System Specification

MUSA Core Processor

Fazemos Qualquer Negócio Inc.

Build 2.1

History Review

Date	Description	Author(s)
09/18/2014	Document conception.	Abdul Udongo
09/30/2014	Adaptation to ipPROCESS model.	Terseu Hunter

SUMÁRIO

1. Introduction

1.1. Purpose

The main purpose of this document is to define the specifications for the MUSA (**Microprocessor Unit for SoC Applications**) processing core. The following sections define the implementation parameters which composes the general MUSA core processing requirements and specification. This requirements include processor operation modes, Instruction Set Architecture (ISA) and internal registers specification.

1.2. Document Outline Description

This document is outlined as follow:

- Section ??: Presents the architecture design overview and general requirements.
- Section ??: Specifies the processor instruction set architecture.
- Section ??: Describes the general purpose registers.

1.3. Acronyms and Abbreviations

Acronym	Description
RISC	Reduced Instruction Set Computer
GPR	General Purpose Registers
FPGA	Field Gate Programmable Array
GPPU	General Purpose Processing Unit
SPRAM	Single-port RAM
HDL	Hardware Description Language
CPU	Central Processing Unit
ISA	Instruction Set Architecture
ALU	Arithmetic and Logic Unit
PC	Program Counter
RF	Flags Register
CST	Constant Value

2. Design Overview

MUSA is a RISC GPPU, featuring a minimal instruction set through three functional groups classification and three addressing modes. The microprocessor hardware structure is intended to cover low complexity applications. MUSA is a 32-bit word-oriented system, composed by 32 GPR in one register file. MUSA is also composed by a 32-bit program counter. MUSA current state supports basic arithmetical and logical operations, including multiplication and division. Those operations, and others, are fully detailed in the following sections.

2.1. Perspectives

The MUSA 32-bit core release targets modern FPGA devices with embedded SPRAM memory, and is intended to be deployed as a GPPU soft IP-core. Its architecture is designed under MIPS original implementation, presenting a slightly reduced ISA.

2.2. Main Characteristics

- Harvard architecture;
- 32 general purpose registers;
- RISC/MIPS-based Instruction Set Architecture;
- Load-Store/Register-Register processor architecture;
- Three instructions functional groups: (1) load and store; (2) computational; and (3) jump and branch.
- Three addressing modes: (1) immediate; (2) base-shift; and (3) by register;
- Architecture with five functional stages;
- Five clock Cycles per Instruction (CPI);
- Big-endian data organization;
- Support for overflow/underflow, above, equal and error flags definition;

2.3. Non-functional Requirements

- The FPGA prototype should run in a Terasic ALTERA Cyclone III (EP3C25F324) Development platform;
- The design must be described using Verilog-HDL;
- The tests must be writen in both Verilog-HDL or SystemVerilog;
- A set of test programs must be provided in order to validate the implementation;

3. Instruction Set Architecture

MUSA is built under RISC Load-Store/Register-Register processor architecture. CPU instructions are 32-bits long word and organized into the following functional groups:

- Load and store
- Computational
- Jump and branch

In MUSE load/store architecture, operations are performed through operands held into the register file (GPR Bank) and main memory is accessed only through load and store instructions. Signed and unsigned integers of 16-bits are supported by loads that either sign-extend or zero-extend the data loaded into the register.

3.1. CPU Load and Store Instructions

Mnemonic	Operands	Realization	Description
LW	$RD, RS1, I_{16}$	$RD\ MEM[RS1 + I_{16}]$	Load word from data memory.
SW	$RS1, RS2, I_{16}$	$MEM[RS1 + I_{16}]\ RS2$	Store word into data memory.

3.2. Computational Instructions

The MUSA provides 32-bit arithmetic operations. The computational instructions uses ALU two-operand instructions. These are signed versions of the following operations:

- Add
- Subtract
- Multiply
- Divide

Mnemonic	Operands	Realization	Description
ADD	RD, $RS1$, $RS2$	$RD\ RS1 + RS2$	Add two words.
SUB	RD, $RS1$, $RS2$	$RD\ RS1 - RS2$	Subtract two words.
MUL	RD, $RS1$, $RS2$	$RD\ RS1 * RS2$	Multiply two words.
DIV	RD, $RS1$, $RS2$	$RD\ RS1/RS2$	Divide two words.
AND	RD, $RS1$, $RS2$	$RD\ RS1 \odot RS2$	Logical AND.
OR	RD, $RS1$	$RD\ RS1 \oplus RS2$	Logical OR.
ADDI	RD, $RS1$, I_{16}	$RD\ RS1 + I_{16}$	Add a stored word and an immediate value.
SUBI	RD, $RS1$, I_{16}	$RD\ RS1 - I_{16}$	Subtract a stored word and an immediate value.
ANDI	RD, $RS1$, I_{16}	$RD\ RS1 \odot I_{16}$	Logical AND of a stored word and an immediate value
ORI	RD, $RS1$, I_{16}	$RD\ RS1 \oplus I_{16}$	Logical OR of a stored word and an immediate value.
CMP	RD, $RS1$	–	Compares RD and $RS1$ and set the flags.
NOT	RD	$RD \sim RD$	Logical NOT.

3.3. Jump/Branch Instructions

Mnemonic	Operands	Characteristic	Description
JR	R	Unconditional	Jump to destination.
JPC	I_{28}	Unconditional	Jump to destination PC-relative.
BRFL	RF, CST	Conditional	Jump to destination if $RF == CST$.
CALL	R	Unconditional	Subroutine call.
RET	–	Unconditional	Subroutine return

3.4. No Operation Instruction

The *No Operation* instruction (`NOP`) is used to control the instruction flow or to insert delays (stalls) into the datapath, such as when computing the result of a jump/branch

instruction. When using a NOP instruction after a branch/jump instruction it is so named a **branch delay slot**.

3.5. End of Opperations

The HALT instruction (system stop) must be implemented as a `L: j L` (a unconditional branch to the current address)

4. Internal General Purpose Registers

The current MUSA architecture provides 32 fixed-point general purpose registers of 32 bits: R_0 to R_{31}. The R_0 have special use for the hardware always returning zero, no matter what software attempts to store to it.