Chromite Core Datasheet

Release: 1.2.0

Last update on : Tuesday 10th May, 2022

Table of contents

List of figures

List of tables

Chapter 1

Overview

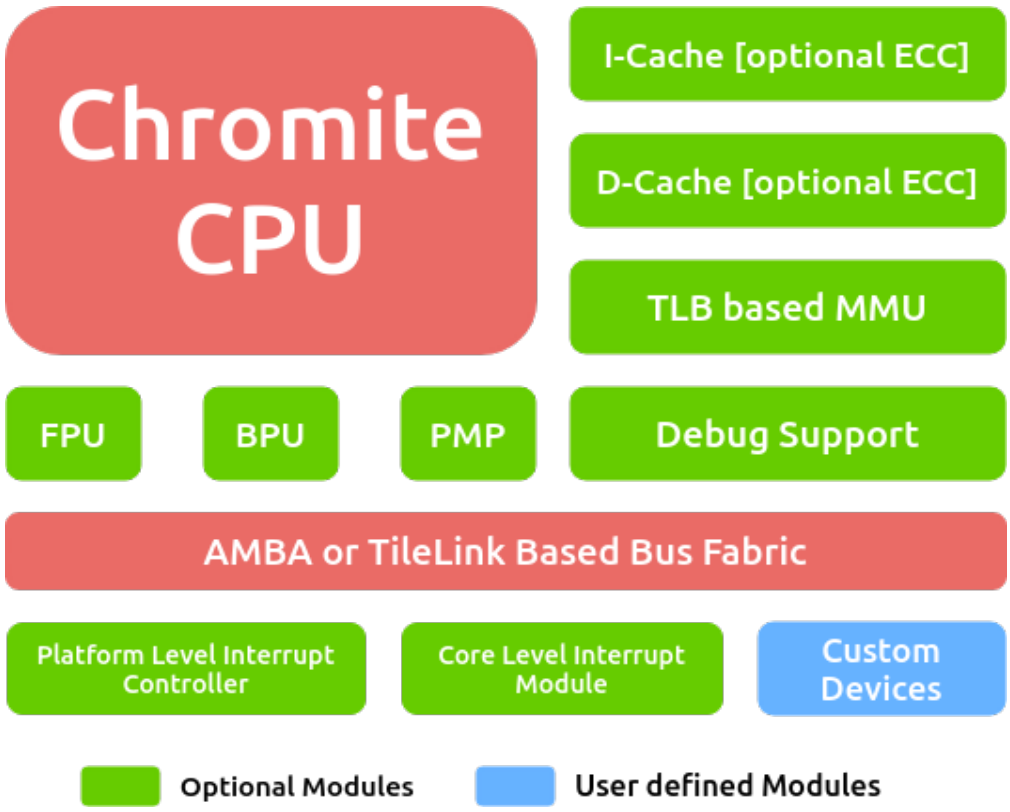

Fig. 1.1: Chromite Processing System

Chromite is a fully configurable and synthesizable Core Generator capable of generating production quality RTL based on the open RISC-V ISA. The core generator can produce variants of a commercial grade 5/6-stage in-order core supporting the RV[64/32]GCSU (or its subsets) extensions of the RISC-V ISA, from the same high-level source code. Chromite leverages the high level abstraction offered by Bluespec System Verilog to build highly parameterized, compact and powerful library components (like arithmetic units, branch predictors, caches, mmu, etc) that can be seamlessly integrated to create a solution catered to your needs. Fig. 1.1 shows the system level diagram of the Chromite core.

The configuration to the generator is provided through a very simple and easy to modify YAML file. The core generator uses a python script to generate the necessary environment variables and macros required to generate the RTL. The

generated RTL is completely synthesizable and can be directly integrated into conventional VLSI flows for FPGA, ASIC or verification. Fig. 1.2 shows the basic flow of the Chromite core generator.

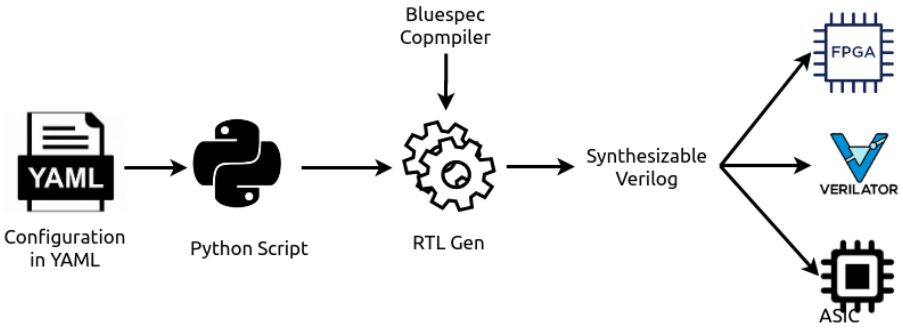

Fig. 1.2: Chromite Core Generator Flow

The various design instances generated through Chromite can serve domains ranging from embedded systems, motor-control, IoT, storage, industrial applications, all the way to low-cost, high-performance Linux based applications such as networking, gateways etc. The extreme parameterization of the design in conjunction with using an HLS like Bluespec, it makes it easy to add new features and design points on a continual basis.

A summary of key features inlcude :

- Support for RV[32|64]IMAFDCSUZicsr_Zifencei (compliant with the latest release of the spec)
- Enhanced fetch micro-architecture to provide higher performance with higher code-density support
- Highly configurable and optional Memory Management Unit
- Configurable Multiply/Divide Units. Can be configured for higher performance or low area.
- Floating Point Unit (FPU), an IEEE-754 compliance Floating-Point Unit with single and double precision support
- Highly configurable and Latency optimized Instruction and Data Caches
- Includes a light-weight, non-intrusive and a full-featured Debug support over JTAG
- Configurable Hardware Breakpoints triggered by instruction/data address/content match, interrupts, exceptions and instruction count.
- A fully configurable RISC-V Privileged architecture. Ability to define and control the behavior of every CSR without breaking compliance to the ISA.
- Physical Memory Protection support as per official ISA spec to provide protection across privilege levels.
- Support for virtualization mode via hypervisor (coming soon)

Chapter 2

Core Level Features

- 5/6 stage in-order pipeline
- 64/32-bit variants available
- Configurable to support the following RISC-V ISA extensions: RV[64/32][IMAFDCSUZicsr_Zifencei]
- Support for enhanced RISC-V privilege features:
  - Vectored interrupts and support for external interrupt controller
  - Programmable trap vector base
  - Atomic interrupt enable/disable feature support
- Supports IEEE-754 based single and double precision Floating Point operations in hardware.
  - The design of the FPU (Floating-Point-Unit) is based on Hard-float. Therefore leverages the berkely recoded-format for efficient implementation to support sub-normals.
  - Single unit per precision to handle fused-multiply-add, multiply, addition and subtraction ops. These units can be configured to meet high-frequency requirements since they are retime-enabled.
  - Runs at 1:1 core/FPU clock ratio.
  - A separate floating point registerfile is used to store floating point operands (in recoded format)
  - Control the number of in-flight instructions by controlling pipeline depths of each module.
- Integrated integer Multiply/Divide units
  - A fully pipelined multiplier. The multiplier is retime-enabled thereby allowing a configurable pipeline depth at build-time.
  - The divider implements an iterative algorithm, with an early out mechanism for corner cases.
- A flexible Co-processor interface to provide closely-coupled integration of custom accelerators.
  - based on on the RoCC specification
  - implements a fire-wait or a fire-and-forget protocol based on the instruction dependency in the program order.
- Supports AXI-4, AXI-4 Lite and TileLink bus protocols
- Configurable L1 Caches
  - Separate instruction and data caches which can be configured with upto 4-ways and 32KiB
  - Optional ECC support for RAMs
  - Write-back with Write-allocate caches
  - Virtually indexed and physically tagged
  - Optimized to use single-ported RAMs
  - Can be disabled/enabled through software
  - Configurable replacement policies: Pseudo LRU, Random, Round-Robin
  - Data Cache has support for atomic extension operations.
- Configurable Memory Management Unit (MMU)
  - Choose between light-weight and configurable fully-associative and set-associative TLB (Translation Looka-side Buffers) architectures for Instruction and Data idependently.
  - Support for super-pages (page size more than 4KiB).
  - Translation happens in paralle with cache lookup - thereby not imposing anye extra penalty.

  - Configurable depth of fully-associative TLBs - which can enable a SW controlled mix of different page sizes in the same capacity.
  - Set-associative architectures provide a data-structure for each page-size (sizes of which is also configurable) thereby allowing a higher capacity per page-size. This choice can also enable better frequency closure for higher capacity
  - Hardware Page-Table-Walk (PTW) support. The page walks are cached to enable higher throughput on a page-miss in the TLBs.
  - The PTW is muxed between the Instruction and Data TLBS, with the latter having higher priority.
- Physical Memory Protection Unit (PMP):
  - Configurable number of regions : upto 16 can be protected.
  - Can protect upto a granularity of 4 bytes for the 32-bit core and 8 bytes for the 64-bit core.
  - Certain regions can be locked for protection against machine level access as well
- Branch Prediction (BPU):
  - Supports Gshare based branch prediction scheme
  - Fully-associative Branch Target Buffer with upto 64-entries
  - Variable size Branch History Buffer
  - Configurable size of Return Address Stack
  - Ability to disable predictor at runtime through software
- Daisy Chained CSRs optimized for frequency and scaling
  - Allows adding of custom CSRs without having to compromise on frequency or features.
  - Can instantiate up to 32 performance counters
- Debug support based on the RISC-V Debug spec v1.0
  - non-intrusive and light-weight implementation
  - interrupt based indication mechanism
  - Includes a program-buffer to execute custom code when halted (in debug-mode)
- Trigger support for data and address.
- Includes Performance counters for :
  - statistics of different types of instructions based on extension
  - statistics of Caches and TLBs (like number of hits, misses, writebacks, etc)
  - statistics of traps (number of interrupts, exceptions, etc)

Chapter 3

Benchmarking the Core

The max DMIPS of the Chromite core is **1.772 DMIPs/MHz.**

The max CoreMarks of the Chromite core is **3.2 CoreMarks/MHz**

The Chromite core is highly configurable and allows workload specific tuning to achieve the maximum performance. This document will highlight some of the settings and their respective benchmark numbers. For the following benchmarks the core has been configured using the default.yaml available in the `samples/` folder.

Note: Make sure you are using gcc 11.1.0 or above to replicate the following results.

3.1 Benchmarking Dhrystone

The following numbers have been obtained via simulation where the number of ITERATIONS was fixed at 10000. The riscv-gnu-toolchain was used to compile the program. The versions used have been populated in the table.

Flags used for compilation:

```
-mcmodel=medany -static -std=gnu99 -O2 -ffast-math \
-fno-common -fno-builtin-printf -march=rv64$(march) -mabi=lp64d \
-w -static -nostartfiles -lgcc
```

The following table provides the DMIPs/MHz numbers for varios configurations for 10K iterations of dhrystone:

HW ISA CONFIG	march	mabi	uArch Configs	gcc version	DMIPs/MHz
RV64IMACSU	rv64imac	lp64	Default	(g5964b5cd727) 11.1.0	1.750
RV64IMACSU/ RV64IMASU	rv64ima	lp64	Default	(g5964b5cd727) 11.1.0	1.767
RV64IMACSU	rv64imac	lp64	over-lap_redirections=True	(g5964b5cd727) 11.1.0	1.756
RV64IMACSU/ RV64IMASU	rv64ima	lp64	over-lap_redirections=True	(g5964b5cd727) 11.1.0	1.772

Note: Enabling the *overlap_redirections* can affect frequency closure in certain nodes as it muxes the redirected PC to the Instruction Memory Subsystem (IMS) obtained via a mis-prediction in the same cycle (as opposed to registering

it before sending it to IMS). Thus, reducing the mis penalty of misprediction by 1. The performance gain obtained by this is visible in the above table.

3.2 Benchmarking CoreMarks

The following numbers have been obtained via simulation where the number of ITERATIONS was fixed at 100

When `$march` is `rv64ima` the CoreMarks/MHz is **3.2**:

```
2K performance run parameters for coremark.
CoreMark Size    : 666
Total ticks      : 31256616
Total time (secs): 31
Iterations/Sec   : 3
Iterations       : 100
Compiler version : riscv64-unknown-elf-11.1.0
Compiler flags   : -mcmodel=medany -DCUSTOM -DPERFORMANCE_RUN=1 -DMAIN_HAS_NOARGC=1 -DHAS_STDIO -DHAS_
↪PRINTF -DHAS_TIME_H -DUSE_CLOCK -DHAS_FLOAT=0 -DITERATIONS=100 -O3 -fno-common -funroll-loops -finline-
↪functions -fselective-scheduling -falign-functions=16 -falign-jumps=4 -falign-loops=4 -finline-limit=1000␣
↪-nostartfiles -nostdlib -ffast-math -fno-builtin-printf -march=rv64imfd -mexplicit-relocs -ffreestanding -
↪fno-builtin -mtune=rocket
Memory location  : STACK
seedcrc          : 0xe9f5
[0]crclist       : 0xe714
[0]crcmatrix     : 0x1fd7
[0]crcstate      : 0x8e3a
[0]crcfinal      : 0x988c
Correct operation validated. See README.md for run and reporting rules.
```

Chapter 4

Core Pipeline Architecture

The core pipeline consists of **PC gen, instruction fetch, decode, execution, write-back and redirection** stages. These modules are described in the following sections.

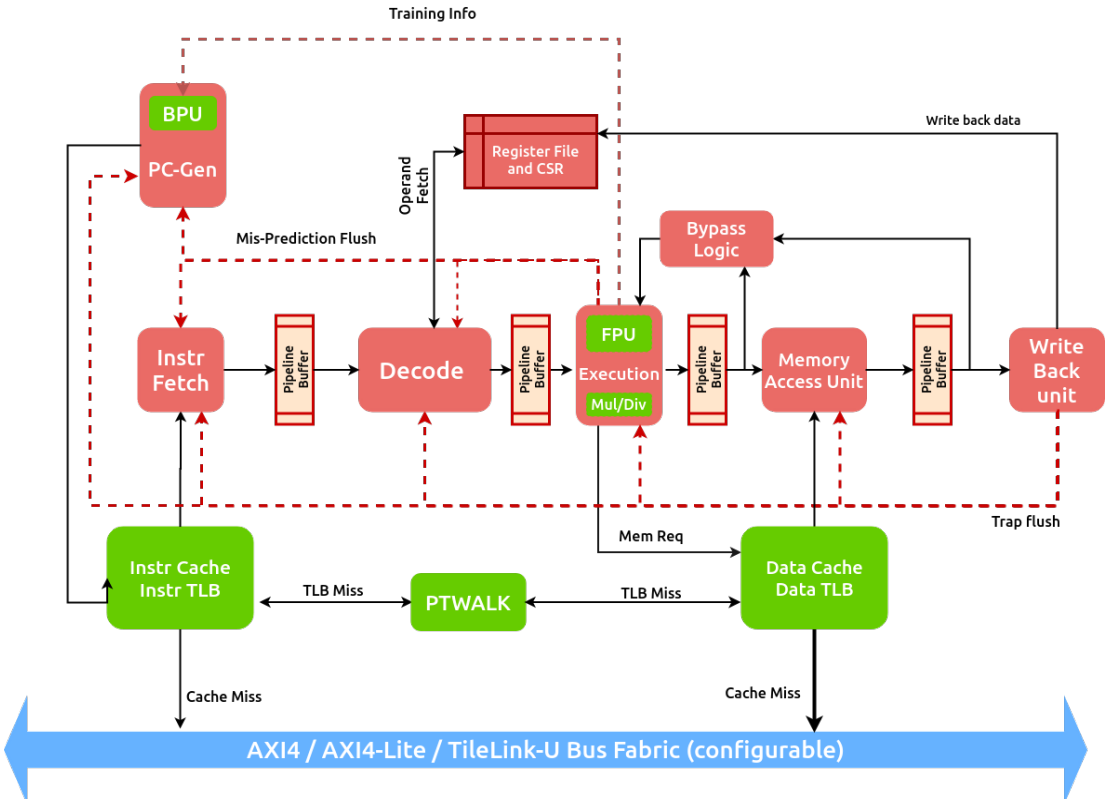

Fig. 4.1: Chromite Pipeline Diagram

4.1 PC Gen Unit

The PC Gen module generates the next value of the Program Counter (PC). It also incorporates the Branch Predictor Unit (BPU). The next PC is generated either as the subsequent intruction-cache line address or as the address obtained from the outputs of the branch predictor and any re-directions received from the pipe in the same cycle. The address calculated by this module decides the next pc for the cache and the instruction fetch-stage.

To achieve high-performance, a gshare based global branch predictor is implemented in this stage.It comprises of a fully-associative Branch Target Buffer (BTB) (up to 64 entries), a Branch History Table (BHT) (up to 512 entries) and a Return Address Stack (RAS) (up to 16 entries) The BTB is trained with all control instructions, but holds the target only for conditional branches and call instructions. The target addresses for return intructions is maintained in the RAS.

When compressed instructions are supported, the BPU provides prediction for 2 PCs. This module is also responsible for ensuring that if an instruction straddles a 4-byte boundary (within the same or different cache line), the correct sequence of addresses are generated to the cache and relevant information is passed to the instruction fetch unit.

4.2 Instruction Fetch Unit

The instruction fetch unit is responsible for fetching the instructions from cache. This unit receives a cache-line size of bytes from the Instruction cache. The fetch-unit stores this cache line to feed the decode-stage with subsequent instructions. This optimization reduces the number of requests sent to the Instruction cache and thereby reduces power as well.

When compressed instructions (16-bit) are supported, the bytes received from the cache can be a combination of both compressed and non-compressed instructions and thus involves a small sate-machine to deduce this information. In case a compressed instruction is detected, it is decompressed to an equivalent 32-bit instruction before being passed on to the decode unit. This unit also detects any instructions triggers, if set. Information of any faults (instruction access or page-faults) received from the cache are simply passed onto the next stage along with the instruction.

4.3 Decode Unit

The decode unit decodes the 32-bit instruction received from the fetch-stage and also performs operand fetch from the register-file. The core consists of 32 general purpose integer registers of XLEN (32/64 bit) size. If floating point support is enabled in hardware, then another 32 floating point registers of FLEN (32/64 bit) size are also available. The integer register-file consists of two read-ports and one write port. The floating point register-file on the other hand requires three read-ports and one write port (to support floating point multiply-accumulate operations). To reduce operation latency in the pipeline, the register file also supports full-bypass.

All interrupts (local or external) are detected in this unit. Illegal traps and traps received from the previous stage are captured here and processed for the next stage.

When a WFI (Wait for Interrupt) instruction is detected, the pipeline is stalled from the next cycle . The pipeline resumes functionality only when an interrupt (local or external) is detected.

Since all CSR operations flush the pipeline, when a CSR instruction or a trap is detected, the decode unit stalls until a re-direction signal is received from either the execution unit or the write-back unit.

4.4 Execution Unit

To increase throughput, the execution unit implements a simple operand-bypass scheme which receives updated operands from the subsequent stages, before they are committed. The unit also instantiates a single-cycle ALU to perform arithmetic and logical operations (e.g. ADD, SUB, XOR, Shift).

All branches and mispredictions are evaluated in this stage and a re-direction signal is generated to the previous stages to re-start fetch and drop fetched instructions. If the branches lead to a mis-aligned address (in cases where compressed is not supported), the module captures them as traps.

The unit also uses a separate 64/32-bit adder to generate the memory access address. If the effective address is mis-aligned then the unit captures it as a trap.

4.4.1 Multiply/Divide Unit

The execution unit utilizes a multi-cycle integer multiply / divide unit to support the M instruction extension of RISC-V. The multiplier is implemented as a retimed module whose latency can be configured at design time. The divider on the other hand implements a non-restoring algorithm which produces output with a variable latency, but a maximum of XLEN (64/32). SRT based high performance dividers are also available and can be chosen at design time.

4.4.2 Floating Point Unit

The optional floating point unit (FPU), compliant with the IEEE-754 2008 standard is also instantiated within the execution unit. The FPU supports single and double precision computations, with denormals handling and all six standard rounding modes.

The FPU uses a retimed fused-multiply-accumulate unit to perform addition, subtraction and FMA operations. The latency of the pipeline can be configured at design time. When double precision is enabled at design time, the unit itself performs the single-precision operations with additional conversion latencies. The FPU uses variable latency, iterative units to perform division and square-root.

4.4.3 Non-Blocking nature of Execute

Even though Chromite is an in-order core, a mild out-of-order execution happens in the execute stage i.e. an instruction is only stalled if :

- the recent value of one of its operands is not available
- if the strucutural unit performing the execution of the instruction is busy.

This allows overlapping the execution of long-latency operations (like integer-divide, floating point, etc) and thereby improving performance.

4.5 Memory and Write-back Units

The memory unit bypasses all non-memory instruction and otherwise waits for a response from the data cache. The write-back stage updates the register file, and also handles traps. In case of traps, the respective CSRs are updated as described in the privileged RISC-V ISA spec, and a re-direction to the trap vector is initiated causing a flush of the pipeline.

4.6 Handling Re-directions

The execution unit and the write-back unit are capable of generating re-direction signals causing the entire pipeline to be flushed. The execution unit generates a redirection in the case of branch mis-prediction (if the BPU is enabled), or for control flow instructions that are taken (if BPU is disabled). The write-back stage on the other hand will generate re-directions for traps if an instruction (such as CSR ops) require a re-run of the subsequent instructions. TODO: What about actual traps and interrupts?

To account for this with least impact on timing and area, the pipeline implements epoch registers within each pipeline stage. The epoch register is maintained constant (madhu reword) for a stream of instructions until a re-direction is generated from pipeline. The re-directions cause the epochs to toggle and thus, each stage will either process the instruction if the epoch values matches or else drop the instruction on a mis-match.

Chapter 5

Modes of Operation

The core supports the following standard modes of operations:

- **Debug**: This is highest level of operation which provides access to all features of the core and the system. Typically, this mode is used for software development and bring-up phases. This mode is available only if the debugging option is enabled at design time.
- **Machine**: This is the highest mode of software execution and is mandatory in all variants generated by the core. The code running in machine mode is inherently trusted and has access to all implementation resources.
- **Supervisor**: This is an optional mode between the machine and user modes, typically used for providing isolation between a supervisor-level operating system and the user execution environment.
- **User**: This is lowest mode of operation where user applications are executed.

Note: Future versions of the core will also include a hypervisor mode, between the machine and supervisor to enable supervisor/OS virtualisation and stronger security semantics.

Chapter 6

Trap Handling

Traps within the Chromite core can be either synchronous traps or interrupts. The synchronous traps are generated by exception causing instructions through various stages of the pipeline. Examples - Instruction Access Fault, Page Faults, Mis-aligned Faults, Illegal Instructions.

The Chromite core supports three basic classes of interrupts: timer interrupts, software interrupts and external interrupts. The timer interrupt is generated through an on-chip timer module. The software interrupt is generated by setting specific bits within the Core-Level-Interrupt (CLINT) module. The external interrupts are generated by a Platform Level Interrupt Controller (PLIC) after capture of external events.

Each of the above class of interrupts can be generated in any of the operation modes - Machine, Supervisor or User. Chromite cores also provide the ability to delegate handling of some of the traps/interrupts to a lower operation mode using standard delegation CSRs.

The core also supports vectored mode of interrupts, where the trap-vector is generated based on the cause of the trap, allowing fast access to interrupt routines.

Chapter 7

Memory Subsystem

The Chromite memory subsystem includes the L1 caches and a TLB based memory management unit.

7.1 L1 Caches

The L1 caches include separate instruction and data caches. The caches are blocking in nature. Each of the caches can be configured for upto 32KiB with the following parameters:

- n_ways: number of ways in the cache
- n_sets: number of sets per way
- n_blocksize: number of bytes per block
- n_fillbuffer: number of fill-buffer entries
- n_buswidth: size of the bus connected to the caches
- replacement policy: Pseudo LRU, RoundRobin or Random
- ECC support: enable or disable for either of the caches

When supervisor mode is enabled, to prevent aliasing issues, the size of each way should not cross 4kiB.

The caches are designed to use single ported (i.e. 1RW) RAM structures for better delay, area and power features. The access to the RAMs require two cycles: one cycle for the actual read of the RAM and another cycle for tag comparison, hit determination and way selection.

The caches also include an array of fill-buffers which hold the lines coming from the lower level memories. The entries in the fill-buffer are released into the RAMs either when the fill-buffer is full or when there is an opportunity where the cache is not receiving requests from the pipeline.

In case of the data cache, the write-policies followed are write-back and write-allocate. The fill-buffers in the data-cache also hold the lines on which stores need to be performed.

The load to use latency of the data-cache is 1 clock cycle.

7.1.1 Cache as RAMs

Certain ways of the caches can be configured to act as scratchpad RAM or as Tightly Integrated Memories (TIMs). The default configuration on reset can be defined at design time. The runtime configuration can be changed through software. Note, at any point of time at least one-way should act as a cache. The runtime switch from cache to TIMs is achieved by simply manipulating the replacement policies. The transition involves minimal overheads.

7.1.2 ECC

The caches can also be configured to be ECC protected, with the granularity being 64/32 bits. This is based on the data width of the processor. The data cache supports single-correction and double-error detection, while the instruction cache only supports single and double error detection. Information of all corrections and detections across the tags and data are logged into a separate Core Control and Status unit, which can be used by the software to perform further analysis. The RAMs within the caches also provided with side-band interfaces which allow the software to directly write data, tags and parity into the RAMs to check for faults.

In the instruction cache, if a single or a double error is detected, the RAM line is invalidated and fetched from the lower memories. For the data cache, in case a single-error is detected in the RAM line, it is corrected and copied to the fill-buffer and invalidated from the RAMs. In case a double-error is detected the line is invalidated, data is lost (in case that line was dirty) and a miss is generated for the same request.

7.2 MMU

The Chromite core has a memory management unit which includes separate instruction and data TLBs. The core supports the sv32 virtualisation scheme of RISC-V ISA for 32-bit cores, with maximum 32-bit physical address and the sv39/sv48 virtualisation schemes for 64-bit cores with maximum physical address size of 56-bits. The sv32 scheme supports 4KiB and 4MiB pages, sv39 supports 4KiB, 2MiB and 1GiB page sizes. The sv48 additionally supports 512GiB page sizes.

The TLBs are implemented as fully-associative buffers, the size of which can be configured at compile time. Apart from converting virtual addresses to physical addresses, the TLBs also perform the task of physical memory protection.

Chapter 8

Bus Interface and Fabric

The Chromite core can be configured to implement any of the following bus interfaces along with a choice of fabric (crossbar, simple-bus or Mesh).

- AXI-4: A high performance AMBA protocol for fast memory transfers and high bandwidth.
- AXI-4 Lite: A low overhead version of AXI-4 targeted for slow peripherals and devices.
- TileLink: An open standard alternative to AXI and is highly customizable. InCore provides an open source implementation.

Each of the above protocols and fabrics come with highly parameterized bridge adaptors to transact between different protocols and varying bus-sizes, thereby optimizing area, power and frequency.

Chapter 9

Interrupt Controller

The Chromite system also comes with a Platform Level Interrupt Controller (PLIC). The PLIC routes all the external interrupts to the core.

- The PLIC allows each interrupt to be enabled or disabled
- Each interrupt also has a runtime or design time configurable priority. Higher priority interrupts are serviced first
- The software application can also define a priority threshold. Interrupts with priority lower than the threshold will not latch an interrupt to the system
- PLIC supports both level and edge-triggered interrupts

This is an optional module and users can choose to connect their own interrupt controller to drive the external-interrupt source to the core.

Chapter 10

Debug Support

The Chromite core includes a JTAG based system level debugger, based on the RISC-V Debug spec [Spe20a]. It can be used for debugging of applications and kernel code. In addition to standard operating modes (machine, supervisor, user), enabling debug support adds the debug mode of operation to the core.The Debug mode is entered through debug exceptions: *halt request from the host, executing ebreak instructions, single-step execution, breakpoints*. The debug mode can be exited by executing a **dret instruction** or through a resume request from the host.

Within the debug mode the host has access to all the integer registers, floating point registers (if F or D extensions are enabled at design time) and all the CSRs through a dedicated side-band access. Moreover, the host has access to all memory mapped registers of the devices that are in the SoC through the master bus interface.

The Debug support also includes host side GDB and OpenOCD software (ports of the standard source) to load and debug programs.

Chapter 11

Software Ecosystem

Since the Chromite core is based on the RISC-V ISA, the users can leverage the entire open source RISC-V toolchain which includes the following:

- Object toolchain:
  - Binutils
  - LLVM
  - Cranelift
- Debugging:
  - GDB
  - OpenOCD
  - Platform IO IDE
- Compilers and Libraries
  - GCC
  - Clang/LLVM
  - Glibc
  - Newlib
- Bootloaders:
  - U-Boot
  - Coreboot
  - Open-SBI
- OS and OS Kernels:
  - Linux
  - FreeRTOS
  - Zephyr RTOS
  - NuttX
  - Sel4
- Compilers and runtimes for other languages:
  - Go
  - OCaml
  - Rust
- Machine Learning/AI
  - TensorFlow Lite

Chapter 12

Build Time Configurations

The Chromite Core allows a wide variety of features to be customized based on the target application. Table 12.1 captures most of the key configurations that can be selected when the core is synthesized and implemented. Note that the choice of these options does not affect the software application porting.

Table 12.1: Chromite build time configurations

Parameter	Choices	Description
Core Level Configuration		
Processor Width (XLEN)	32/64	implement a 32-bit or a 64-bit core. Defines the size of the register files and interface to the caches.
ISA Extensions (ISA)	Combination of : A, M, F, D, C, S, U, Zifencei, Zicsr	• A: enable Atomic extension support • M: enable MulDiv extension support • F: enable Single precission support • D: enable Double precission support • C: enable Compressed extension support • S: enable Supervisor extension support • U: enable User extension support • Zifence : enable fence for instructions • Zicsr : enable CSR operations
System Level Configurations		
Physical Address (PADDR)	Integer	Size physical address on the SoC bus
Debug Vector	PADDR sized integer	the address of the where the self-loop for the debugger lies
Reset PC	Integer	The Value of the Reset PC from where instructions will be fetched

Continued on next page

Table 12.1 – continued from previous page

Parameter	Choices	Description
Debug support	Enabled/Disabled	Debug support available on the system and core or not.
Co-Processor Interface	Enabled/Disabled	If enabled instantiates a cutom co-processor inter-face to the core.
Supervisor Extension Configurations		
Supervisor Mode	sv32, sv39, sv48, sv57	the virualisation scheme to be adopted for Supervisor translation
ITLB size	integer < 64	Number of entries in the Instruction TLB
DTLB size	integer < 64	Number of entries in the Data TLB
ASID size	integer <9 for 32-bit OR integer <16 for 64-bit	Size of the Address Space ID used in the SATP csr
Multiplier/Divider Configuration		
Multipler Stages	integer < XLEN	the number of cycles the multiplier takes for execution
Divider Stages	integer < XLEN	the number of cycles the divider takes for execution
Floating Point Configuration		
Inflight instructions	ordering_depth > 0	Number of inflight Floating point instructions in the pipeline
SPFMA Stages	in > 0 out > 0 configuration of each stage	Number of retime registers at input Number of retimr registers at output Configuration of each stage
DPFMA Stages	in > 0 out > 0 configuration of each stage	Number of retime registers at input Number of retimr registers at output Configuration of each stage
Branch Predictor Configurations		
Enable	True/False	if the branch predictor should be instantiated inthe design or not.
Predictor Type	Gshare or Bimodal	the type of predictor to be incorporated
On Reset	Enabled/Disabled	if the BPU should be enabled/disabled on Reset
Branch Target Buffer size	Integer	the number of entries in the Branch Target Buffer
Branch History Table size	Integer	the number of entries in the Branch History Table
History Length	Integer	The number of history bits to be used in Gshare mode.
Extra History Bits	Integer	the number of speculative history bits to be used in Gshare mode.
Return Address Stack Size	Integer	the number of entries in the RAS
Instruction Cache Configurations		
Enable	True/False	if the I$ should be instantiated within the design or not.
On Reset	Enabled/Disabled	if the I$ should be enabled or disabled on reset.
Sets	Integer	Number of sets per way of the I$
Block Size	Integer	Number of 32-bit words per cache-block

Continued on next page

Table 12.1 – continued from previous page

Parameter	Choices	Description
Ways	Integer	Number of ways in the I$
Fill buffer size	Integer	Number of entries in the Fill-Buffer
Replacement Policy	PLRU, RoubdRobin, Random	The replacement policy to be implemented within the I$
Banks for Data RAMs	Integer	Number of banks the data blocks should be divided into
ECC	Enabled/Disabled	Is ECC enabled or disabled for the I$
Data Cache Configurations		
Enable	True/False	if the D$ should be instantiated within the design or not.
On Reset	Enabled/Disabled	if the D$ should be enabled or disabled on reset.
Sets	Integer	Number of sets per way of the D$
Block Size	Integer	Number of words per cache-block
Ways	Integer	Number of ways in the D$
Fill buffer size	Integer	Number of entries in the Fill-Buffer
Replacement Policy	PLRU, RoubdRobin, Random	The replacement policy to be implemented within the D$
Banks for Data RAMs	Integer	Number of banks the data blocks should be divided into
ECC	Enabled/Disabled	Is ECC enabled or disabled for the D$
PLIC Configurations		
Number of interrupts	Integer < 1024	Number of interrupts that can be processed by the PLIC
Number of Priority Levels	Integer	Number of priority levels within the PLIC
Number of NMI	Integer	Number of NMI interrupts connected to PLIC
Phyiscal Memory Protection (PMP)		
Enable	True/False	If PMP support is present or not
Entries	Integer < 16	Number of regions for pmp
Granularity	Integer >= 4	Granularity in terms of bytes. Minumim is 4 for 32-bit cores and 8 for 64-bit cores

Chapter 13

Revisions

Version [1.2.0]

- revised pipeline working
- improved features list
- added configurator flow
- improved modes description
- adding a summary of key features in overview
- correct non-blocking nature of execute stage

Version [1.1.0]

- Added granularity constraints for PMP in feature list
- Updated options for PMP support
- Added Revisions section

Bibliography

[Spe20a] RISC-V ISA Debug Specification. *https://riscv.org/specifications/debug-specification/*, 2020.

[Spe20b] RISC-V ISA Privileged Specification. *https://riscv.org/specifications/privileged-isa/*, 2020.

[Spe20c] RISC-V ISA Unprivileged Specification. *https://riscv.org/specifications/isa-spec-pdf/*, 2020.