

# Advanced Computer Organisation and Architecture

Smruti R. Sarangi

May 22, 2020

# Contents

<b>1</b>	<b>Introduction</b>	<b>11</b>
1.1	Moving from In-order to Out-of-order Pipelines . . . . .	13
1.2	Moving to Multi-core Processors . . . . .	15
1.2.1	GPUs . . . . .	16
1.2.2	Large Multicore Processors . . . . .	17
1.3	High Performance Memory System . . . . .	17
1.4	Power, Temperature, Variation, and Reliability . . . . .	18
1.5	Security . . . . .	19
<b>I</b>	<b>Processor Design</b>	<b>21</b>
<b>2</b>	<b>Out-of-order Pipelines</b>	<b>23</b>
2.1	Overview of In-Order Pipelines . . . . .	24
2.1.1	Processor Design . . . . .	24
2.1.2	Notion of Pipelining . . . . .	28
2.1.3	Interlocks . . . . .	31
2.1.4	Forwarding . . . . .	37
2.2	Performance Considerations . . . . .	42
2.2.1	The Performance Equation . . . . .	42
2.2.2	Multi-issue In-order Pipelines . . . . .	44
2.3	Overview of Out-of-order Pipelines . . . . .	47
2.3.1	Motivation . . . . .	47
2.3.2	Program Order vs Data Dependence Order . . . . .	49
2.3.3	Basics of an Out-of-order Machine . . . . .	51
2.4	Summary and Further Reading . . . . .	59
2.4.1	Summary . . . . .	59
2.4.2	Further Reading . . . . .	60
<b>3</b>	<b>The Fetch and Decode Stages</b>	<b>63</b>
3.1	Instruction Delivery from the I-Cache . . . . .	64
3.2	Problem 1: Is an instruction with a given PC a branch? . . . . .	67
3.2.1	Recording the Type of the Branch . . . . .	70
3.3	Problem 2: Is a branch taken or not taken? . . . . .	71
3.3.1	Bimodal Predictor . . . . .	71
3.3.2	Predictor with Saturating Counters . . . . .	74
3.3.3	Loop Predictor . . . . .	76
3.3.4	Predictors with Global History . . . . .	76
3.3.5	A 2-Level Predictor . . . . .	78
3.3.6	GAg, GAp, PAg, and PAp Predictors . . . . .	79

3.3.7	GShare Predictor . . . . .	83
3.3.8	Tournament Predictor . . . . .	84
3.4	Problem 3: What is the Target of a Branch? . . . . .	86
3.4.1	Branch Target Buffer (BTB) . . . . .	86
3.4.2	Call and Return Instructions . . . . .	86
3.5	The Decode Stage . . . . .	88
3.5.1	Pre-decoding CISC Instructions . . . . .	89
3.5.2	Decode Time Optimisation: Optimising Operations on the Stack Pointer . . . . .	90
3.5.3	Instruction Compression . . . . .	93
3.6	Summary and Further Reading . . . . .	96
3.6.1	Summary . . . . .	96
3.6.2	Further Reading . . . . .	97
<b>4</b>	<b>The Issue, Execute, and Commit Stages</b>	<b>99</b>
4.1	Instruction Renaming . . . . .	100
4.1.1	Overview of Renaming with Virtual Registers . . . . .	101
4.1.2	Renaming using Physical Registers . . . . .	102
4.1.3	The Rename Table . . . . .	103
4.1.4	Dependence Check Logic . . . . .	107
4.1.5	The Free List . . . . .	112
4.2	Instruction Dispatch, Wakeup, and Select . . . . .	113
4.2.1	Instruction Window . . . . .	114
4.2.2	Broadcast and Wakeup . . . . .	118
4.2.3	Instruction Select . . . . .	120
4.2.4	Early Broadcast . . . . .	125
4.2.5	Tricky Issues with Early Broadcast . . . . .	129
4.3	The Load-Store Queue (LSQ) . . . . .	132
4.3.1	Memory Dependencies . . . . .	132
4.3.2	Conceptually Handling Loads and Stores . . . . .	134
4.3.3	Design of the LSQ . . . . .	137
4.4	Instruction Commit . . . . .	141
4.4.1	Notion of Precise Exceptions and In-order Commit . . . . .	142
4.4.2	The Reorder Buffer (ROB) . . . . .	143
4.4.3	Releasing Resources and Bookkeeping . . . . .	145
4.4.4	Checkpointing and Restoring State . . . . .	148
4.5	Summary and Further Reading . . . . .	155
4.5.1	Summary . . . . .	155
4.5.2	Further Reading . . . . .	157
<b>5</b>	<b>Alternative Approaches to Issue and Commit</b>	<b>159</b>
5.1	Introduction . . . . .	159
5.1.1	Organisation of this Chapter . . . . .	160
5.2	Load Speculation . . . . .	160
5.2.1	Introduction . . . . .	160
5.2.2	Address Speculation . . . . .	162
5.2.3	Load-Store Dependence Speculation . . . . .	164
5.2.4	Latency Speculation . . . . .	169
5.2.5	Value Prediction . . . . .	170
5.3	Replay Mechanisms . . . . .	172

5.3.1	Pipeline Flushing . . . . .	173
5.3.2	Non-Selective Replay . . . . .	174
5.3.3	Methods to Replay Instructions . . . . .	175
5.3.4	Delayed Selective Replay . . . . .	178
5.3.5	Token Based Replay . . . . .	181
5.4	Simpler OOO Processor without a Register File . . . . .	184
5.4.1	Overview of the Design . . . . .	185
5.4.2	Detailed Design . . . . .	186
5.4.3	Comparison . . . . .	190
5.5	Compiler Based Techniques . . . . .	191
5.5.1	Data Flow Optimisations . . . . .	191
5.5.2	Loop Optimisations . . . . .	195
5.5.3	Software Pipelining . . . . .	200
5.6	EPIC Processors . . . . .	208
5.6.1	Pros and Cons of EPIC/VLIW Processors . . . . .	208
5.6.2	Difference between VLIW and EPIC Processors . . . . .	210
5.7	Design of the Intel Itanium Processor . . . . .	211
5.7.1	Overview of the Constraints . . . . .	212
5.7.2	Fetch Stage . . . . .	212
5.7.3	Instruction Dispersal Stage . . . . .	214
5.7.4	Register Remapping Stage . . . . .	215
5.7.5	High Performance Execution Engine . . . . .	219
5.7.6	Support for Aggressive Speculation . . . . .	223
5.8	Summary and Further Reading . . . . .	224
5.8.1	Summary . . . . .	224
5.8.2	Further Reading . . . . .	226
<b>6</b>	<b>Graphics Processors</b>	<b>229</b>
6.1	Traditional Technologies . . . . .	231
6.1.1	ASICs and ASIPs . . . . .	231
6.1.2	FPGAs . . . . .	232
6.2	Traditional GPUs . . . . .	233
6.2.1	Early Days of GPU . . . . .	235
6.2.2	High Level View of a Graphics Pipeline . . . . .	236
6.2.3	Vertex Processor . . . . .	238
6.2.4	Polymorph Engine . . . . .	239
6.2.5	Rasterisation . . . . .	240
6.2.6	Fragment Processor . . . . .	241
6.2.7	Pixel Engine . . . . .	242
6.2.8	Other Uses of a GPU . . . . .	243
6.3	Programming GPGPUs . . . . .	244
6.3.1	GPU ISAs . . . . .	244
6.3.2	Kernels, Threads, Blocks, and Grids . . . . .	245
6.3.3	Memory Access . . . . .	248
6.3.4	Streams, Graphs, and Events . . . . .	250
6.4	General Purpose Graphics Processors . . . . .	253
6.4.1	Overview of the Architecture of a GPU . . . . .	253
6.4.2	Structure of a GPC . . . . .	254
6.4.3	Structure of an SM . . . . .	254
6.4.4	Concept of a Warp . . . . .	257

6.4.5	The GPU Pipeline . . . . .	263
6.4.6	The Register File . . . . .	264
6.4.7	L1 Caches . . . . .	266
6.5	Summary and Further Reading . . . . .	266
6.5.1	Summary . . . . .	266
6.5.2	Further Reading . . . . .	269

## II The Memory System 271

<b>7</b>	<b>Caches</b>	<b>273</b>
7.1	Memory Hierarchy and the Notion of Caches . . . . .	274
7.1.1	Temporal and Spatial Locality . . . . .	275
7.1.2	Notion of a Cache . . . . .	277
7.1.3	Hierarchy of Caches . . . . .	279
7.1.4	Organisation of a Cache . . . . .	280
7.1.5	Basic Operations in a Cache . . . . .	287
7.1.6	Mathematical Analysis . . . . .	290
7.1.7	Optimising the Cache Design . . . . .	291
7.2	Virtual Memory . . . . .	293
7.2.1	Overlap and Size Problems . . . . .	294
7.2.2	Implementation of Virtual Memory . . . . .	296
7.3	Modelling and Designing a Cache . . . . .	301
7.3.1	Memory Technologies used in a Cache: SRAM and CAM Arrays . . . . .	301
7.3.2	Designing a Cache . . . . .	312
7.3.3	Circuit Level Modelling of a Cache: Elmore Delay Model	317
7.4	Advanced Cache Design . . . . .	327
7.4.1	Pipelined Caches . . . . .	327
7.4.2	Non-blocking Caches . . . . .	328
7.4.3	Skewed Associative Caches . . . . .	330
7.4.4	Way Prediction . . . . .	332
7.4.5	Loop Tiling . . . . .	333
7.4.6	Virtually Indexed Physically Tagged (VIPT) Caches . . .	336
7.5	Trace Caches . . . . .	339
7.5.1	Design of the Trace Cache . . . . .	340
7.5.2	Operation . . . . .	343
7.6	Instruction Prefetching . . . . .	345
7.6.1	Next Line Prefetching . . . . .	346
7.6.2	Markov Prefetching . . . . .	346
7.6.3	Call Graph Prefetching . . . . .	348
7.6.4	Other Approaches . . . . .	350
7.7	Data Prefetching . . . . .	351
7.7.1	Stride based Prefetching . . . . .	351
7.7.2	Pointer Chasing . . . . .	354
7.7.3	Runahead Execution and Helper Threads . . . . .	355
7.8	Summary and Further Reading . . . . .	358
7.8.1	Summary . . . . .	358
7.8.2	Further Reading . . . . .	359

<b>8</b>	<b>The On-chip Network</b>	<b>361</b>
8.1	Overview of an NoC	363
8.1.1	Nodes and Links	363
8.1.2	Network Topology	365
8.2	Message Transmission	373
8.2.1	Basic Concepts	373
8.2.2	Flow Control across a Single Link	375
8.2.3	Message Based Flow Control	380
8.2.4	Packet Based Flow Control: Store and Forward (SAF)	383
8.2.5	Packet Based Flow Control: Virtual Cut Through (VCT)	384
8.2.6	Flit based Flow Control: Wormhole Switching	386
8.2.7	Flit based Flow Control: Virtual Channel Based	388
8.3	Routing	390
8.3.1	Handling Starvation and Livelocks	394
8.3.2	Deadlocks in Routing Algorithms	395
8.3.3	Dimension Ordered Routing	399
8.3.4	Oblivious Routing	400
8.3.5	Adaptive Routing	403
8.3.6	Preventing Deadlocks by using Virtual Channels	406
8.4	Design of a Router	409
8.4.1	Input Buffering	410
8.4.2	Route Computation	411
8.4.3	Virtual Channel Allocation	414
8.4.4	Switch Allocation	414
8.4.5	Switch Traversal	416
8.4.6	Allocators and Arbiters	419
8.4.7	The Router's Pipeline	429
8.5	Non-Uniform Cache Architectures	434
8.5.1	Static NUCA(S-NUCA)	435
8.5.2	Dynamic NUCA(D-NUCA)	436
8.5.3	Advanced Schemes	439
8.6	Performance Aspects	441
8.6.1	Evaluation Metrics	441
8.6.2	Simulation Methodologies	442
8.7	Summary and Further Reading	445
8.7.1	Summary	445
8.7.2	Further Reading	448
<b>9</b>	<b>Multicore Systems: Coherence, Consistency, and Transactional Memory</b>	<b>449</b>
9.1	Parallel Programming	450
9.1.1	Shared Memory	451
9.1.2	Message Passing	454
9.1.3	Amdahl's Law	455
9.1.4	Gustafson-Barsis's Law	457
9.1.5	Design Space of Multiprocessors	458
9.2	Overview of Issues in Parallel Hardware	459
9.2.1	Shared and Distributed Caches	459
9.2.2	Memory Consistency	461
9.3	Cache Coherence	463

9.3.1	Theoretical Fundamentals . . . . .	463
9.3.2	Write-Update Protocol using a Bus . . . . .	478
9.3.3	Write-Invalidate Protocol using a Bus . . . . .	484
9.3.4	MESI Protocol . . . . .	487
9.3.5	MOESI Protocol . . . . .	489
9.3.6	Write-Invalidate Protocol using a Directory . . . . .	492
9.3.7	Optimisations and Corner Cases in the Directory Protocol . . . . .	496
9.3.8	Atomic Operations . . . . .	500
9.3.9	Lock-free Algorithms using Atomic Operations . . . . .	509
9.4	Memory Consistency . . . . .	512
9.4.1	Sequential Consistency . . . . .	513
9.4.2	Basic Terminology . . . . .	516
9.4.3	Memory Models . . . . .	524
9.4.4	Safety Conditions for Accesses to a Single Location . . . . .	533
9.4.5	Safety Conditions for Data and Control Dependencies . . . . .	537
9.4.6	Correctness of Executions . . . . .	539
9.5	Data Races . . . . .	539
9.5.1	Critical Sections and Concurrency Bugs . . . . .	539
9.5.2	Data Races in the Context of Memory Models . . . . .	541
9.5.3	Properly Labelled Programs . . . . .	544
9.5.4	Lock Set Algorithm . . . . .	547
9.5.5	Data Race Detection with Vector Clocks . . . . .	549
9.6	Transactional Memory . . . . .	553
9.6.1	Basic Fundamentals . . . . .	557
9.6.2	Correctness Conditions . . . . .	562
9.6.3	Software Transactional Memory . . . . .	565
9.6.4	Hardware Transactional Memory . . . . .	571
9.7	Summary and Further Reading . . . . .	575
9.7.1	Summary . . . . .	575
9.7.2	Further Reading . . . . .	579
<b>10</b>	<b>Main Memory</b> . . . . .	<b>581</b>
10.1	Introduction . . . . .	581
10.2	Dynamic RAMs: Devices, Circuits, and Systems . . . . .	582
10.2.1	DRAM Cell . . . . .	582
10.2.2	Capacitors used in DRAM Cells . . . . .	583
10.2.3	Array of DRAM Cells . . . . .	585
10.2.4	A Computer System with DRAM Cells . . . . .	593
10.3	Design Space of DRAMs . . . . .	598
10.3.1	DRAM Access Protocols . . . . .	598
10.3.2	DDR Generations and Timing . . . . .	603
10.3.3	Buffered DIMMs . . . . .	606
10.4	DRAM Timing . . . . .	611
10.4.1	State Diagram . . . . .	611
10.4.2	Activate and Precharge Commands . . . . .	613
10.4.3	Read Operation . . . . .	615
10.4.4	Write Operation . . . . .	617
10.4.5	Interaction between the Read, Write, and Precharge Operations . . . . .	618
10.4.6	Refresh Operation . . . . .	619

10.4.7	Example of a Protocol . . . . .	620
10.5	Memory Controller . . . . .	620
10.5.1	DRAM Transaction Scheduling . . . . .	622
10.5.2	Address Mapping . . . . .	624
10.5.3	Command Scheduling . . . . .	626
10.6	Emerging Memory Technologies . . . . .	627
10.6.1	Flash Memory . . . . .	628
10.6.2	Ferroelectric RAM (FeRAM) . . . . .	634
10.6.3	MRAM . . . . .	637
10.6.4	Phase Change Memory (PCM) . . . . .	640
10.6.5	Resistive RAM (ReRAM) . . . . .	643
10.6.6	3-D and Embedded Memory Technologies . . . . .	646
10.7	Roofline Model . . . . .	647
10.7.1	Overview . . . . .	647
10.7.2	Adding Ceilings . . . . .	649
10.7.3	Uses of the Roofline Model . . . . .	651
10.8	Summary and Further Reading . . . . .	652
10.8.1	Summary . . . . .	652
10.8.2	Further Reading . . . . .	656
 <b>III Advanced Topics</b>		<b>659</b>
 <b>11 Power and Temperature</b>		<b>661</b>
11.1	Power Consumption Model . . . . .	663
11.1.1	Dynamic Power . . . . .	663
11.1.2	Leakage Power . . . . .	669
11.1.3	Summary . . . . .	673
11.2	Temperature Model . . . . .	674
11.2.1	Overview of the System . . . . .	674
11.2.2	Basic Physics . . . . .	675
11.2.3	The Finite Difference Method (FDM) . . . . .	678
11.2.4	Electrical Analogue of a Heat Transfer Problem . . . . .	679
11.2.5	The Finite Element Method (FEM) . . . . .	680
11.2.6	Green's Functions . . . . .	681
11.3	Power Management . . . . .	683
11.3.1	Managing Dynamic Power . . . . .	683
11.3.2	Managing Leakage Power . . . . .	686
11.4	Temperature Management . . . . .	690
11.4.1	A Typical Placement Problem . . . . .	691
11.5	Summary and Further Reading . . . . .	692
11.5.1	Summary . . . . .	692
11.5.2	Further Reading . . . . .	694
 <b>12 Reliability</b>		<b>695</b>
12.1	Introduction . . . . .	695
12.2	Soft Errors . . . . .	697
12.2.1	Physics of Soft Errors . . . . .	697
12.2.2	Circuit and Device Level Techniques to Mitigate Soft Errors	700
12.2.3	Architecture Level Techniques to Mitigate Soft Errors . .	702



12.3	Inductive Noise . . . . .	708
12.3.1	Basic Physics . . . . .	708
12.3.2	Implementation Issues . . . . .	708
12.4	Faults due to Inherent Nondeterminism . . . . .	709
12.4.1	Sources of Nondeterminism . . . . .	709
12.4.2	Methods to Enforce Determinism . . . . .	710
12.5	Design Faults . . . . .	711
12.5.1	Verification and Validation . . . . .	712
12.5.2	Nature of Design Faults . . . . .	714
12.5.3	Using Signals for Debugging and Post-Silicon Validation . . . . .	717
12.6	Faults due to Parameter Variation . . . . .	718
12.6.1	Introduction to Different types of Parameter Variation . . . . .	718
12.6.2	A Mathematical Model of Parameter Variation . . . . .	721
12.6.3	Methods to Mitigate Parameter Variation at the Archi- tectural Level . . . . .	724
12.7	Hard Errors and Ageing . . . . .	726
12.7.1	Ageing . . . . .	726
12.7.2	Hard Errors . . . . .	727
12.7.3	Failure Rate of the Entire System . . . . .	729
12.7.4	Methods to Reduce or Tolerate Hard Errors . . . . .	730
12.8	Summary and Further Reading . . . . .	730
12.8.1	Summary . . . . .	730
12.8.2	Further Reading . . . . .	732
<b>13</b>	<b>Secure Processor Architectures</b>	<b>735</b>
13.1	Data Encryption . . . . .	736
13.1.1	AES Block Cipher . . . . .	736
13.1.2	RC4 Stream Cipher . . . . .	740
13.1.3	Hardware Implementation . . . . .	741
13.1.4	Symmetric and Asymmetric Ciphers . . . . .	742
13.1.5	Session Keys . . . . .	744
13.2	Hashing and Data Integrity . . . . .	746
13.2.1	Common Cryptographic Attacks . . . . .	746
13.2.2	SHA based Hashing . . . . .	747
13.2.3	Message Authentication Code (MAC) . . . . .	747
13.2.4	Preventing Replay Attacks . . . . .	748
13.3	Secure Architectures . . . . .	749
13.3.1	Security in Traditional Processors . . . . .	749
13.3.2	Hardware Security: Key Concepts . . . . .	751
13.3.3	Design of a Secure Processor . . . . .	754
13.3.4	The Software Environment . . . . .	758
13.3.5	Oblivious RAM . . . . .	760
13.4	Side-Channel Attacks . . . . .	762
13.4.1	Classification of Side Channels . . . . .	763
13.4.2	Type 1: Attacker Monitoring Itself . . . . .	763
13.4.3	Type 2: Attacker Monitoring or Manipulating the Victim using Software Techniques . . . . .	765
13.4.4	Type 3: Attacker Monitoring the Victim by Physically Accessing the System . . . . .	765
13.4.5	Countermeasures . . . . .	766

13.5	Summary and Further Reading . . . . .	766
13.5.1	Summary . . . . .	766
13.5.2	Further Reading . . . . .	768
<b>14</b>	<b>Architectures for Machine Learning</b>	<b>771</b>
14.1	Basics of Deep Learning . . . . .	772
14.1.1	Formal Model of the Learning Problem . . . . .	773
14.1.2	Neural Networks . . . . .	774
14.1.3	Convolutional Neural Networks (CNNs) . . . . .	778
14.2	Design of a CNN . . . . .	781
14.2.1	Overview . . . . .	781
14.2.2	Design Space of Loop Transformations . . . . .	785
14.2.3	Hardware Architectures . . . . .	788
14.3	Intra-PE Parallelism . . . . .	794
14.3.1	1-D Convolution . . . . .	794
14.3.2	2-D Convolution . . . . .	798
14.4	Optimisations . . . . .	800
14.4.1	Reduction of the Computing Time . . . . .	801
14.4.2	Reduction of the Memory Access Time . . . . .	802
14.5	Memory System Organisation . . . . .	803
14.5.1	DRAM+SRAM based Organisation . . . . .	803
14.5.2	Processing in Memory . . . . .	805
14.6	Summary and Further Reading . . . . .	807
14.6.1	Summary . . . . .	807
14.6.2	Further Reading . . . . .	809
<b>IV</b>	<b>Appendix</b>	<b>811</b>
<b>A</b>	<b><i>SimpleRisc</i> ISA</b>	<b>813</b>
<b>B</b>	<b>Tejas Architectural Simulator</b>	<b>815</b>
B.1	Overview . . . . .	815
B.2	Tejas Architectural Simulator . . . . .	816
B.2.1	Design of Tejas . . . . .	816
B.2.2	Semi-event Driven Simulation . . . . .	817
B.2.3	Optimisations and Corner Cases . . . . .	818
B.2.4	Parallelisation . . . . .	819
B.2.5	Evaluation . . . . .	819
<b>C</b>	<b>Intel Processors</b>	<b>821</b>
C.1	Sunny Cove Microarchitecture . . . . .	821
C.1.1	ISA Extensions . . . . .	821
C.1.2	Processor Design . . . . .	821
C.2	Tremont Microarchitecture . . . . .	824
C.3	Lakefield Processor . . . . .	825

---

<b>D AMD Processors</b>	<b>827</b>
D.1 Zen2 Microarchitecture . . . . .	827
D.1.1 Fetch and Decode Logic . . . . .	827
D.1.2 Scheduling and Execution . . . . .	828
D.1.3 Data Caches . . . . .	828
D.1.4 Instruction Retirement . . . . .	829
D.2 Matisse Chip . . . . .	829
<b>E Qualcomm Processors</b>	<b>831</b>
E.1 Compute Cores . . . . .	831
E.2 Accelerators . . . . .	832
<b>F Bibliography</b>	<b>835</b>
<b>Index</b>	<b>859</b>