The ZAP Processor User Guide

Version 0.1
April 2017

THIS PROJECT IS IN AN EXPERIMENTAL STATE.

The GNU General Public License

Preamble

The licenses for most software are designed to take away your freedom to share and change it. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change free software--to make sure the software is free for all its users. This General Public License applies to most of the Free Software Foundation's software and to any other program whose authors commit to using it. (Some other Free Software Foundation software is covered by the GNU Lesser General Public License instead.) You can apply it to your programs, too.

When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for this service if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs; and that you know you can do these things.

To protect your rights, we need to make restrictions that forbid anyone to deny you these rights or to ask you to surrender the rights. These restrictions translate to certain responsibilities for you if you distribute copies of the software, or if you modify it.

For example, if you distribute copies of such a program, whether gratis or for a fee, you must give the recipients all the rights that you have. You must make sure that they, too, receive or can get the source code. And you must show them these terms so they know their rights.

We protect your rights with two steps: (1) copyright the software, and (2) offer you this license which gives you legal permission to copy, distribute and/or modify the software.

Also, for each author's protection and ours, we want to make certain that everyone understands that there is no warranty for this free software. If the software is modified by someone else and passed on, we want its recipients to know that what they have is not the

original, so that any problems introduced by others will not reflect on the original authors' reputations.
Finally, any free program is threatened constantly by software patents. We wish to avoid the danger that redistributors of a free program will individually obtain patent licenses, in effect making the program proprietary. To prevent this, we have made it clear that any patent must be licensed for everyone's free use or not licensed at all.

The precise terms and conditions for copying, distribution and modification follow.

TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION

0. This License applies to any program or other work which contains a notice placed by the copyright holder saying it may be distributed under the terms of this General Public License. The "Program", below, refers to any such program or work, and a "work based on the Program" means either the Program or any derivative work under copyright law: that is to say, a work containing the Program or a portion of it, either verbatim or with modifications and/or translated into another language. (Hereinafter, translation is included without limitation in the term "modification".) Each licensee is addressed as "you".

Activities other than copying, distribution and modification are not covered by this License; they are outside its scope. The act of running the Program is not restricted, and the output from the Program is covered only if its contents constitute a work based on the Program (independent of having been made by running the Program). Whether that is true depends on what the Program does.

1. You may copy and distribute verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice and disclaimer of warranty; keep intact all the notices that refer to this License and to the absence of any warranty; and give any other recipients of the Program a copy of this License along with the Program.

You may charge a fee for the physical act of transferring a copy, and you may at your option offer warranty protection in exchange for a fee.

2. You may modify your copy or copies of the Program or any portion of it, thus forming a work based on the Program, and copy and distribute such modifications or work under the terms of Section 1 above, provided that you also meet all of these conditions:

a) You must cause the modified files to carry prominent notices stating that you changed the files and the date of any change.

b) You must cause any work that you distribute or publish, that in whole or in part contains or is derived from the Program or any part thereof, to be licensed as a whole at no charge to all third parties under the terms of this License.

c) If the modified program normally reads commands interactively when run, you must cause it, when started running for such interactive use in the most ordinary way, to print or display an announcement including an appropriate copyright notice and a notice that there is no warranty (or else, saying that you provide a warranty) and that users may redistribute the program under these conditions, and telling the user how to view a copy of this License. (Exception: if the Program itself is interactive but does not normally print such an announcement, your work based on the Program is not required to print an announcement.)

These requirements apply to the modified work as a whole. If identifiable sections of that work are not derived from the Program, and can be reasonably considered independent and separate works in themselves, then this License, and its terms, do not apply to those sections when you distribute them as separate works. But when you distribute the same sections as part of a whole which is a work based on the Program, the distribution of the whole must be on the terms of this License, whose permissions for other licensees extend to the entire whole, and thus to each and every part regardless of who wrote it.

Thus, it is not the intent of this section to claim rights or contest your rights to work written entirely by you; rather, the intent is to exercise the right to control the distribution of derivative or collective works based on the Program.

In addition, mere aggregation of another work not based on the Program with the Program (or with a work based on the Program) on a volume of a storage or distribution medium does not bring the other work under the scope of this License.

3. You may copy and distribute the Program (or a work based on it, under Section 2) in object code or executable form under the terms of Sections 1 and 2 above provided that you also do one of the following:

a) Accompany it with the complete corresponding machine-readable source code, which must be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,

b) Accompany it with a written offer, valid for at least three years, to give any third party, for a charge no more than your cost of physically performing source distribution, a complete machine-readable copy of the corresponding source code, to be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,

c) Accompany it with the information you received as to the offer to distribute corresponding source code. (This alternative is allowed only for noncommercial distribution and only if you received the program in object code or executable form with such an offer, in accord with Subsection b above.)

The source code for a work means the preferred form of the work for making modifications to it. For an executable work, complete source code means all the source code for all modules it contains, plus any associated interface definition files, plus the scripts used to control compilation and installation of the executable. However, as a special exception, the source code distributed need not include anything that is normally distributed (in either source or binary form) with the major components (compiler, kernel, and so on) of the operating system on which the executable runs, unless that component itself accompanies the executable.

If distribution of executable or object code is made by offering access to copy from a designated place, then offering equivalent access to copy the source code from the same place counts as distribution of the source code, even though third parties are not compelled to copy the source along with the object code.

4. You may not copy, modify, sublicense, or distribute the Program except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense or distribute the Program is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

5. You are not required to accept this License, since you have not signed it. However, nothing else grants you permission to modify or distribute the Program or its derivative works. These actions are prohibited by law if you do not accept this License. Therefore, by modifying or distributing the Program (or any work based on the Program), you indicate your acceptance of this License to do so, and all its terms and conditions for copying, distributing or modifying the Program or works based on it.

6. Each time you redistribute the Program (or any work based on the Program), the recipient automatically receives a license from the original licensor to copy, distribute or modify the Program subject to these terms and conditions. You may not impose any further restrictions on the recipients' exercise of the rights granted herein. You are not responsible for enforcing compliance by third parties to this License.

7. If, as a consequence of a court judgment or allegation of patent infringement or for any other reason (not limited to patent issues), conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot distribute so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not distribute the Program at all. For example, if a patent license would not permit royalty-free redistribution of the Program by all those who receive copies directly or indirectly through you, then the only way you could satisfy both it and this License would be to refrain entirely from distribution of the Program.

If any portion of this section is held invalid or unenforceable under any particular circumstance, the balance of the section is intended to apply and the section as a whole is intended to apply in other circumstances.

It is not the purpose of this section to induce you to infringe any patents or other property right claims or to contest validity of any such claims; this section has the sole purpose of protecting the integrity of the free software distribution system, which is implemented by public license practices. Many people have made generous contributions to the wide range of software distributed through that system in reliance on consistent application of that system; it is up to the author/donor to decide if he or she is willing to distribute software through any other system and a licensee cannot impose that choice.

This section is intended to make thoroughly clear what is believed to be a consequence of the rest of this License.

8. If the distribution and/or use of the Program is restricted in certain countries either by patents or by copyrighted interfaces, the original copyright holder who places the Program under this License may add an explicit geographical distribution limitation excluding those countries, so that distribution is permitted only in or among countries not thus excluded. In such case, this License incorporates the limitation as if written in the body of this License.

9. The Free Software Foundation may publish revised and/or new versions of the General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Program specifies a version number of this License which applies to it and "any later version", you have the option of following the terms and conditions either of that version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of this License, you may choose any version ever published by the Free Software Foundation.

10. If you wish to incorporate parts of the Program into other free programs whose distribution conditions are different, write to the author to ask for permission. For software which is copyrighted by the Free Software Foundation, write to the Free

Contents

0 Running Simulations

This chapter gives a brief introduction on how to run simulations.

0.1 Pre-requisites

This section assumes that your system meets the following requirements:

- Has Perl, Icarus Verilog and GTKWave installed in a Linux environment.
- Has the latest version of bare-metal GCC installed compatible with ARM7 (ARMv4T architecture if you need Thumb, else ARMv4 is sufficient).
- The environment variable `ZAP_HOME` points to the root directory of the project.

0.2 Automated Script

A Perl script present in `$ZAP_HOME/hw/sim/run_sim.pl` undertakes the task of calling external programs to compile C/ASM code, link it and calling Icarus Verilog to simulate the RTL.

0.2.1 Script Options

The `run_sim.pl` script command call has a general format of…

```
perl run_sim.pl +opt1+[val1] +opt2+[val2] +opt3+[val3] … +optN+[valN]  …
```

Supported options are shown in the table below (You can type `perl run_sim.pl` without any command line arguments for a list of options)…

Table 1. Script Options

Option	Meaning
`+zap_root+<root_directory>`	Root directory of the ZAP project.
`[+seed+<seed_value>]`	Force a specific seed for simulation.
`[+sim]`	Force register file debug and some extra error messages.
`+test+<test_case>`	Run a specific test case. only `+test+factorial` is available by default, you may add new tests if you wish.
`[+cmmu_en]`	Enable cache and MMU (This feature is very experimental).
`+ram_size+<ram_size>`	Set size of RAM in bytes in simulation.
`+dump_start+<start_addr_of_dump>+<number_of_words_in_dump>`	Starting memory address to start logging and number of words to log.
`[+cache_size+<data_cache_size>+<code_cache_size>]`	Specify data and instruction cache size in bytes.

`[+dtlb+<section_dtlb_entries>+<small_page_entries>+<large_page_entries>]`	Specify data TLB entries for section, small and large page TLBs.
`[+itlb+<section_itlb_entries>+<small_page_entries>+<large_page_entries>]`	Specify instruction TLB sizes (number of entries).
`[+irq_en]`	Trigger IRQ interrupts from bench.
`[+fiq_en]`	Trigger FIQ interrupts from bench.
`+scratch+<scratch_dir>`	Set scratch directory. Usually set this to /tmp. VCD and logs go scratch.
`+max_clock_cycles+<max_clock_cycles>`	Set maximum clock cycles for which the simulation should run.
`+rtl_file_list+<rtl_file_list>`	Specify RTL file list. See hw/rtl folder for the file list (rtl_files.list).
`+tb_file_list+<tb_file_list>`	Specify testbench file list. See hw/tb folder (tb_files.list).
`+bp+<branch_predictor_entries>`	Number of entries in branch predictor memory.
`+fifo+<fifo_depth>`	Depth of prefetch FIFO in the processor.
`+post_process+<post_process_perl_script_path>`	Point this to post_process.pl or any other Perl script. Script runs after simulation is complete.
`+nodump`	Do not write VCD output.

0.2.2 Running Default Testcase

A default factorial program is included with the processor. The sample program essentially calculates the factorial of 5 (that is stored at byte address 2000 (decimal)) and writes the result to byte address 2001. The 32-bit resulting value starting at location 2000 must be 0x00007805 where address 2000 contains 5, 2001 contains 0x78 and both 2002 and 2003 are 0x0. After factorial calculation, some multiplications are also performed. The code switches on cache and MMU and uses identity mapping using a section descriptor placed at 16KB. If MMU/cache is not used, the CP15 instructions will trigger an undefined instruction exception which simply does nothing in the subroutine and returns control to the program.

To run the default testcase, use the following command...

```
perl run_sim.pl +zap_root+$ZAP_HOME +sim +test+factorial +ram_size+32768
+dump_start+1992+10 +scratch+/tmp +irq_en +max_clock_cycles+100000 +bp+1024
+fifo+4 +rtl_file_list+../rtl/rtl_files.list +tb_file_list+../tb/bench_files.list
+post_process+post_process.pl
```

After running the simulation, a memory dump of locations 1992 to 2002 will be displayed.

1 Introduction

This chapter presents an overview of the ZAP processor.

1.1 Overview

ZAP is a 32-bit ARMv4T compatible open source soft processor core fitted with I and D caches that are also capable of virtual memory management. ZAP is binary compatible with the 32-bit ARMv4 instruction and the 16-bit Thumb v1 instruction set. Memory interfaces are compatible with the Wishbone B3 specification.

The processor features a 9 stage pipeline as shown in the figure below. The pipeline has an extensive bypass network built into it to minimize unnecessary pipeline stalls. A load accelerator allows data to be forwarded from the memory directly to issue. Most non-multiply data processing instructions are single cycle and can be executed back-to-back without interlocks. The exceptions to this are when non-trivial shifts are used. The following code takes 3 cycles to execute:

```
ADD R1, R2, R3
ADD R4, R5, R1 LSL #1
```

If the second register is not source shifted, there is no data dependency check. Thus, the following code takes 2 cycles to execute:

```
ADD R1, R2, R3
ADD R4, R1, R9 LSL #1
```

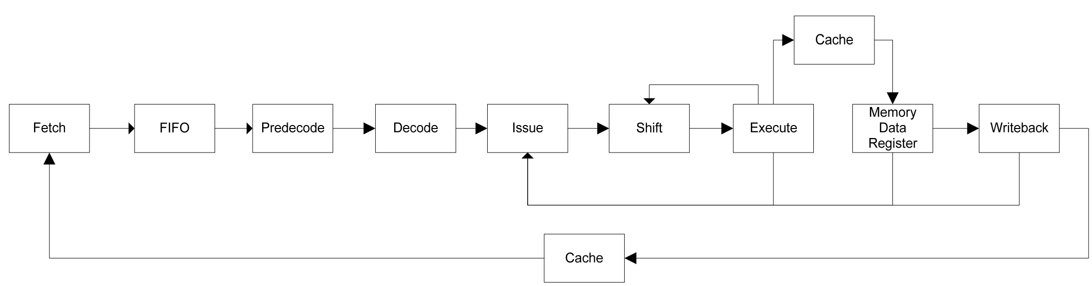

Figure 1. The Basic Pipeline Structure

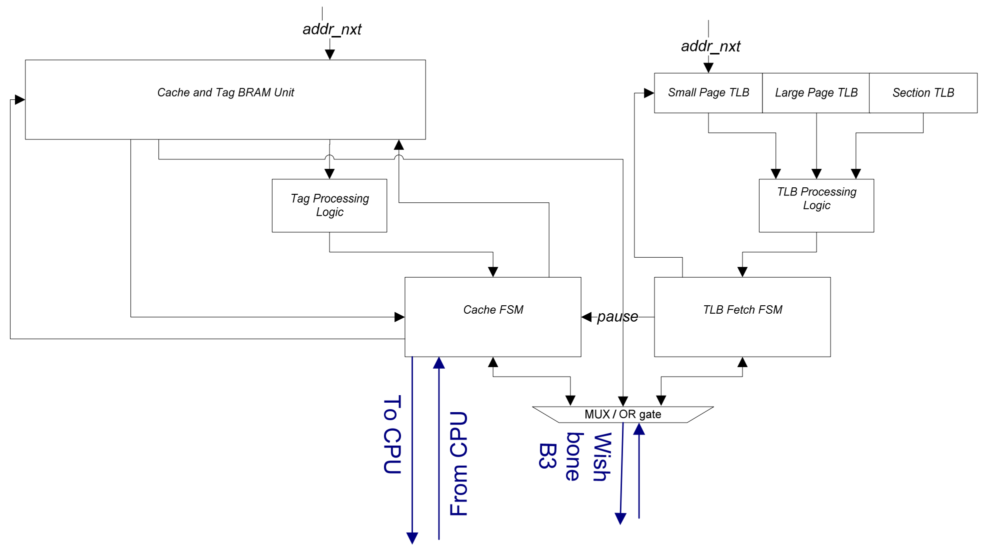

Figure 2. Cache Design

Short multiplication (32x32+32=32) takes 6 cycles while long multiplication (32x32+64=64) takes 12 cycles to execute.

The table below briefly describes each stage.

Table 2. Pipeline Description

Stage	Purpose
Fetch	Clocks data in from the instruction cache into the instruction register.
FIFO	Instructions from fetch are clocked into a shallow FIFO.
Predecode	Decodes Thumb v1 instructions and sequences complex ARM instructions (LDM, STM, long multiplication etc). Also performs branch prediction.
Decode	Decodes ARM instructions.
Issue	Performs operand read from the bypass network and register file.
Shift	Performs shift operations. Also contains the multiplier state machine. Contains a single level bypass network from the ALU

	to allow back-to-back operations without stalling.
Execute	Contains the ALU.
Memory Data Register (Simple called *memory*)	Clocks in data from cache into the data register.
Writeback	Writes new values into the register file. The program counter is present here.

1.2 Features

- Binary compatible with ARMv4 and Thumb v1 code. Supports M variant instructions.
- LDR/STR instructions with base update can issue in a single cycle instead of issuing as a memory access and an ALU operation. To allow this, the data address bus can be picked off either after the ALU or at the input of the ALU based on the instruction.
- High performance 9 stage pipeline with extensive bypass network. A load accelerator improves memory read performance.
- Supports two Wishbone B3 memory interfaces, one for instructions and the other for data. Cache uploads and downloads are down as incrementing bursts using the CTI signal.
- Supports direct mapped I and D caches with memory management capabilities. Dedicated TLBs for 1M, 4K and 64K pages. The size of the TLBs and caches may be configured using parameters. TLBs are all direct mapped as well. Note that instruction and data accesses use separate TLBs.
- Caches are writeback to improve performance. Each cache line is 16 byte long and has a dirty bit. The physical address is stored in the cache along with the cache line to prevent the possibility of needing to walk the page table during global cache cleaning. Supports single cycle cache invalidation.
- The cache and the memory management subsystem may be configured using CP15 in a similar way as other ARM processors in the v4T family that feature split writeback cache memories.

1.3 Directory Structure

The directory tree should be as shown below starting from the project's root directory.

```
├── doc
│   ├── ZAPUG100.pdf
├── hw
│   ├── rtl
│   │   ├── cache
│   │   │   ├── zap_cache_fsm.v
│   │   │   ├── zap_cache_tag_ram.v
```

```
│   │   │   └── zap_cache.v
│   │   ├── cpu
│   │   │   ├── zap_alu_main.v
│   │   │   ├── zap_core.v
│   │   │   ├── zap_cp15_cb.v
│   │   │   ├── zap_decode_main.v
│   │   │   ├── zap_decode.v
│   │   │   ├── zap_fetch_main.v
│   │   │   ├── zap_fifo.v
│   │   │   ├── zap_issue_main.v
│   │   │   ├── zap_memory_main.v
│   │   │   ├── zap_predecode_compress.v
│   │   │   ├── zap_predecode_coproc.v
│   │   │   ├── zap_predecode_main.v
│   │   │   ├── zap_predecode_mem_fsm.v
│   │   │   ├── zap_regf_block_ram.v
│   │   │   ├── zap_regf_bram_wrapper.v
│   │   │   ├── zap_register_file.v
│   │   │   ├── zap_shifter_main.v
│   │   │   ├── zap_shifter_multiply.v
│   │   │   └── zap_shift_shifter.v
│   │   ├── inc
│   │   │   ├── zap_defines.vh
│   │   │   ├── zap_functions.vh
│   │   │   ├── zap_localparams.vh
│   │   │   └── zap_mmu_functions.vh
│   │   ├── lib
│   │   │   ├── mem_inv_block.v
│   │   │   ├── zap_mem_ben_block128.v
│   │   │   ├── zap_ram_simple.v
│   │   │   ├── zap_reset_sync.v
│   │   │   └── zap_sync_fifo.v
│   │   ├── rtl_files.list
│   │   ├── tlb
│   │   │   ├── zap_tlb_check.v
│   │   │   ├── zap_tlb_fsm.v
│   │   │   └── zap_tlb.v
│   │   └── TOP
│   │       └── zap_top.v
│   ├── sim
│   │   ├── command.csh
│   │   ├── post_process.pl
│   │   └── run_sim.pl
│   └── tb
│       ├── bench_files.list
│       └── zap_tb.v
├── LICENSE.md
├── README.md
├── spec
│   ├── ARM-ARM-RevB.pdf
│   ├── armref.pdf
│   └── wbspec_b3.pdf
└── sw
    ├── factorial
```

```
|     ├── factorial.c
|     ├── factorial.ld
|     └── factorial.s
└── tools
      ├── bin2vlog.pl
      └── casm2bin.pl
```

1.4 Core Configuration

The top module may be found in `hw/rtl/TOP/zap_top.v`. The file list for compiling the processor may be found in `hw/rtl/rtl_files.list`. Make sure that the environment variable `ZAP_HOME` is set to point to the root of the project.

Table 3

Parameter	Description	Default Value
CACHE_MMU_ENABLE	0x1 - Include cache/MMU with core. 0x0 – Do not include cache/MMU in core.	1'd1
BP_ENTRIES	Number of branch predictor entries available.	1024
FIFO_DEPTH	Depth of fetch FIFO.	4
DATA_SECTION_TLB_ENTRIES	Section TLB entries for data.	4
DATA_LPAGE_TLB_ENTRIES	Large page TLB entries for data.	8
DATA_SPAGE_TLB_ENTRIES	Small page TLB entries for data.	16
DATA_CACHE_SIZE	Data cache size in bytes.	1024
CODE_SECTION_TLB_ENTRIES	Section TLB entries for code.	4
CODE_LPAGE_TLB_ENTRIES	Large page TLB entries for code.	8
CODE_SPAGE_TLB_ENTRIES	Small page TLB entries for code.	16
CODE_CACHE_SIZE	Code cache size in bytes.	1024

2 Clocks and Resets

The design requires 2 clocks, `i_clk` and `i_clk_multipump`. The `i_clk_multipump` is double the speed of `i_clk` and is solely used by the register file to provide a 2W+4R capability. Most FPGAs offer register files with 2 write ports but have a restriction that one of the write port is dependent on the read address. The multi-pumped clock divides a register file operation into 2 phases (Write Phase and Read Phase) and allowing the `i_clk` domain to see a 2W+4R block RAM device.

Table 4. Clocks

Clock Name	Clock Port	Frequency	Source	Description	Notes
Core Clock	`i_clk`		Independent Clock Source	Times the entire core logic and the Wishbone interface.	1
Register Clock	`i_clk_multipump`	2 x Core Clock	Core Clock	Times part of the register file.	2, 3

Notes:
1. Frequency depends on FPGA and synthesis constraints. A Spartan 6 part reaches 70MHz with cache and MMU.
2. Must be exactly double the frequency of the core clock. Also, the rising edges of the two clocks must be aligned.
3. May be generated using a phase locked loop. Most FPGAs contain PLL blocks.

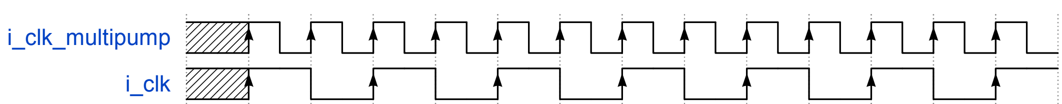

Figure 3. Clock Waveforms

The two clocks may be accurately described using the following SDCs assuming a 100MHz core clock (10ns cycle time, this results in a 200MHz multi-pumped clock)...

```
create_clock \
        -period 10 \
        -waveform {0 5} \
        [get_ports i_clk] \
        -name i_clk

create_clock \
        -period 5  \
        -waveform {0 2.5} \
        [get_ports i_clk_multipump] \
        -name i_clk_multipump
```

The overall block RAM implementation for the register file is shown below. The multi-pumped clock overcomes the limitation of dual write port block RAM functions in most FPGAs that tie one of the write ports to the read port. The multi-pump clock allows a dual write port block RAM to appear as if it had 2 independent write ports and an *independent* read port for devices operating with `i_clk`. The only module operating on the `i_clk_multipump` is the register file in the CPU.

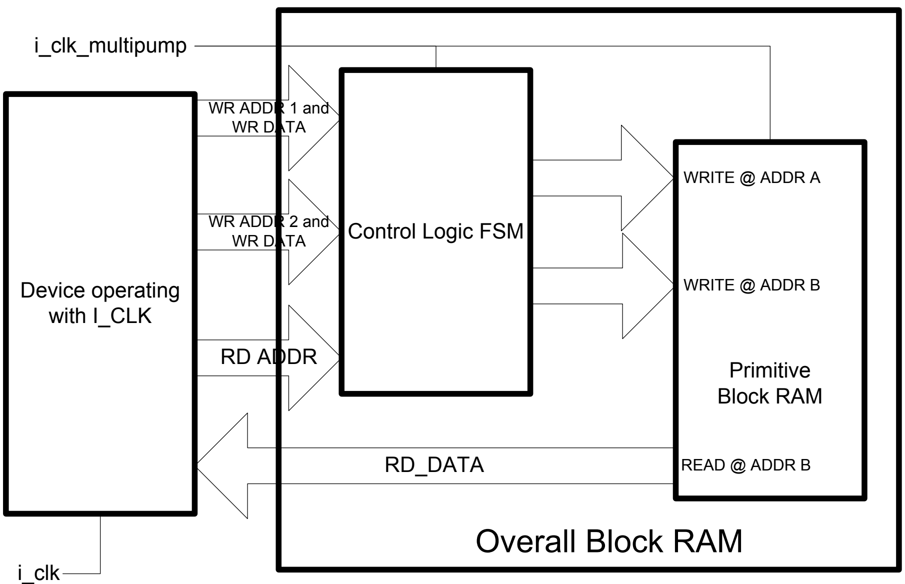

Figure 4. Overall Block RAM for register file

3 IO Ports

Table 5. IO Ports

Port Name	Port Direction	Description
CLOCKS AND RESETS		
`i_clk`	In	The master clock.
`i_clk_multipump`	In	The register file clock. This must be twice as fast as the master clock.
`i_reset`	In	Active high reset. Passes through a dual flop reset synchronizer internally to drive internal synchronous resets.
INTERRUPTS		
`i_irq`	In	Active high IRQ request line. Some non-standard way must be used to inform the external world that the interrupt has been serviced.
`i_fiq`	In	Active high FIQ request line. Some non-standard way must be used to inform the external world that the interrupt has been serviced.
WISHBONE 32-BIT CODE BUS		
`o_instr_wb_stb`	Out	Wishbone strobe signal.
`o_instr_wb_cyc`	Out	Wishbone CYC signal.
`o_instr_wb_adr[31:0]`	Out	Wishbone address.
`o_instr_wb_we`	Out	Wishbone write. Always zero.
`o_instr_wb_sel[3:0]`	Out	Byte lane enable. Always 0b1111.
`o_instr_wb_cti[2:0]`	Out	Wishbone Cycle Type Indicator. Either CLASSIC, INCREMENTAL or END-OF-BURST.
`i_instr_wb_ack`	In	Wishbone acknowledge.
`i_instr_wb_dat[31:0]`	In	Wishbone data.
WISHBONE 32-BIT DATA BUS		
`o_data_wb_stb`	Out	Wishbone strobe signal.
`o_data_wb_cyc`	Out	Wishbone CYC signal.
`o_data_wb_adr[31:0]`	Out	Wishbone address.
`o_data_wb_we`	Out	Wishbone write.
`o_data_wb_sel[3:0]`	Out	Byte lane enable.
`i_data_wb_ack`	In	Wishbone acknowledge.
`i_data_wb_dat[31:0]`	In	Wishbone data.
`o_data_wb_cti[2:0]`	Out	Wishbone Cycle Type Indicator. Either CLASSIC, INCREMENTAL or END-OF-BURST.

NOTE: All Wishbone bursts are LINEAR.

4 CP15 Registers

The ZAP processor (when configured with a cache/MMU unit) supports CP15 access. The register definitions are compatible with the v4 specification. Note that flush and invalidate are used synonymously.

NOTE: CP15 registers are not available if MMU/cache is not installed. Attempting to write to CP15 will trigger an undefined exception in such situations.

NOTE: Some advanced CP15 operations are supported. In particular, invalidating or cleaning an isolated cache line is not supported. In a similar fashion, invalidating an isolated line in a TLB is not supported. Attempting to perform such operations will result in UNPREDICTABLE behavior.

4.1 Register List

Register fields not described in the table below should be treated as UNDEFINED. Software should not rely on specific values for undefined bits.

Table 6. CP15 register description

Index	Name	Description	Notes
0	ID	[23:16] – Always reads 0x01 indicating a v4 implementation.	1
1	CON	[0] – MMU enable. [1] –. RAZ. Processor does not check for address alignment. [2] – Data cache enable. [3] – RAZ. Writeback caches do not need a write buffer. [7:4] – Reads as 4'b1111. Processor only supports Little Endian ordering. [8] – S bit. [9] – R bit. [11] – Read as 1. Always indicates predictable cache strategy. [12] – Instruction cache enable. [13] – RAZ. Processor does not support high vectors.	
2	TRBASE	Holds 16KB aligned base address of L1 table to be used.	
3	DAC	Domain access control register.	
4	--	RESERVED.	
5	FSR	Fault status register. Only data MMU can update this. For debugging purposes, a value can be written into this register.	
6	FAR	Fault address register. Only data MMU can update this. For debugging purposes, a value can be written into this register.	
7	CACHECON	Cache flush/clean control. List of supported operations are shown in the table below. Data written to this register should be zero (SBZ). Writing non-zero data will result in UNPREDICTABLE results. Performing operations other than those listed in the table below will lead to UNPREDICTABLE	2

<table>
<tr><td></td><td></td><td>results.

The table below describes the operations that can be performed using this register.

Table 7. CACHECON Register
<table>
<tr><th>Opcode2</th><th>CRm</th><th>Description</th></tr>
<tr><td>000</td><td>0111</td><td>Flush all caches.</td></tr>
<tr><td>000</td><td>0101</td><td>Flush I cache.</td></tr>
<tr><td>000</td><td>0110</td><td>Flush D cache.</td></tr>
<tr><td>000</td><td>1011</td><td>Clean all caches. Same as clean D cache since I cache is read only and is always clean.</td></tr>
<tr><td>000</td><td>1010</td><td>Clean D cache.</td></tr>
<tr><td>000</td><td>1111</td><td>Clean and flush all caches. Same as clean and flush D cache, flush I cache since I cache is read only and is always clean.</td></tr>
<tr><td>000</td><td>1110</td><td>Clean and flush D cache.</td></tr>
</table></td><td></td></tr>
<tr><td>8</td><td>TLBCON</td><td>TLB flush control. Data written to this register should be zero (SBZ). Writing non-zero data will result in UNPREDICTABLE results. Performing operations other than those listed in the table below will lead to UNPREDICTABLE operation.

Table 8. TLBCON Register
<table>
<tr><th>Opcode2</th><th>CRm</th><th>Description</th></tr>
<tr><td>000</td><td>0111</td><td>Flush both TLBs.</td></tr>
<tr><td>000</td><td>0101</td><td>Flush I TLB.</td></tr>
<tr><td>000</td><td>0110</td><td>Flush D TLB.</td></tr>
</table></td><td>2</td></tr>
</table>

NOTE
1. Read only. Writes have NO EFFECT.
2. Reads are UNPREDICTABLE.