**Ace 21064 Microprocessor**

Architecture Specification

**r1p0**

**Revision History**

|  |  |  |  |
| --- | --- | --- | --- |
| **Revision** | **Date** | **Description** | **Author** |
| 1.0 | 10/13/2016 | Initial version | Yijun LI |
|  |  |  |  |

**Content**

[1 Ace 21064 Core Basics 6](#_Toc481563056)

[1.1 Overview 6](#_Toc481563057)

[1.2 Feature 6](#_Toc481563058)

[1.3 Components 7](#_Toc481563059)

[1.4 Pipeline architecture of ACE21064 7](#_Toc481563060)

[2 Level 1 Instruction Cache 8](#_Toc481563061)

[2.1 Overview 8](#_Toc481563062)

[2.1.1 Introduction 8](#_Toc481563063)

[2.1.2 Architecture 9](#_Toc481563064)

[2.1.3 Signals 9](#_Toc481563065)

[2.1.4 Submodules 10](#_Toc481563066)

[2.1.5 Timing 10](#_Toc481563067)

[2.2 Instruction Alignment Unit 10](#_Toc481563068)

[2.2.1 Introduction 10](#_Toc481563069)

[2.2.2 Architecture 10](#_Toc481563070)

[2.2.3 Signals 10](#_Toc481563071)

[2.2.4 Submodules 10](#_Toc481563072)

[2.2.5 Timing 10](#_Toc481563073)

[3 Instruction Fetch Unit (IFU) 11](#_Toc481563074)

[3.1 Overview 11](#_Toc481563075)

[3.1.1 Introduction 11](#_Toc481563076)

[3.1.2 Signals 11](#_Toc481563077)

[3.1.3 Submodules 12](#_Toc481563078)

[3.2 Branch Prediction Unit 13](#_Toc481563079)

[3.2.1 Introduction 13](#_Toc481563080)

[3.2.2 Architecture 13](#_Toc481563081)

[3.2.3 Ports 13](#_Toc481563082)

[3.2.4 Timing 14](#_Toc481563083)

[3.2.5 Branch History Table (BHT) 14](#_Toc481563084)

[3.2.6 Pattern History Table (PHT) 16](#_Toc481563085)

[3.3 Return Address Stack (RAS) 17](#_Toc481563086)

[3.4 Branch Target Buffer (BTB) 17](#_Toc481563087)

[3.4.1 Introduction 17](#_Toc481563088)

[3.4.2 BTB Way 17](#_Toc481563089)

[3.4.2.1 Ports 17](#_Toc481563090)

[4 Instruction Decoder Unit(IDU) 18](#_Toc481563091)

[4.1 Overview 18](#_Toc481563092)

[4.1.1 Introduction 18](#_Toc481563093)

[4.1.2 Signals 18](#_Toc481563094)

[4.1.3 Submodules 19](#_Toc481563095)

[5 Renaming Unit 22](#_Toc481563096)

[5.1 Overview 22](#_Toc481563097)

[5.1.1 Introduction 22](#_Toc481563098)

[5.1.2 Signals 22](#_Toc481563099)

[5.1.3 Submodules 23](#_Toc481563100)

[6 Instruction Schedule Unit(ISU) 24](#_Toc481563101)

[6.1 Overview 24](#_Toc481563102)

[6.1.1 Introduction 24](#_Toc481563103)

[6.1.2 Signals 24](#_Toc481563104)

[6.1.3 Submodules 25](#_Toc481563105)

[7 Execution Unit 27](#_Toc481563106)

[7.1 Overview 27](#_Toc481563107)

[7.1.1 Introduction 27](#_Toc481563108)

[7.1.2 Signals 27](#_Toc481563109)

[7.1.3 Submodules 28](#_Toc481563110)

[8 Level 1 data Cache 29](#_Toc481563111)

[8.1.1 Overview 29](#_Toc481563112)

[8.1.2 Introduction 29](#_Toc481563113)

[8.1.3 Signals 29](#_Toc481563114)

[8.1.4 Submodules 30](#_Toc481563115)

[9 Retire Unit 31](#_Toc481563116)

[9.1 Overview 31](#_Toc481563117)

[9.1.1 Introduction 31](#_Toc481563118)

[9.1.2 Signals 31](#_Toc481563119)

[9.1.3 Submodules 32](#_Toc481563120)

[10 Interrupt 33](#_Toc481563121)

[11 Debug Unit 34](#_Toc481563122)

[12 Coprocessor 35](#_Toc481563123)

**Table of Figures**

[Figure 1‑1 Block Diagram of Ace21064 Processor 6](#_Toc481563124)

[Figure 2‑1 iCache hierarchy overview 8](#_Toc481563125)

[Figure 2‑2 ace\_icache block diagram 9](#_Toc481563126)

[Figure 3‑1 BHT Port Timing 15](#_Toc481563127)

**Table of Tables**

[Table 3‑1 fetch unit port signals 11](#_Toc481563128)

[Table 3‑2 ace\_fetch sub-modules 12](#_Toc481563129)

[Table 3‑3 nvmc\_biu ports 13](#_Toc481563130)

[Table 3‑4 BHT Port List 14](#_Toc481563131)

[Table 3‑5 BHT Port List 16](#_Toc481563132)

[Table 4‑1 decode unit port signals 18](#_Toc481563133)

[Table 4‑2 ace\_decode sub-modules 19](#_Toc481563134)

[Table 5‑1 decode unit port signals 22](#_Toc481563135)

[Table 5‑2 ace\_rename sub-modules 23](#_Toc481563136)

[Table 6‑1 decode unit port signals 24](#_Toc481563137)

[Table 6‑2 ace\_decode sub-modules 25](#_Toc481563138)

[Table 7‑1 decode unit port signals 27](#_Toc481563139)

[Table 7‑2 ace\_decode sub-modules 28](#_Toc481563140)

[Table 8‑1 decode unit port signals 29](#_Toc481563141)

[Table 8‑2 ace\_decode sub-modules 30](#_Toc481563142)

[Table 9‑1 decode unit port signals 31](#_Toc481563143)

[Table 9‑2 ace\_decode sub-modules 32](#_Toc481563144)

# Ace 21064 Core Basics

## Overview

ACE 21064 is a RISCV ISA based quard-issue superscalar processor core, which has 12 stage integer pipeline

Figure ‑ Block Diagram of Ace21064 Processor

## Features

* Quard-issue out-of-order 12-stage integer pipeline
* 64bit virtual address, 40bits physical address
* 8KB Virtually indexed physically tagged 2-way set associative instruction cache
* 4KB per-page in memory manager
* G-Share and Pap Tournament Hybird branch predictor
* 8 entry return address stack(RAS)
* 32 entry instruction buffer
* 4-instruction can be decoded in one CPU cycle
* 4-wide rename from 80 physical registers file, which is 65-bits each, renaming in unified method
* the physical register file can be accessed from 11 read port or 7 write port
* speculative and architectural rename maps, with respective RAT and FREELIST
* Store Set based memory dependence prediction
* 32-entry scheduler used in issue stage
* speculative wakeup mechanism
* instruction replay
* centralized and distributed combined reservation station is used for efficiency
* Non-data-capture architecture is used in register read phase
* the reservation station is designed in compressed mode
* 2 simple ALU is used for add sub shift etc.
* 1 complex ALU for multiply and divide operation
* 1 branch unit designed for branch instructions
* 2 address generation unit, designed for load and store operation
* 16-entry load and store queue, to keep data before data cache access
* Dual-port 2-way associative l1 data cache is designed with 8 interleaved banks 4KB each bank
* 64-entry reorder buffer is used to support 8 wide retirement

## Components

## Pipeline architecture of ACE21064



# Level 1 Instruction Cache

## Overview

Level 1 instruction cache of Ace21064 is a two way-associative, virtually indexed and physically tagged cache, each data entry has 32 bytes, 128 cache lines in each way, 40bits physical address is designed for temporary, so 28bits higher address range is kept in tag array of each cache line, Total size of l1icache is 8KB, and memory is managed in 4KB page size.

### Introduction

In order to reduce the hit time of level 1 cache, virtually indexed, physically tagged cache architecture is used, as widely used in modern processor. The overview of icache system of Ace21064 is showing below:



Figure ‑ iCache hierarchy overview

Ace21064 is 64bits architecture, and physical address is remained to 40bits for current design. Totally 1TB address range can be accessed, the memory system is managed in 4KB page size, so the low 12bits of virtual address is used as page offset, due to the two-way associative cache organization the total cache capability is 8KB large. To support the 4-issue pipeline, the fetch stage is designed in 8 instructions per-fetch, so the icache is designed to 8 words per-entry for design convenient. Totally 32bytes needs 5 bits to index each byte in a cache entry, named block offset. The width of index segment can be calculate following the formula:

Or following the page width, 7 bits are left for cache line index, so there are 128 cache line in each way.

According the width of physical address, 28bits need to be kept in tag array of icache, which will be compared with the hit-value of iTLB to determine the fetch operation is hit or not.

### Architecture

Figure ‑ ace\_icache block diagram

### Signals

|  |  |  |  |
| --- | --- | --- | --- |
| Port | Direction | Width | Description |
| Common Interface | | | |
| clock | input | 1 | system clock |
| reset\_n | input | 1 | system reset low active |
| pc\_s0\_i | input | 64 | program counter in pipeline stage 0 |
| pc\_s1\_i | input | 64 | program counter in pipeline stage 1 |
| Interface Between MMU and l1icache | | | |
| itlb\_phystag0\_i | input | 28 | physical tag from itlb0 |
| itlb\_phystag1\_i | input | 28 | physical tag from itlb1 |
| itlb\_miss0\_i | input | 1 | itlb miss flag |
| itlb\_miss1\_i | input | 1 | itlb miss flag |
| l1icache output | | | |
| ic\_stall\_s0\_o | output | 1 | l1icache stall flag |
| ic\_inst\_s1\_o | output | 256 | l1icache data out |
|  | | | |
|  |  |  |  |

### Submodules

### Timing

## Instruction Alignment Unit

### Introduction

### Architecture

### Signals

### Submodules

### Timing

# Instruction Fetch Unit (IFU)

## Overview

### Introduction

This module instances the sub-modules including inst\_align, bob, bpd0, bpd1, brdec, btb etc, The block diagram is shown as below.

### Signals

Table ‑ fetch unit port signals

|  |  |  |  |
| --- | --- | --- | --- |
| Port | Direction | Width | Description |
| clock | input | 1 | system clock |
| reset\_n | input | 1 | system reset low active |
| pc\_f0 | input | 64 | program counter in fetch stage 0 |
| pc\_f1 | input | 64 | program counter in fetch stage 1 |
| signals from retire stage | | | |
| flush\_rt\_i | input | 1 | flush signal from retire stage |
| flush\_pc\_rt\_i | input | 64 | flush PC from retire stage |
| brcond\_vld\_rt\_i | input | 1 | condition branch retired flag |
| brindir\_vld\_rt\_i | input | 1 | indirect branch retired flag |
| brdir\_rt\_i | input | 1 | branch direction from retire stage |
| Signals from iCache | | | |
| inst\_align\_i | input | 256 | instruction fetch group1 |
| icache\_stall\_i | input | 1 | indicate the stall status of icache |
| Temporary | | | |
| inst\_q\_full\_i | input | 1 | come from instruction queue, which indicates the full status |
| Output signal for pc generator | | | |
| branch\_pc\_o | output | 64 | branch target pc from BTB or RAS |
| override\_pc\_o | output | 64 | override pc value from fetch1 stage |
| override\_vld\_o | output | 1 | override valid flag from fetch1 stage |
|  |  |  |  |
| Output signal for instruction queue | | | |
| inst0\_vld\_r | output | 1 | instruction0 valid signal registered from fetch stage1 |
| inst1\_vld\_r | output | 1 | instruction1 valid signal registered from fetch stage1 |
| inst2\_vld\_r | output | 1 | instruction2 valid signal registered from fetch stage1 |
| inst3\_vld\_r | output | 1 | instruction3 valid signal registered from fetch stage1 |
| inst4\_vld\_r | output | 1 | instruction4 valid signal registered from fetch stage1 |
| inst5\_vld\_r | output | 1 | instruction5 valid signal registered from fetch stage1 |
| inst6\_vld\_r | output | 1 | instruction6 valid signal registered from fetch stage1 |
| inst7\_vld\_r | output | 1 | instruction7 valid signal registered from fetch stage1 |
| inst0\_ali\_r | output | 32 | aligned instruction0 registered from fetch stage1 |
| inst1\_ali\_r | output | 32 | aligned instruction1 registered from fetch stage1 |
| inst2\_ali\_r | output | 32 | aligned instruction2 registered from fetch stage1 |
| inst3\_ali\_r | output | 32 | aligned instruction3 registered from fetch stage1 |
| inst4\_ali\_r | output | 32 | aligned instruction4 registered from fetch stage1 |
| inst5\_ali\_r | output | 32 | aligned instruction5 registered from fetch stage1 |
| inst6\_ali\_r | output | 32 | aligned instruction6 registered from fetch stage1 |
| inst7\_ali\_r | output | 32 | aligned instruction7 registered from fetch stage1 |
| inst0\_pc\_f1\_r | output | 64 | pc value of aligned instruction0 registered from fetch stage1 |
| inst1\_pc\_f1\_r | output | 64 | pc value of aligned instruction1 registered from fetch stage1 |
| inst2\_pc\_f1\_r | output | 64 | pc value of aligned instruction2 registered from fetch stage1 |
| inst3\_pc\_f1\_r | output | 64 | pc value of aligned instruction3 registered from fetch stage1 |
| inst4\_pc\_f1\_r | output | 64 | pc value of aligned instruction4 registered from fetch stage1 |
| inst5\_pc\_f1\_r | output | 64 | pc value of aligned instruction5 registered from fetch stage1 |
| inst6\_pc\_f1\_r | output | 64 | pc value of aligned instruction6 registered from fetch stage1 |
| inst7\_pc\_f1\_r | output | 64 | pc value of aligned instruction7 registered from fetch stage1 |

### Submodules

Table ‑ ace\_fetch sub-modules

|  |  |
| --- | --- |
| Module Name | Description |
| brdec | pre-decode logic for branch identification |
| bpd0 | branch prediction stage0 |
| bpd1 | branch prediction stage1 |
| btb | branch target buffer |
| bob | branch order buffer |
| ras | return address stack |
| inst\_q | instruction queue between fetch stage and decode stage |

## Branch Prediction Unit

### Introduction

Ace 21064 use a Tournament predictor

### Architecture



### Ports

Table ‑ nvmc\_biu ports

|  |  |  |  |
| --- | --- | --- | --- |
| Port | Direction | Width | Description |
| Global interface |  |  |  |
| hclk | Input | 1 | Clock |
| hrst\_b | Input | 1 | Reset |
| Memory access bus interface | | | |
| hsel | Input | 1 | Memory access bus interface |
| htrans | Input | 2 | Memory access bus interface |
|  |  |  |  |
|  |  |  |  |
| Slave 1 access bus interface | | | |
| s1\_hsel | Output | 1 | EFC access bus interface |
| s1\_htrans | Output | 2 | EFC access bus interface |
|  |  |  |  |
| Slave 2 access bus interface | | | |
| s2\_hsel | Output | 1 | ROM1 access bus interface |
| s2\_htrans | Output | 2 | ROM1 access bus interface |

### Timing

### Branch History Table (BHT)

BHT in Ace21064 has 10 bits wide, which can record ten branch taken history. And has 1024 entries for different branches.

Read BHT :

Read Address: (Read Index) there are 1K entries in design, so the read index width should be 10 bits from current PC. As we know the RISCV ISA is 32bit width, so the bits [1:0] is reserved. bht\_rd\_index\_i[9:0] comes from cur\_pc\_i[11:2].

Read Data: when we find the entry which the cur\_pc has indexed, read the data out for PHT index. We rename it as bht\_br\_hist\_o[9:0].

Write BHT:

Write Address: (Write Index) we update BHT when the branch instruction is committed (after execute stage, the branch direction is confirmed) we named reference pc as confirmed pc. So bht\_wt\_index\_i[9:0] comes from cm\_pc\_i[11:2]

Write Data: As the definition of BHT (a shifter), the write data is the branch direction of the confirmed pc instruction, (bht\_cm\_brdir\_i)

Write enable: except address and data we need to know when to update the BHT entry, so write enable signal is needed. As we know the BHT entry can only be update when the branch is confirmed for the correct branch history record. (bht\_cm\_brdir\_se\_i) (fixme: when to update the BHT [SuperScalar RISC Processor Design p118])

Table ‑ BHT Port List

|  |  |  |  |
| --- | --- | --- | --- |
| Port | Direction | Width | Description |
| Global interface |  |  |  |
| clk | Input | 1 | Clock |
| rst\_b | Input | 1 | Reset |
| BHT Read Interface | | | |
| bht\_rd\_index\_i | Input | 10 | BHT read index, part of current pc |
| bht\_br\_hist\_o | Output | 10 | Branch history, BHT read data |
| BHT Write Interface | | | |
| bht\_wt\_index\_i | Input | 10 | BHT write index, part of confirmed branch instruction’PC |
| bht\_cm\_brdir\_i | Input | 1 | confirmed branch instruction’s direction |
| bht\_cm\_brdir\_se\_i | Input | 1 | confirmed branch instruction direction shift enable |



Figure ‑ BHT Port Timing

### Pattern History Table (PHT)

BHT in Ace21064 has 10 bits wide, which can record ten branch taken history. And has 1024 entries for different branches.

Read BHT :

Read Address: (Read Index) there are 1K entries in design, so the read index width should be 10 bits from current PC. As we know the RISCV ISA is 32bit width, so the bits [1:0] is reserved. bht\_rd\_index\_i[9:0] comes from cur\_pc\_i[11:2].

Read Data: when we find the entry which the cur\_pc has indexed, read the data out for PHT index. We rename it as bht\_br\_hist\_o[9:0].

Write BHT:

Write Address: (Write Index) we update BHT when the branch instruction is committed (after execute stage, the branch direction is confirmed) we named reference pc as confirmed pc. So bht\_wt\_index\_i[9:0] comes from cm\_pc\_i[11:2]

Write Data: As the definition of BHT (a shifter), the write data is the branch direction of the confirmed pc instruction, (bht\_cm\_brdir\_i)

Write enable: except address and data we need to know when to update the BHT entry, so write enable signal is needed. As we know the BHT entry can only be update when the branch is confirmed for the correct branch history record. (bht\_cm\_brdir\_se\_i) (fixme: when to update the BHT [SuperScalar RISC Processor Design p118])

Table ‑ BHT Port List

|  |  |  |  |
| --- | --- | --- | --- |
| Port | Direction | Width | Description |
| Global interface |  |  |  |
| clk | Input | 1 | Clock |
| rst\_b | Input | 1 | Reset |
| BHT Read Interface | | | |
| bht\_rd\_index\_i | Input | 10 | BHT read index, part of current pc |
| bht\_br\_hist\_o | Output | 10 | Branch history, BHT read data |
| BHT Write Interface | | | |
| bht\_wt\_index\_i | Input | 10 | BHT write index, part of confirmed branch instruction’PC |
| bht\_cm\_brdir\_i | Input | 1 | confirmed branch instruction’s direction |
| bht\_cm\_brdir\_se\_i | Input | 1 | confirmed branch instruction direction shift enable |

## Return Address Stack (RAS)

## Branch Target Buffer (BTB)

### Introduction

4-Way associative branch target buffer is realized in Ace 21064 processor

### BTB Way

#### Ports

|  |  |  |  |
| --- | --- | --- | --- |
| Port | Direction | Width | Description |
| Global interface |  |  |  |
| clk | Input | 1 | Clock |
| rst\_n | Input | 1 | Reset |
| Memory access bus interface | | | |
| hsel | Input | 1 | Memory access bus interface |
| htrans | Input | 2 | Memory access bus interface |
|  |  |  |  |
|  |  |  |  |
| Slave 1 access bus interface | | | |
| s1\_hsel | Output | 1 | EFC access bus interface |
| s1\_htrans | Output | 2 | EFC access bus interface |
|  |  |  |  |
| Slave 2 access bus interface | | | |
| s2\_hsel | Output | 1 | ROM1 access bus interface |
| s2\_htrans | Output | 2 | ROM1 access bus interface |

# Instruction Decoder Unit(IDU)

## Overview

### Introduction

4 wide decoder, 4 instructions can be decoded in one cycle

.

### Signals

Table ‑ decode unit port signals

|  |  |  |  |
| --- | --- | --- | --- |
| Port | Direction | Width | Description |
| clock | input | 1 | system clock |
| reset\_n | input | 1 | system reset low active |
| pc\_f0 | input | 64 | program counter in fetch stage 0 |
| pc\_f1 | input | 64 | program counter in fetch stage 1 |
|  | | | |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  | | | |
|  |  |  |  |
|  |  |  |  |
|  | | | |
|  |  |  |  |
|  | | | |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

### Submodules

Table ‑ ace\_decode sub-modules

|  |  |
| --- | --- |
| Module Name | Description |
| inst\_buf | instruction buffer, the fetch stage will fetch 8 instructions per cycle, but decode stage can only decode 4 instructions one time |
| decoder\_0 | decode unit 0 |
| decoder\_1 | decode unit 1 |
| decoder\_2 | decode unit 2 |
| decoder\_3 | decode unit 3 |

# Renaming Unit

## Overview

### Introduction

4 wide rename from 80 physical registers, both speculative and architectural rename maps maintained

Memory dependence prediction using store sets.

### Signals

Table ‑ decode unit port signals

|  |  |  |  |
| --- | --- | --- | --- |
| Port | Direction | Width | Description |
| clock | input | 1 | system clock |
| reset\_n | input | 1 | system reset low active |
| pc\_f0 | input | 64 | program counter in fetch stage 0 |
| pc\_f1 | input | 64 | program counter in fetch stage 1 |
|  | | | |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  | | | |
|  |  |  |  |
|  |  |  |  |
|  | | | |
|  |  |  |  |
|  | | | |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

### Submodules

Table ‑ ace\_rename sub-modules

|  |  |
| --- | --- |
| Module Name | Description |
| spec\_rat | speculative RAT with 8 read ports(four-wide renaming each instruction has two source register in worst case), 4 write ports(four-wide renaming, each instruction has one destination register in worst case). |
| spec\_freelist | speculative FreeList with 4 read ports(four-wide renaming per cycle), 8 write ports (eight wide retire per cycle). |
| mdp\_ssit | memory dependence predictor store set ID table |
| mdp\_ssit\_dpd\_chk | identifies intra instruction bundle dependencies for store set IDs |
| mdp\_lfst | last fetched store table for memory dependence prediction |
| mdp\_dpd\_chk | does dependence check between the 4 instructions currently in rename stage0 |
|  |  |
|  |  |

# Instruction Schedule Unit(ISU)

## Overview

### Introduction

32-entry scheduler.

Speculative wakeup

One cycle execute instruction: wakeup and select in one cycle, to make RAW dependency instruction can be executed back to back

Multiply cycle execute instruction: delayed wakeup strategy

Instruction replay(replay queue based replay is used)

Several function unit share the same reservation station, combined with centralized and distributed reservation station are used.

Non-data-capture architecture(read source data after instruction was selected in reservation station) is used

Compressed reservation station is choosed.

### Signals

Table ‑ decode unit port signals

|  |  |  |  |
| --- | --- | --- | --- |
| Port | Direction | Width | Description |
| clock | input | 1 | system clock |
| reset\_n | input | 1 | system reset low active |
| pc\_f0 | input | 64 | program counter in fetch stage 0 |
| pc\_f1 | input | 64 | program counter in fetch stage 1 |
|  | | | |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  | | | |
|  |  |  |  |
|  |  |  |  |
|  | | | |
|  |  |  |  |
|  | | | |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

### Submodules

Table ‑ ace\_decode sub-modules

|  |  |
| --- | --- |
| Module Name | Description |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

# Execution Unit

## Overview

### Introduction

.

### Signals

Table ‑ decode unit port signals

|  |  |  |  |
| --- | --- | --- | --- |
| Port | Direction | Width | Description |
| clock | input | 1 | system clock |
| reset\_n | input | 1 | system reset low active |
| pc\_f0 | input | 64 | program counter in fetch stage 0 |
| pc\_f1 | input | 64 | program counter in fetch stage 1 |
|  | | | |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  | | | |
|  |  |  |  |
|  |  |  |  |
|  | | | |
|  |  |  |  |
|  | | | |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

### Submodules

Table ‑ ace\_decode sub-modules

|  |  |
| --- | --- |
| Module Name | Description |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

# Level 1 data Cache

### Overview

### Introduction

.

### Signals

Table 8‑1 decode unit port signals

|  |  |  |  |
| --- | --- | --- | --- |
| Port | Direction | Width | Description |
| clock | input | 1 | system clock |
| reset\_n | input | 1 | system reset low active |
| pc\_f0 | input | 64 | program counter in fetch stage 0 |
| pc\_f1 | input | 64 | program counter in fetch stage 1 |
|  | | | |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  | | | |
|  |  |  |  |
|  |  |  |  |
|  | | | |
|  |  |  |  |
|  | | | |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

### Submodules

Table 8‑2 ace\_decode sub-modules

|  |  |
| --- | --- |
| Module Name | Description |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

# Retire Unit

## Overview

### Introduction

64-entry reorder buffer, with eight-wide retire

Missprediction

Exception

Interrupt

Store operation

storebuffer

### Signals

Table 9‑1 decode unit port signals

|  |  |  |  |
| --- | --- | --- | --- |
| Port | Direction | Width | Description |
| clock | input | 1 | system clock |
| reset\_n | input | 1 | system reset low active |
| pc\_f0 | input | 64 | program counter in fetch stage 0 |
| pc\_f1 | input | 64 | program counter in fetch stage 1 |
|  | | | |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  | | | |
|  |  |  |  |
|  |  |  |  |
|  | | | |
|  |  |  |  |
|  | | | |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

### Submodules

Table 9‑2 ace\_decode sub-modules

|  |  |
| --- | --- |
| Module Name | Description |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

# Interrupt

# Debug Unit

# Coprocessor

Power management

Performance Monitors