UNIVERSITY OF CALIFORNIA

Los Angeles

AMPIRE:

Asynchronous Microprocessor with Instruction Retry

A thesis submitted in partial satisfaction of the

requirements for the degree Master of Science

in Computer Science

by

Chia-Chi Chao

1995

© Copyright by

Chia-Chi Chao

1995

The thesis of Chia-Chi Chao is approved.

Milos D. Ercegovac

David A. Rennels

Yuval Tamir, Committee Chair

University of California, Los Angeles

1995

Table of Contents

Chapter One – Introduction 1
1.1. Asynchronous Design 1
1.2. Fault Tolerance 3
1.3. Scope of Thesis 4

Chapter Two – Previous Work 6
2.1. Handshake Protocols 6
2.2. Asynchronous Processors and Circuits 8
2.3. Micro Rollback 11

Chapter Three – Processor Architecture 14
3.1. Instruction Set 14
3.2. Processor Overview 16
3.3. Normal Operation 16
3.3.1. Instruction Fetch 16
3.3.2. Instruction Issue 20
3.3.3. Instruction Execution 21
3.3.3.1. ALU 21
3.3.3.2. Data Memory Load/Store 22
3.3.3.3. Flow Control 23
3.3.4. Register File Access 25
3.4. Fault Tolerance 26
3.4.1. Sequence Number and Check Vector 26
3.4.2. Error Detection 29
3.4.3. Instruction Log and Validation 30
3.4.4. Rollback 31
3.4.5. Delayed Write Buffer 32
3.5. Synchronization 34
3.5.1. Module Level 34
3.5.2. Processor Level 34

Chapter Four – Behavioral Modeling 36

4.1. Building Blocks 37
4.1.1. Tri-State Bus 37
4.1.2. Muller C-Element 37
4.1.3. Buffer 38
4.1.4. Reset 39
4.1.5. Arbiter 40
4.1.6. Rollback 41
4.2. Processor Modules 44
4.2.1. Queues 44
4.2.2. Checkers 45
4.2.3. Memories 45
4.2.4. Program Counter 47
4.2.5. Arithmetic Logic Unit 47
4.2.6. Controller 48
4.2.6.1. Instruction Issuing Unit 48
4.2.6.2. Reservation Table 52
4.2.7. Delayed Write Buffers 53
4.2.7.1. REGDWB 53
4.2.7.2. MEMDWB 55
4.2.8. Instruction Log 56
4.3. Putting the Processor Together 58
4.3.1. Parameters and Timing 58
4.3.2. Top-Level Wiring and Testing Module 59

Chapter Five – Behavioral Simulation 60
5.1. Instruction Fetch/Issue and Queue Operations 60
5.2. Delayed Write Buffers and Checker Arbitration 64
5.3. Arbitration for R_bus and Out-of-Order Completion 67
5.4. Fault Detection and Rollback 70
5.5. Running a Real Program 76

Chapter Six – Hardware Design and Gate-Level Simulation 78
6.1. C-Elements 78
6.2. Arbiter 80
6.3. Register Completion Detector 81

6.4. Instruction Queue 83
6.5. Data Queue 85
6.5.1. Sequence Number Comparator 85
6.5.2. DQ Control 87
6.5.3. DQ Simulation 89
6.6. Fault Simulation with Gate-Level Modules 93

Chapter Seven – Conclusion 95

Appendix A – Verilog Simulation Code 98

Appendix B – AMPIRE Assembler 196
B.1. Assembly Code Format 196
B.2. Assembler Source Code 196

Bibliography 206

List of Figures

2.1. 2-Phase Handshake Protocol 6
2.2. 4-Phase Handshake Protocol 6
2.3. Delay-Insensitive 4-Phase Handshake Protocol 8
2.4. Request Generation for Double-Rail Data 8
2.5. MIPS as a Function of Supply Voltage for Caltech Processor 9
2.6. Handshake Model for Berkeley DSP 10
2.7. Micro Rollback, Restoring a Saved Snapshot 11
2.8. Register File with Support for Micro Rollback 12
3.1. AMPIRE Instruction Format 14
3.2. AMPIRE Block Diagram 18
3.3. Sequence Number Windows 27
3.4. Check Vector for ALU Operation 28
3.5. Validation and Rollback Signal Flow Diagram 31
3.6. Asynchronous Delayed Write Buffer for Register File 33
3.7. Halt and Rollback Sequence 35
4.1. Data Transfer Sequence 43
5.1. Test 1 with Slow IMEM 61
5.2. Test 1 with Fast IMEM 61
5.3. Test 1 with Fast IMEM, Close-Up 61
5.4. Test 1, Detailed Activities 63
5.5. Test 2 with All Delays=1 64
5.6. Test 2 with Long DMEM Write Cycle 65
5.7. Test 2 with Slow Memory Checker 66
5.8. Test 2, Showing Arbitration of Checkers 67
5.9. Test 3 with Slow DMEM 68
5.10. Test 3 with Fast DMEM 69
5.11. Test 4, All Faults 70
5.12. Test 4, CKI Fault 71
5.13. Test 4, CKF Fault 72
5.14. Test 4, ALU CKR Fault 73
5.15. Test 4, CKM Fault 74
5.16. Test 4, DMEM CKR Fault 75

5.17. Test 4, Rollback Sequence 75
5.18. Test 5, Find the Largest Number 77
6.1. 3-Input C-Element 78
6.2. 14-Input C-Element 79
6.3. 2-Input C-Element with Clear 79
6.4. Arbiter for R_bus 80
6.5. Gate-Level Arbiter 81
6.6. D-Latch 81
6.7. D-Latch with Set/Reset 82
6.8. Register Completion Detector 82
6.9. Control for IQ Buffer 84
6.10. Gate-Level IQ 84
6.11. Sequence Number Comparator 86
6.12. Borrow Circuits, Full and Half 86
6.13. XOR Gate 87
6.14. Difference (SUB) Circuit 87
6.15. Control for DQ Buffer 88
6.16. Test 6, Gate-Level DQ with DLY_CKI_CHK=48 90
6.17. Test 6, Gate-Level DQ with DLY_CKI_CHK=52 91
6.18. Test 7, Gate-Level DQ with DLY_DMEM_RD=8 92
6.19. Test 7, Gate-Level DQ with DLY_DMEM_RD=1 93
6.20. Fault Simulation with Gate-Level Modules 94

List of Tables

2.1. Average Instruction Periods for Berkeley DSP 9
3.1. AMPIRE Instruction Set 15
3.2. Processor Modules 17
4.1. Frequently Used Verilog Keywords 36
4.2. DWB Bits in the Instruction Log 57
A.1. Verilog Modules 98

ABSTRACT OF THE THESIS

AMPIRE:

Asynchronous Microprocessor with Instruction Retry

by

Chia-Chi Chao

Master of Science in Computer Science

University of California, Los Angeles, 1995

Professor Yuval Tamir, Chair

As we build faster digital circuits, the clock skew problem becomes a major limiting factor in large scale synchronous systems. Asynchronous design has an advantage of not requiring a global clock, and the high modularity that can be achieved is especially beneficial for large systems. Reliability of a system is an important issue in many applications, and hardware fault-tolerant techniques can be applied for fast recovery from transient faults.

This thesis reports the design of an asynchronous microprocessor that supports concurrent error detection and software-transparent fault recovery. On-chip parity checkers operate in parallel with other functional units, and when an error is detected, the state of the processor is restored to the instruction in which the error occurred. A Verilog behavioral model of the architecture has been written and simulated, and some gate-level self-timed circuits have also been designed.