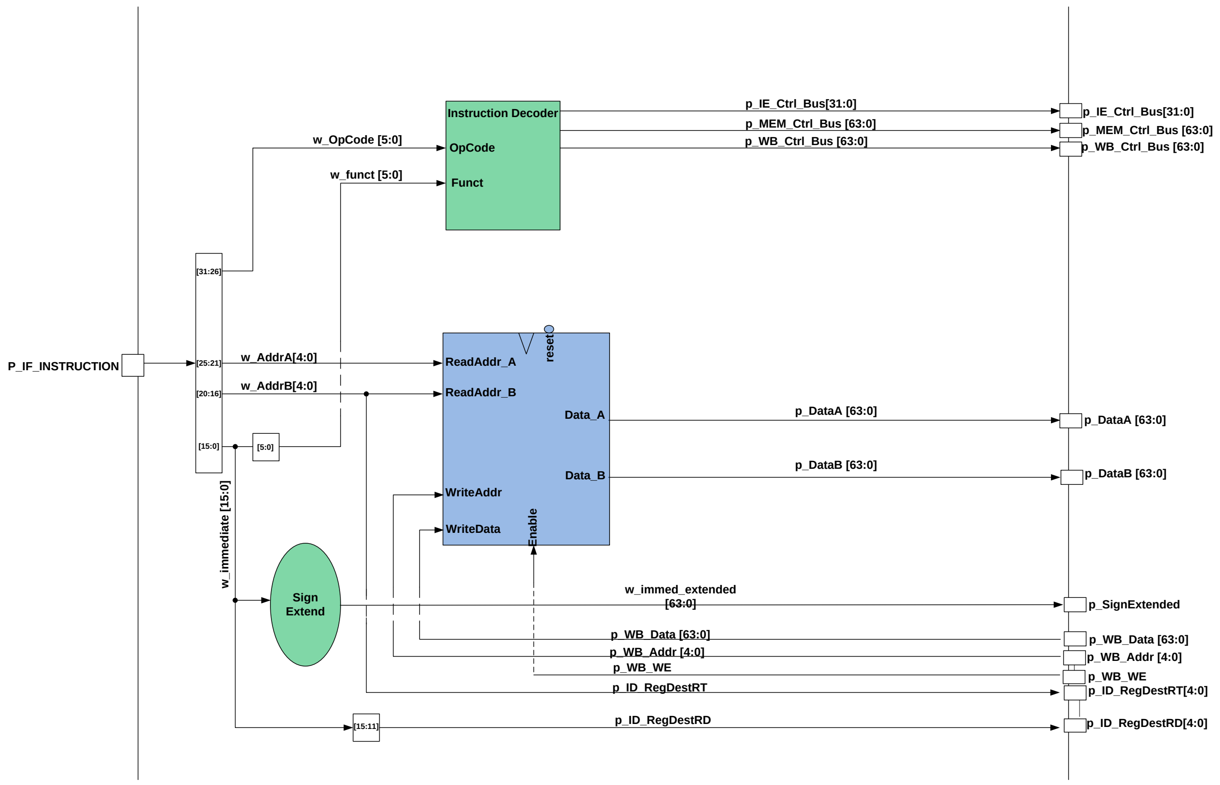


Table 2: Instruction Decode Block Diagram

The Instruction Decode block is responsible for:

- Decoding the current address
- Generating the correct IE, MEM, and WB stage control signals
- Read out the required register data from the 32x64 register block
- Sign extend any immediate values
- Pass instruction RT and RD fields to the IE stage (Possible writeback Destination)

To accomplish this, the 32 bit instruction from the IF stage is broken down to multiple bit fields as outlined below:

Instruction[31:26]	**Instruction[25:21]**	**Instruction[20:16]**	**Instruction[15:25]**	**Instruction[4:0]**
OpCode	Register Address A	Register Address B	Immediate Value	
				Funct

Table 3: Instruction Fields

The following table describes each of these fields in more detail:

Instruction Field	Description
OpCode	Instruction operation code, used by the control decoder to determine the instructions ID and assign the correct control signals.
Register Address A	Register block port A address
Register Address B	Register block port B address
Immediate Value	For Immediate instructions this 15 bit value is sign extended and used for the immediate operation.

Funct	Instruction function code, used by the control decoder to determine the instructions ID and assign the correct control signals.

Table 4: Instruction Field Descriptions

3.2.2 Port Descriptions

The ID block design is contained in a single Verilog file: **ID_Block.v**. The block contains the following I/O ports.

I/O Name	Direction	Size	Description
P_clk	**IN**	**1**	**Core clock**
P_rst_l	**IN**	**1**	**Core reset**
P_ID_IN_Instruction	**IN**	**32**	**Instruction from IF stage**
P_ID_DataA	**OUT**	**64**	**Data from port A of the register block**
P_ID_DataB	**OUT**	**64**	**Data from port B of the register block**
P_ID_SignExtended	**OUT**	**64**	**Sign extended immediate to IE stage**
P_ID_RT_RegDest	**OUT**	**5**	**Instruction RT field for WB destination**
P_ID_RD_RegDest	**OUT**	**5**	**Instruction RD field for WB destination**
P_ID_WB_Data	**IN**	**64**	**Writeback data**
P_ID_WB_Addr	**IN**	**5**	**Writeback address**
P_ID_WB_WE	**IN**	**1**	**Writeback address enable**
P_ID_IE_Ctrl_Bus	**OUT**	**32**	**IE stage control signals**
P_ID_MEM_Ctrl_Bus	**OUT**	**32**	**MEM stage control signals**
P_ID_WB_Ctrl_Bus	**OUT**	**32**	**WB stage control signals**

Table 5: ID Block I/O

The ID block consists of the following two sub components: Register Block and the control table. The next two sections will outline these in more detail.

3.2.3 Register Block

The ID block contains 32 general purpose 64 bit registers arranged in a single write port, dual read port register block:

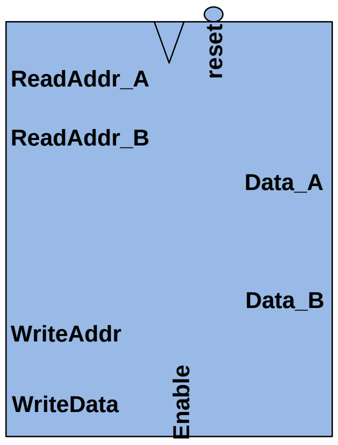


Figure 2: 32x64 Dual Port Register Block

All registers are read/writable with the exception of register 0 which always returns the value zero on a read.

3.2.4 Control Table

The control table block contains all the logic to decode the current instruction and produce three control bus groups: IE_control, MEM_Control, and WB_Control.

OpCode	Funct	Branch	Instruction Name	IE Control	Mem Control	WB Control
000000	100000	XXXXX	ADD			
001000	XXXXXX	XXXXX	ADDI			
001001	XXXXXX	XXXXX	ADDIU			
000000	100001	XXXXX	ADDU			
000000	100100	XXXXX	AND			
001100	XXXXXX	XXXXX	ANDI			
000100	XXXXXX	XXXXX	BEQ			
000001	XXXXXX	XXXXX	BGEZAL			
010100	XXXXXX	XXXXX	BEQL			
000001	XXXXXX	00001	BGEZ			
000001	XXXXXX	10001	BGEZAL			
000001	XXXXXX	10011	BGEZALL			
000001	XXXXXX	00011	BGEZL			
000111	XXXXXX	00000	BGTZ			
010111	XXXXXX	00000	BGTZL			
000110	XXXXXX	00000	BLEZ			
010110	XXXXXX	00000	BLEZL			
000001	XXXXXX	00000	BLTZ			
000001	XXXXXX	10000	BLTZAL			
000001	XXXXXX	10010	BLTZALL			
000001	XXXXXX	00010	BLTZL			
000101	XXXXXX	XXXXX	BNE			
010101	XXXXXX	XXXXX	BNEL			
000000	001101	XXXXX	BREAK			
011100	100001	XXXXX	CLO			
011100	100000	XXXXX	CLZ			
000000	101100	XXXXX	DADD			
011000	XXXXXX	XXXXX	DADDI			
011001	XXXXXX	XXXXX	DADDIU			
000000	101101	XXXXX	DADDU			
011100	100101	XXXXX	DCLO			
011100	100100	XXXXX	DCLZ			
000000	011110	XXXXX	DDIV			
000000	011111	XXXXX	DDIVU			
000000	011010	XXXXX	DIV			
000000	011011	XXXXX	DIVU			
000000	011100	XXXXX	DMULT			
000000	011101	XXXXX	DMULTU			
000000	111000	XXXXX	DSLL			
000000	111100	XXXXX	DSLL32			
000000	010100	XXXXX	DSSLV			
000000	111011	XXXXX	DSRA			
000000	111111	XXXXX	DSRA32			
000000	010111	XXXXX	DSRAV			
000000	111010	XXXXX	DSRL			
000000	111110	XXXXX	DSRL32			
000000	010110	XXXXX	DSRLV			

000000	101110	XXXXX	DSUB			
000000	101111	XXXXX	DSUBU			
000010	XXXXXX	XXXXX	J			
000011	XXXXXX	XXXXX	JAL			
000000	001001	XXXXX	JALR			
000000	001000	XXXXX	JR			
100000	XXXXXX	XXXXX	LB			
100100	XXXXXX	XXXXX	LBU			
110111	XXXXXX	XXXXX	LD			
011010	XXXXXX	XXXXX	LDL			
011011	XXXXXX	XXXXX	LDR			
100001	XXXXXX	XXXXX	LH			
100101	XXXXXX	XXXXX	LHU			
110000	XXXXXX	XXXXX	LL			
110100	XXXXXX	XXXXX	LLD			
001111	XXXXXX	XXXXX	LUI			
100011	XXXXXX	XXXXX	LW			
100010	XXXXXX	XXXXX	LWL			
100110	XXXXXX	XXXXX	LWR			
100111	XXXXXX	XXXXX	LWU			
011100	XXXXXX	XXXXX	MADD			
011100	XXXXXX	XXXXX	MADDU			
000000	010000	XXXXX	MFHI			
000000	010010	XXXXX	MFLO			
000000	001011	XXXXX	MOVN			
000000	001010	XXXXX	MOVZ			
011100	000100	XXXXX	MSUB			
011100	000101	XXXXX	MSUBU			
000000	010001	XXXXX	MTHI			
000000	010011	XXXXX	MTLO			
011100	000010	XXXXX	MUL			
000000	011000	XXXXX	MULT			
000000	011001	XXXXX	MULTU			
000000	000000	XXXXX	NOP			
000000	100111	XXXXX	NOR			
000000	100101	XXXXX	OR			
001101	XXXXXX	XXXXX	ORI			
101000	XXXXXX	XXXXX	SB			
111000	XXXXXX	XXXXX	SC			
111100	XXXXXX	XXXXX	SCD			
111111	XXXXXX	XXXXX	SD			
011100	111111	XXXXX	SDBBP			
101100	XXXXXX	XXXXX	SDL			
101101	XXXXXX	XXXXX	SDR			
101001	XXXXXX	XXXXX	SH			
000000	000000	XXXXX	SLL			
000000	000100	XXXXX	SLLV			
000000	101010	XXXXX	SLT			
001010	XXXXXX	XXXXX	SLTI			
001011	XXXXXX	XXXXX	SLTIU			
000000	101011	XXXXX	SLTU			
000000	000011	XXXXX	SRA			
000000	000111	XXXXX	SRAV			

000000	000010	XXXXX	SRL			
000000	000110	XXXXX	SRLV			
000000	100010	XXXXX	SUB			
000000	100011	XXXXX	SUBU			
101011	XXXXXX	XXXXX	SW			
101010	XXXXXX	XXXXX	SWL			
101110	XXXXXX	XXXXX	SWR			
000000	001100	XXXXX	SYSCALL			
000000	110100	XXXXX	TEQ			
000001	XXXXXX	01100	TEQI			
000000	110000	XXXXX	TGE			
000001	XXXXXX	01000	TGEI			
000001	XXXXXX	01001	TGEIU			
000000	110010	XXXXX	TLT			
000001	XXXXXX	01010	TLTI			
000001	XXXXXX	01011	TLTIU			
000000	110011	XXXXX	TLTU			
000000	110110	XXXXX	TNE			
000001	XXXXXX	01110	TNEI			
000000	100110	XXXXX	XOR			
001110	XXXXXX	XXXXX	XORI			

Figure 3: CPU Instruction Decode table

3.3 Instruction Execute (IE)

3.3.1 Overview

The Instruction Execute stage of pipeline is responsible for executing the following functions in the MIPS pipeline:

- All Arithmetic Operations
- Branch Condition Calculations
- Jump Condition Calculations
- Memory Address Calculations

3.3.2 IE Stage Control Signals

The 32 bit control bus generated by the control decoder in the ID stage is broken out into the following control signals in the IE block.

IE Stage Control Bus	IE Stage Signal	Description
IE_Control[3:0]	W_IE_ALUop	ALU OpCode. Selects which ALU operation to perform
IE_Control[4]	W_IE_RegDest	Selects between the RT and RD instruction fields for the writeback result address.
IE_Control[5]	W_IE_ALUSRC	Selects between register data B or the sign extended immediate as the second ALU input
IE_Control[6]	W_IE_ShiftInputSwitch	Selects between register data A or register data B as the first ALU input.
IE_Control[7]	W_IE_SHAMTSel	Selects between register data A or the sign extended immediate (bits [10:6]) as the ALU shift input
IE_Control[8]	W_IE_BranchSel	
IE_Control[9]	W_IE_ReverseB	

3.3.3 Arithmetic Operations

The IE stage of the MIPS pipeline is capable of performing the following arithmetic operations:

Operation	Latency	Notes

3.3.4 ALU

3.4 Memory Access (MEM)

3.4.1 Overview

3.5 Writeback (WB)

3.5.1 Overview

3.6 Data Forwarding Unit

3.7 Hazard Unit

4. Coprocessor 0 Detailed Design

4.1 Overview

4.2 Control Register Set

5. Coprocessor 1 Detailed Design

5.1 Overview

5.2 Control Register Set