Datasheet

CCRV32ST-C Processor Core Integer Unit Microarchitecture

1.0

Scope

This document contains the CCRV32ST-C Processor Core Integer Unit Microarchitecture description. Document covers integer unit pipeline description as well as particular instruction groups timing and dependency.

Contents

1. Overview

The CCRV32ST-C processor adheres to the **"The RISC-V Instruction Set Manual Volume I: Unprivileged ISA, Document Version 20190608-Base-Ratified"** and **"The RISC-V Instruction Set Manual Volume II: Privileged Architecture Document Version 20190608-Priv-MSU-Ratified"**.

The processing unit comprises of single-issue, 6-stage fully interlocked pipeline executing **RV32I|E[MAFDCX]** instruction set including **Zicsr** and **Zifence** extensions with **U** and **S** privileged levels.

Figure 1.1 presents the processor's core integer unit datapath diagram.

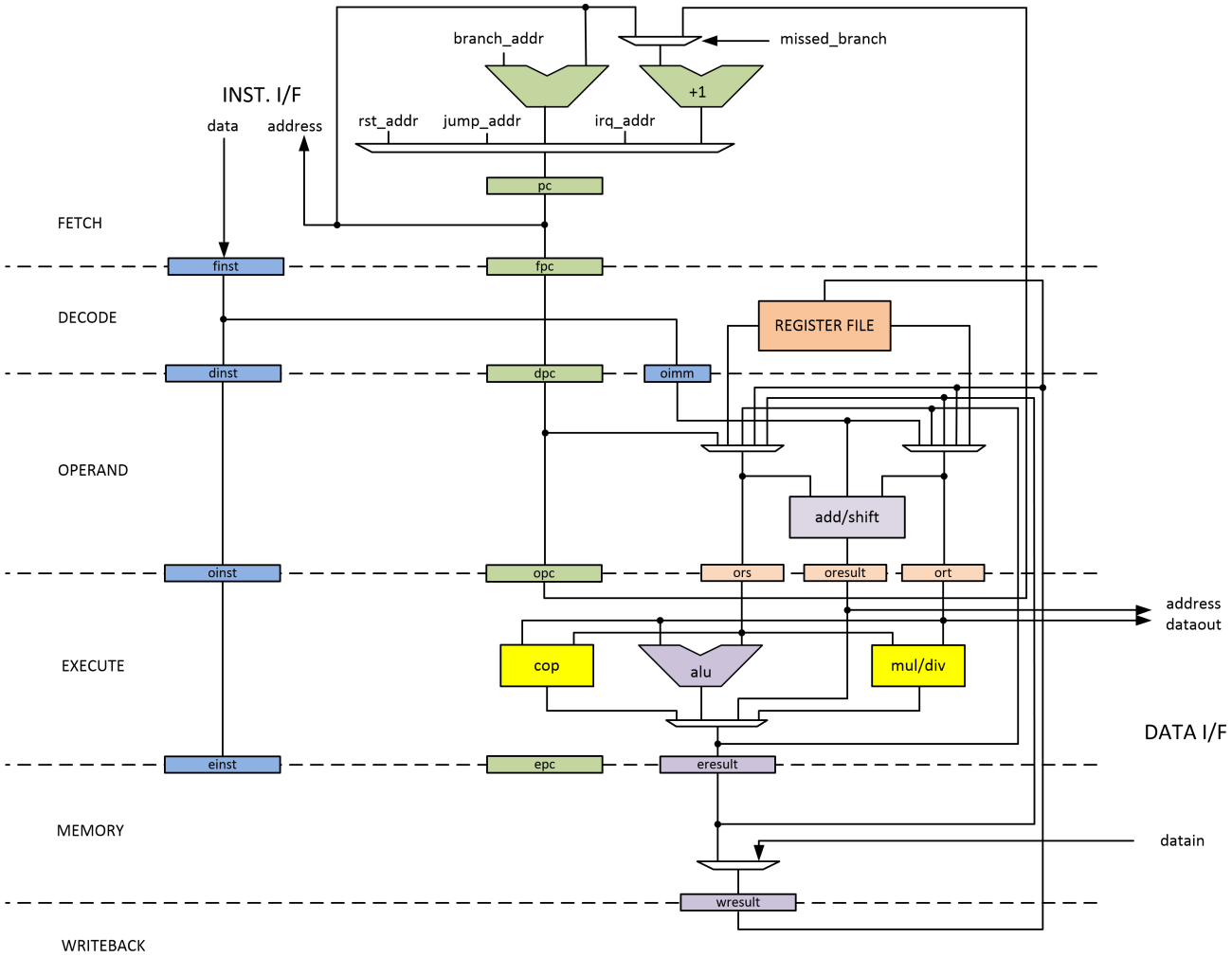

Figure 1.1. Integer unit datapath diagram.

2. Instruction Pipeline

The processor's core integer unit comprises of single-issue, in-order fully interlocked pipeline with 6 stages:

1. **FETCH**: Instruction is fetched from instruction cache or system bus.
2. **DECODE**: Instruction is decoded, branch and jump addresses are generated.
3. **OPERAND**: Operands are selected from register file or data bypasses. Early ALU executes shift/add operations. Load and store addresses are generated. Branch prediction is resolved.
4. **EXECUTE**: Instruction is executed and ALU result is selected. Exceptions are evaluated.
5. **MEMORY**: Load result is available, instruction result is seleted. System bus exceptions are detected.
6. **WRITEBACK**: Instruction result is written back to the register file.

3. Instruction Timing

In this chapter detailed instructions timing is described. Only integer unit pipeline stalls are taken into account. Particular instructions execution time can be prolonged by the instruction cache, data cache or system bus interconnect stalls.

3.1 Timing Summary

Instruction	Cycles (fast-mul)	Cycles (no fast-mul)
J, JAL	2	2
JR, JALR	4-6[1]	4-6[1]
Load	1-3[1]	1-3[1]
Store	1	1
MULT, MULTU (16-bit)	4	8 (ASIC), 6 (FPGA)
MULT, MULTU (32-bit)	8 (ASIC), 6 (FPGA)	8 (ASIC), 6 (FPGA)
DIV, DIVU	37	37
Predicted branch (32-bit mode)	2	2
Miss-predicted branch (32-bit mode)	4	4
Trapped instruction	5	5
All other instructions	1	1

[1] Depends on pipeline data hazard. See next sections.

3.2 Jump Instruction

Jump instruction takes 2 clock cycles to start executing target instruction. Figure 3.1 shows pipeline timing while executing Jump instruction.

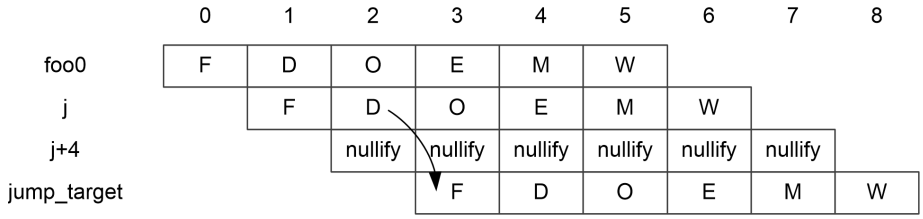

Figure 3.1. Jump Register execution with no data hazard.

Jump Register instruction takes 4 clock cycles to start executing target instruction. Figure 3.2 presents execution of the Jump Register instruction with no pipeline data hazard present.

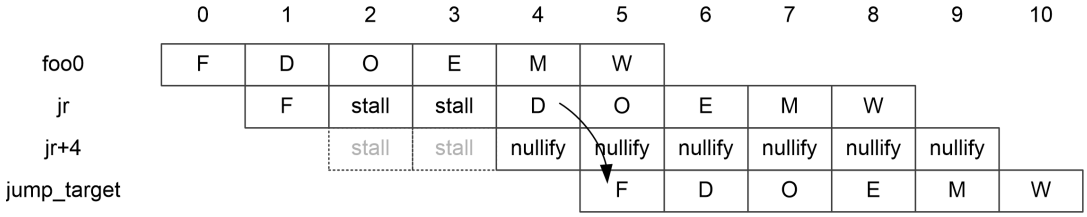

Figure 3.2. Jump Register execution with no data hazard.

In case of pipeline data hazard, the execution of Jump Register instruction can be prolonged by another up to two clock cycles. See Figure 3.3 and Figure 3.4.

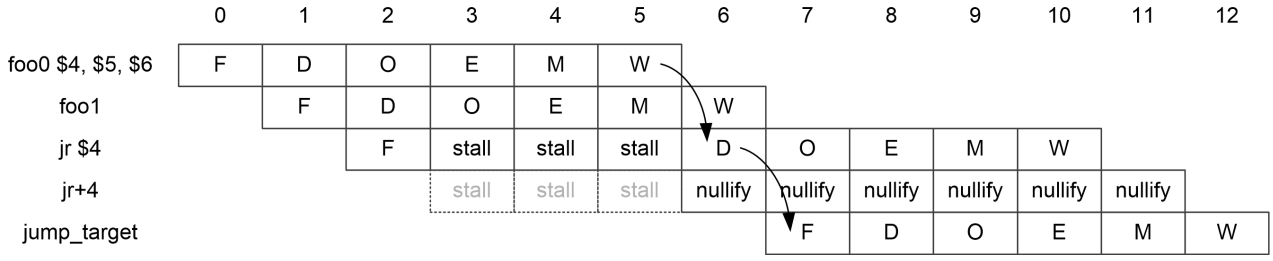

Figure 3.3. Jump Register execution with additional one clock cycle.

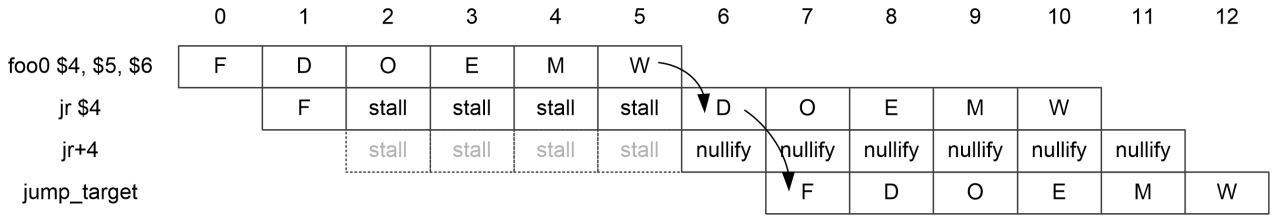

Figure 3.4. Jump Register execution with additional two clock cycles.

3.3 Load Instruction

Load Instructions in general is executed in one clock cycle unless data hazard exists. Figure 3.5 shows pipeline timing when load result is needed two instructions after load. Figure 3.6 shows pipeline timing when load result is needed immediately after load.

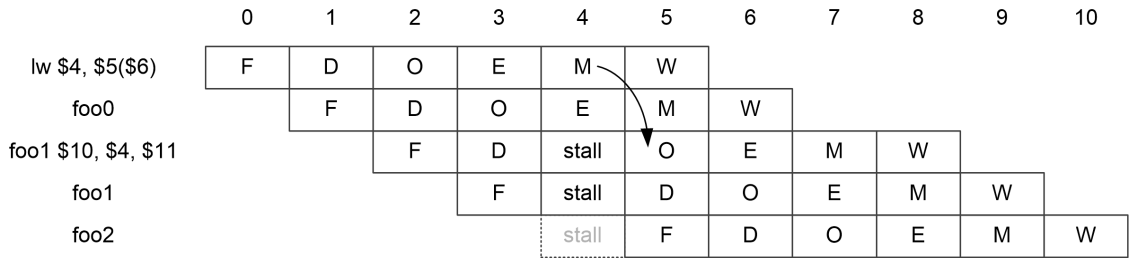

Figure 3.5. Load result is needed two instructions after load.

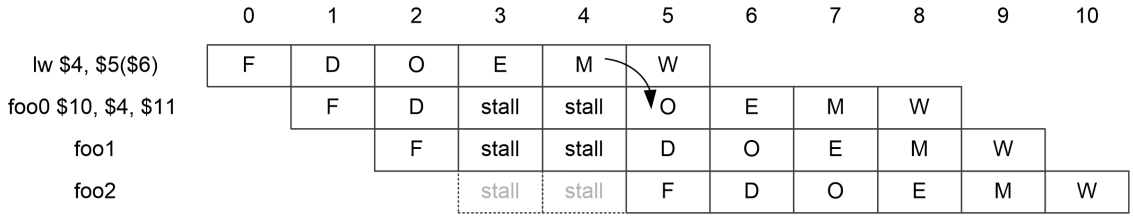

Figure 3.6. Load result is needed immediataly after load.

Figure 3.7 shows pipeline timing when load result is needed two instructions after load, while the following Jump Register instruction is prolonged with additional one clock cycle.

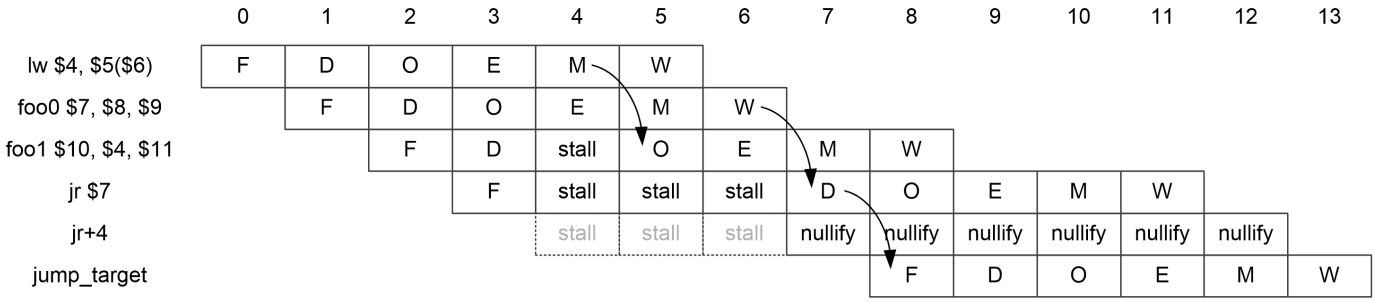

Figure 3.7. Integer unit datapath diagram.

3.4 Multiply Instruction

Figure 3.8 presents the pipeline timing of MUL instruction.

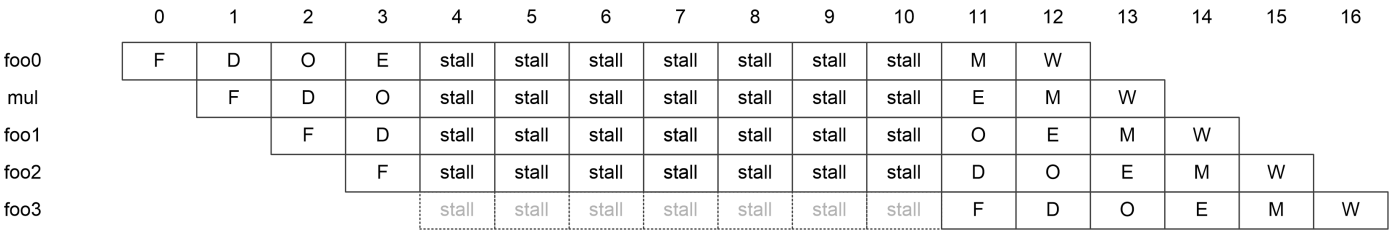

Figure 3.8. MUL instruction timing.

3.5 Branch Instruction

Correctly predicted branch instruction has the same timing as jump instruction and takes two clock ccyles. Figure 3.9 shows the two clock cycle penalty of miss-predicted branch.

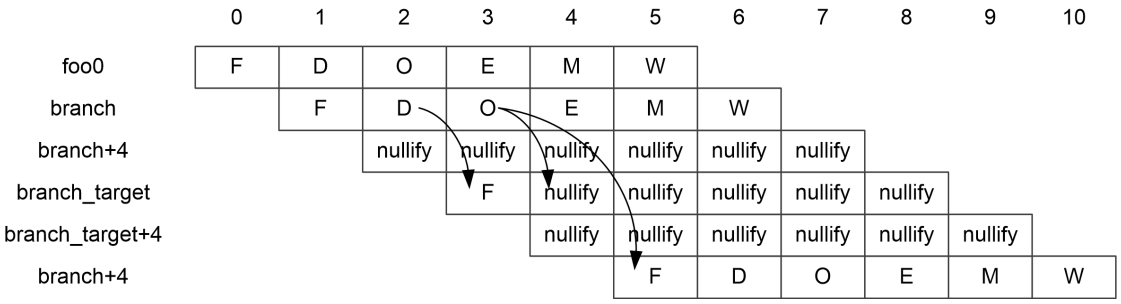

Figure 3.9. Miss-predicted branch instruction pipeline timing.

3.6 Trapped Instruction

Figure 3.10 shows pipeline timing of trapped instruction. Synchronous traps can be triggered by executing ECALL or EBREAK instructions or by encountering exception condition while executing regular instruction. Asynchronous traps are caused by interrupts.

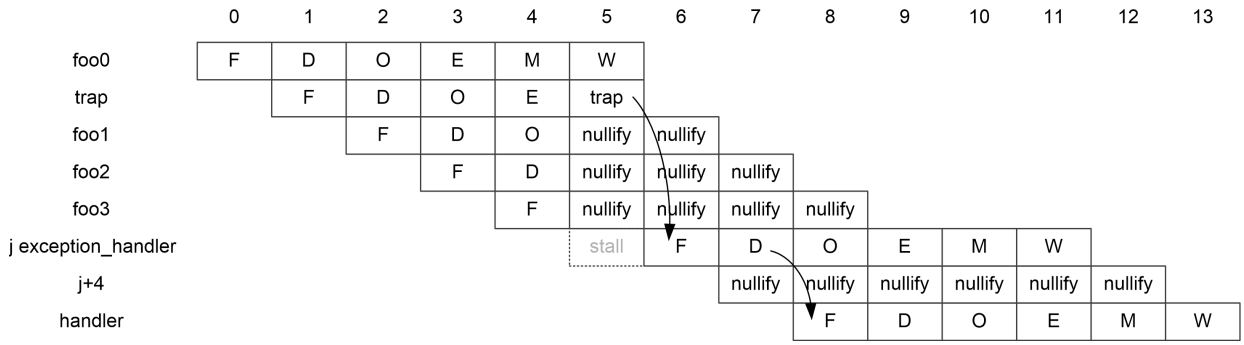

Figure 3.10. Trapped instruction pipeline timing.

4. Revision History

Doc. Rev.	Date	Comments
1.0	12-2019	First Issue.

ChipCraft Sp. z o.o.

Dobrzańskiego 3 lok. BS073, 20-262 Lublin, POLAND

www.chipcraft-ic.com

CCRV32ST-IU-Doc_122019.