P1 App GFE System Description

1. Capabilities, System and Components

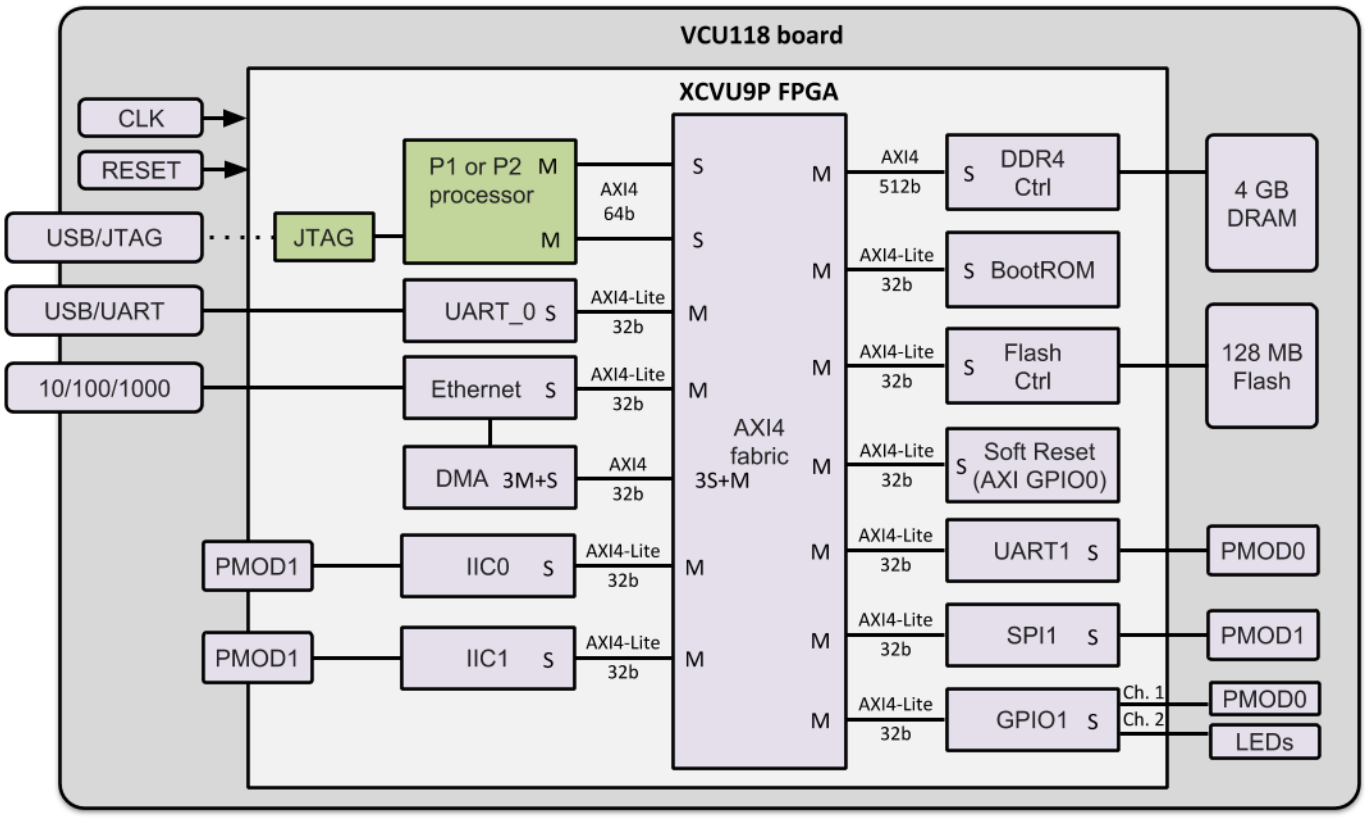
Figure 1 shows a block diagram of the Release X GFE hardware. The system in the XCVU9P FPGA is a physical memory system with P1 processors inserted and a virtual memory system with P2 processors. Programs can run on bare metal or in an operating system (FreeRTOS for P1 and Linux for P2). A host-based gdb debugger connects to the system over the USB/JTAG connector, and a host-based console connects over the USB/UART connector. The Release X system can access 1 GB of the 4GB on-board DRAM through the Xilinx DDR controller and PHY.

Figure 1: GFE P1/P2 VCU118 FPGA board and FPGA device block diagram

The GFE system consists of a Rocket, Piccolo or Flute RISC-V core (see specifications below), a BootROM, Soft Reset and JTAG, UART, Ethernet/DMA, DDR4 and Flash controllers. All components but the P{1,2,3} processors and the JTAG block (green) are Xilinx IP (purple). The BootROM is a read only memory constructed from a BRAM and AXI BRAM controller. The GFE system is the same whether the processor is Rocket, Piccolo or Flute.

The P{1,2,3} processors include Debug Modules (not shown) that connect to a custom JTAG block (https://gitlab-ext.galois.com/ssith/gfe/tree/master/jtag) that uses Xilinx BSCAN primitives under the hood. This enables a tap into the FPGA's JTAG system controller to multiplex the P{1,2,3} JTAG channel onto the external USB/JTAG port (dotted line connection).

The GFE also includes the Security Verification Factory (SVF) block, which transmits Tandem Verification (TV) traces from the processor to the host for verification against a reference model using the *Tandem Verifier*. The TV trace connection between the processor and SVF block is implemented by a point-to-point AXI stream interface.



The components are connected through an AXI4 fabric with X ports. The fabric automatically translates between protocols (AXI4-Lite and AXI4) and channel widths (32, 64, 256, 512 bit).

See section “Xilinx UltraScale+ Implementation” for the Xilinx circuit implementation of this block diagram.

Table 1 lists the Xilinx IP cores used in the GFE system:

Component	Product	Documentation
AXI Interconnect	PG059	https://www.xilinx.com/support/documentation/ip_documentation/axi_interconnect/v2_1/pg059-axi-interconnect.pdf
DDR4 Controller	PG150	https://www.xilinx.com/support/documentation/ip_documentation/ultrascale_memory_ip/v1_4/pg150-ultrascale-memory-ip.pdf
AXI BRAM Controller	PG078	https://www.xilinx.com/support/documentation/ip_documentation/axi_bram_ctrl/v4_0/pg078-axi-bram-ctrl.pdf
AXI UART 16550	PG143	https://www.xilinx.com/support/documentation/ip_documentation/axi_uart16550/v2_0/pg143-axi-uart16550.pdf
AXI 1G/2.5G Ethernet	PG138	https://www.xilinx.com/support/documentation/ip_documentation/axi_ethernet/v7_0/pg138-axi-ethernet.pdf
AXI DMA	PG021	https://www.xilinx.com/support/documentation/ip_documentation/axi_dma/v7_1/pg021_axi_dma.pdf
AXI Quad SPI (Flash ctrl)	PG153	https://www.xilinx.com/support/documentation/ip_documentation/axi_quad_spi/v3_2/pg153-axi-quad-spi.pdf

AXI GPIO Controller	PG144	https://www.xilinx.com/support/documentation/ip_documentation/axi_gpio/v2_0/pg144-axi-gpio.pdf
AXI IIC Bus Interface	PG090	https://www.xilinx.com/support/documentation/ip_documentation/axi_iic/v2_0/pg090-axi-iic.pdf

Table 1: Xilinx IP Cores

2. PROCESSOR SPECIFICATIONS

	P1 Rocket	P1 Piccolo	P2 Rocket	P2 Flute
Instruction set	RV32IMAC		RV64IMAFDC	
Privilege levels	machine, user		machine, user, supervisor (SV39)	
Memory management	No		Yes	
Physical memory protection	No		No	
Compliance tests	https://github.com/riscv/riscv-tests			
	rv32u{i,m,a,c}-p, rv32mi-p		rv64u{i,m,a,f,d,c}-{p,v}, rv64{m,s}i-{p,v}	
Pipeline stages	5	3	5	5
Default GFE clock MHz	50	50	50	50
Max GFE clock MHz	150	83	150	50
Superscalar	No			
Out-of-order	No			
ICache/DCache	4 KB		8 KB	
Multiply-Divide unit	Yes			
Single/double FPU unit	No		Yes	
RISC-V debug module	RISC-V External Debug Support Version 0.13			
	program buffer	system bus	program buffer	system bus
Interrupts (PLIC)	Chapter 8, SiFive U54-MC-RVCoreIP v1p0 – 16 interrupts			
Timer (CLINT)	Chapter 9, SiFive U54-MC-RVCoreIP v1p0			
System bus	64-bit AXI4			
Operating system	FreeRTOS		Linux kernel	
Implementation language	Chisel	BSV	Chisel	BSV

Table 2: Processor Specifications

"Default GFE clock MHz" is the current frequency setting for the ACLK signal (see Figures 2 and 3) that clocks all GFE components except the DDR4 controller. ACLK is generated by the clock generator inside the DDR4 controller (for convenience, not necessity). The ACLK frequency can be modified by adjusting the advanced clocking settings of the DDR controller.

“Max GFE clock MHz” is the maximum clock frequency at which the GFE has been tested with Rocket and Piccolo.

For Rocket, it may be possible to run faster than 150 MHz if the rest of the system can keep up (this hasn’t been tried). For Piccolo, the GFE clock frequency is currently limited by Piccolo, not the rest of the system.

	P3 BOOM	P3 Toooba
Instruction set	RV64IMAFDC	
Privilege levels	machine, user, supervisor (SV39)	
Memory management	Yes	
Physical memory protection	No	
Compliance tests	https://github.com/riscv/riscv-tests	
	rv64u{i,m,a,f,d,c}-{p,v}, rv64{m,s}i-{p,v}	
Pipeline stages	10	8
Default GFE clock MHz	25	25
Max GFE clock MHz	50	25
Superscalar	dual-issue	
Out-of-order	yes	
ICache/DCache	32 KB	16 KB
Multiply-Divide unit	yes	yes
Single/double FPU unit	yes	yes
RISC-V debug module	RISC-V External Debug Support Version 0.13	
	program buffer	system bus
Interrupts (PLIC)	Chapter 8, SiFive U54-MC-RVCoreIP v1p0 16 interrupts	
Timer (CLINT)	Chapter 9, SiFive U54-MC-RVCoreIP v1p0	
System bus	64-bit AXI4	
Operating system	Linux kernel	
Implementation language	Chisel	BSV

3. Xilinx UltraScale+ Design Implementation

The block diagram in Figure 1 is implemented using Vivado’s IP Integrator design flow: https://www.xilinx.com/content/dam/xilinx/support/documentation/sw_manuals/xilinx2018_3/ug995-vivado-ip-subsystems-tutorial.pdf

The IP Integrator enables quick and easy graphical connections of Xilinx IP that is instantiated and configured within an IP Integrator project session. In addition, the P{1,2,3} processors are imported into IP Integrator as 3rd party reusable IP blocks that can be connected as easily as Xilinx IP.

The top-level GFE system is a Vivado block design. It consists of three instances (Figure 2): a Xilinx JTAG module (xilinx_jtag_0) and a P{1,2} processor (mkP2_Core_0) that correspond to blocks in Figure 1; and a subsystem module (gfe_subsystem) that instantiates all the other Xilinx modules.

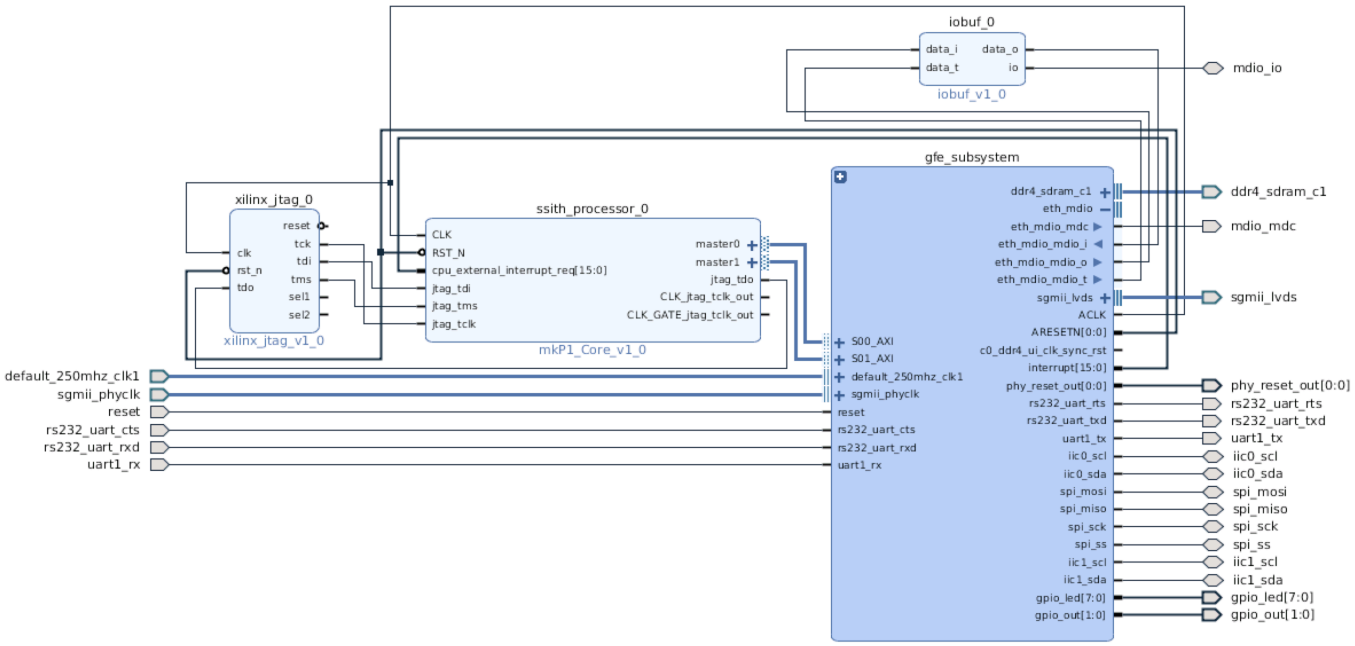


Figure 2: Top-level Vivado block design for P1 App GFE

The *gfe_subsystem* module is shown in Figure 3. It instantiates all the Xilinx cores that are necessary to implement the fabric, UART, DDR4, Ethernet, Flash, BootROM, GPIO, SPI and IIC. in Figure 1. In addition, it instantiates two blocks to implement reset.

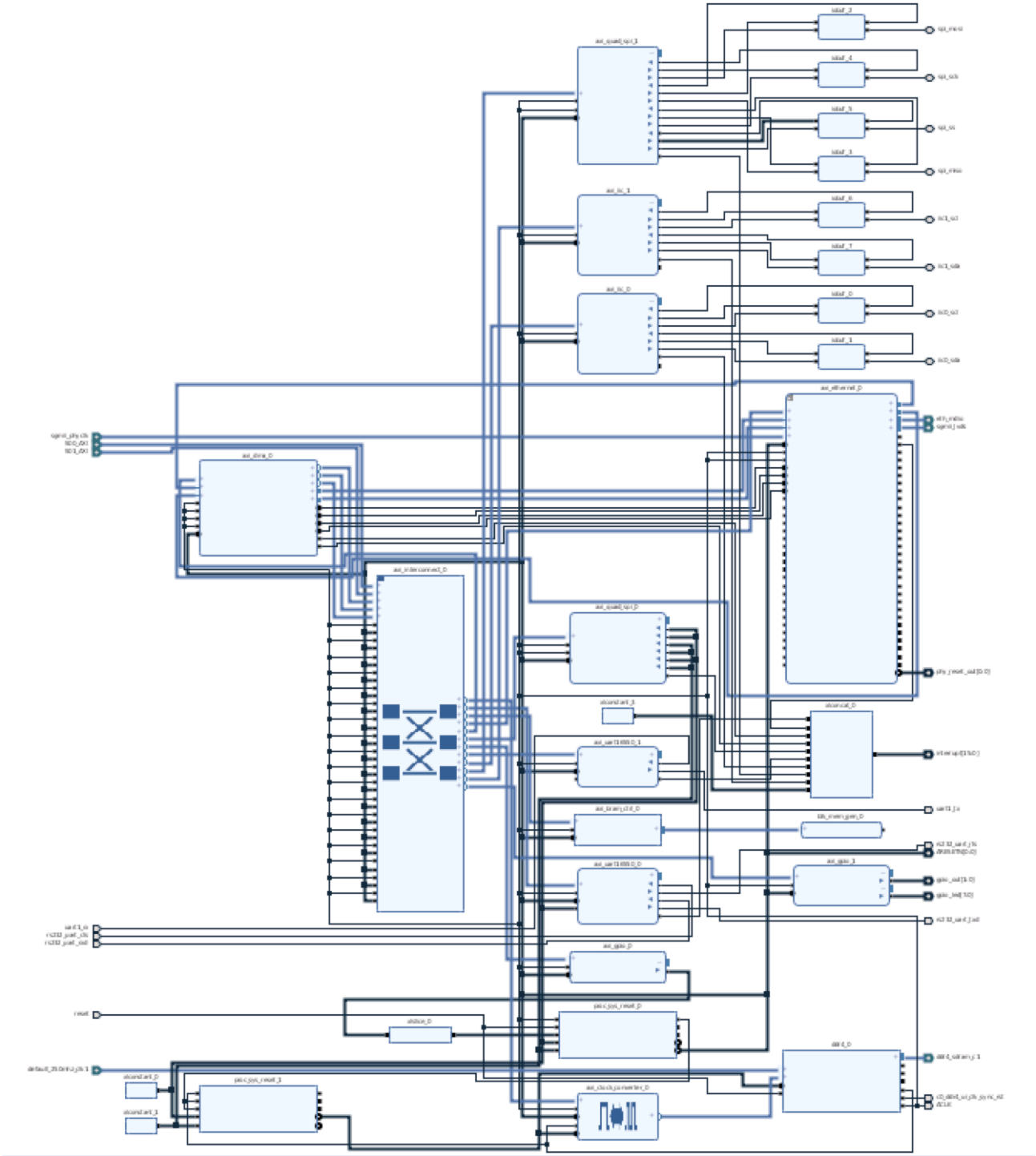


Figure 3: Xilinx block design implementation for *gfe_subsystem*

4. ADDRESS MAPPING

The following table provides the memory map for the memory mapped devices. The CLINT and PLIC are included inside the P{1,2,3} processors and are therefore not mapped into the AXI fabric.

Device	GFE Instance Name	Slave Interface	Base Name	Offset Address	Range	High Address
CLINT	Addressed inside processor			0x0000_0000_1000_0000	64K	0x0000_0000_1000_FFFF
PLIC	Addressed inside processor			0x0000_0000_0C00_0000	4M	0x0000_0000_0C3F_FFFF
DRAM	ddr4_0	C0_DDR4_S_AXI	C0_DDR4_S_AXI_ADDRESS_BLOCK	0x0000_0000_8000_0000	2G	0x0000_0000_FFFF_FFFF
Boot ROM	axi_bram_ctrl_0	S_AXI	Mem0	0x0000_0000_7000_0000	4K	0x0000_0000_7000_0FFF
DMA	axi_dma_0	S_AXI_LITE	Reg	0x0000_0000_6220_0000	64K	0x0000_0000_6220_FFFF
ETHERNET	axi_ethernet_0	s_axi	Reg0	0x0000_0000_6210_0000	256K	0x0000_0000_6213_FFFF
FLASH	axi_quad_spi_0	AXI_FULL	MEM0	0x0000_0000_4000_0000	256M	0x0000_0000_4FFF_FFFF
FLASH	axi_quad_spi_0	AXI_LITE	Reg	0x0000_0000_6240_0000	4K	0x0000_0000_6240_0FFF
Soft Reset	axi_gpio_0	S_AXI	Reg	0x0000_0000_6FFF_0000	64K	0x0000_0000_6FFF_FFFF
UART0	axi_uart16550_0	S_AXI	Reg	0x0000_0000_6230_0000	4K	0x0000_0000_6230_0FFF
UART1	axi_uart16550_1	S_AXI	Reg	0x0000_0000_6234_0000	4K	0x0000_0000_6234_0FFF
I2C0	axi_iic_0	S_AXI	Reg	0x0000_0000_6231_0000	4K	0x0000_0000_6231_0FFF
SPI	axi_quad_spi_1	AXI_LITE	Reg	0x0000_0000_6232_0000	4K	0x0000_0000_6232_0FFF
UART2	axi_uart16550_2	S_AXI	Reg	0x0000_0000_6236_0000	4K	0x0000_0000_6236_0FFF
GPIO1	axi_gpio_1	S_AXI	Reg	0x0000_0000_6233_0000	4K	0x0000_0000_6233_0FFF

Table 3: Address Mapping

The memory map for devices connected to the AXI fabric can be viewed in the “Address Editor” tab next to the “Diagram” tab in the Vivado GFE project window.

Addresses 0x8000_0000 to 0xBFFF_FFFF are configured as uncached in the Bluespec and Chisel processors. This designation enables coherent memory between the DMA engine and RISC-V processors. The Linux device tree (bootrom/devicetree.dts) reserves this region for DMA.

Note that the first UART register is at 0x6230_0000, not 0x6230_1000 as the UART documentation would suggest. This is a result of the small address space allocated to UART in the Xilinx interconnect. The intercon-

nect allocates 0x6230_0000 to 0x6230_0fff to the UART, but the UART registers in the Xilinx documentation are defined at 0x6230_100X. Synthesis causes the upper bits to be ignored within the UART block, so the resulting UART registers are located at 0x6230_000X

Our reference code uses the proper addresses (0x6230_000X). Writing to and reading from 0x6230_100X causes undefined behavior (this is processor dependent). This is also true for the UART_1 with offset address 0x6234_0000.

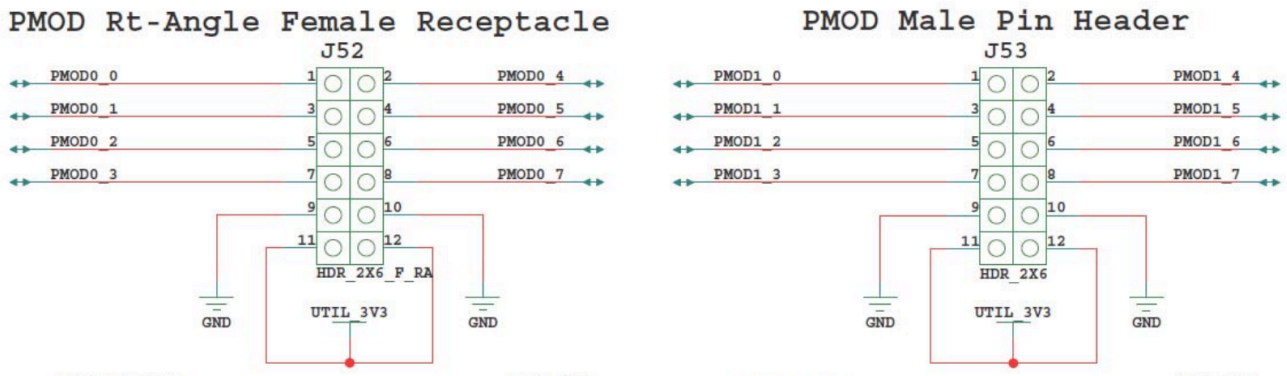
6. Interrupt Mapping

The following table shows the interrupt mapping for cpu_external_interrupt_req[15:0]:

Pin	Connection
0	uart_0
1	interrupt pin on axi_ethernet
2	mm2s_introut on axi_dma
3	s2mm_introut on axi_dma
4	quad_spi_0
5	uart_1
6	iic_0
7	quad_spi_1
8	uart_2
15:9	tied to zero

Table 4: Interrupt Mapping

The PLIC used by the GFE processors indexes interrupts starting from 1. This means that the interrupt signal connected to cpu_external_interrupt_req[0] corresponds to PLIC interrupt 1. This offset is reflected in the Linux device tree provided in bootrom/devicetree.dts.



6. PMOD GPIO HEADER PIN ASSIGNMENTS

Several of the memory-mapped peripherals are connected to the Pmod GPIO headers. The Pmod header pinout from the VCU118 Evaluation Board User Guide is shown below, and the pin assignment is below that.

Figure 4: Pmod GPIO Header Pinout

PMOD0				PMOD1			
0	GPIO_1[0]	**4**	UART_1 TX	**0**	IIC_0 SDA	**4**	SPI_1 SS
1	GPIO_1[1]	**5**	UART_1 RX	**1**	IIC_0 SCL	**5**	SPI_1 MOSI
2	GPIO_1[2]	**6**	GPIO_1[4]	**2**	GPIO_1[6]	**6**	SPI_1 MISO
3	GPIO_1[3]	**7**	GPIO_1[5]	**3**	GPIO_1[7]	**7**	SPI_1 SCK

Table 5: Pmod GPIO Header Pin Assignment

The pin assignment can be changed in xdc/vcu118_soc.xdc. Note that GPIO1 has 2 channels: Channel 1 connects to the Pmod GPIO header outputs pins, while Channel 2 corresponds to the onboard LEDs on the VCU118 board.

7. FREERTOS

The P1 processors boot FreeRTOS 10.2.0.

The Galois FreeRTOS repository is a fork of https://github.com/coldnew/FreeRTOS-mirror which is just a mirror of FreeRTOS SVN, the upstream FreeRTOS. The folder FreeRTOS-mirror/FreeRTOS/Demo/RISC-V_Galois_P1/demo contains applications specific to the P1 and GFE.

See gfe/README.md for instructions on how to run the FreeRTOS demo applications.

8. LINUX KERNEL

The P2 processors boot the Linux 4.20 kernel.

The kernel configuration is in the repository at gfe/bootmem/linux.config. It indicates which Linux drivers are in use.

The “Linux/Busybox” test exercises the basic kernel boot and initialization of devices. See gfe/README.md for instructions on how to boot Linux.

9. *GFE-SVF* CAPABILITIES AND COMPONENTS

The *Security Verification Factory* (*SVF*) provides enhanced RISC-V verification by comparing the architectural state of RISC-V implementations against independently developed high-level reference models of the RISC-V ISA. The current release of the *SVF* uses Bluespec's *Cissr* C model. In addition, in future releases an API will enable use of alternate reference models such as the Haskell models from Galois (*GRIFT*) and Bluespec (*Forvis*).

The *GFE-SVF* is the version of the *SVF* that works with the GFE. It is the *SVF* described in this document, and it requires the *GFE* to run. Currently, the *GFE* runs only on the VCU118, not in simulation.

There is a generic version of the *SVF* that runs in simulation using the free simulator Verilator. The SoC for the generic *SVF* does not use any of the Xilinx IP but it does support the same RISC-V core interfaces as the *GFE*. Therefore, *GFE* P{1,2,3} processors plug directly into the generic *SVF*. This is useful for lightweight testing of RISC-V cores without the long FPGA build times and with the powerful controllability and observability of Verilog simulation.

The *SVF* includes *Tandem Verification* (*TV*), a powerful verification and debug capability. *TV* is particularly good at localizing root causes for failures that occur deep into operating systems and applications because discrepancies are reported immediately, rather than waiting to be detected by changes in system behavior, often many cycles later. In future releases *TV* will also be extended to track system state and support user-defined property checkers, providing the infrastructure for dynamically checking security properties.

Figure 4 is a block diagram of the VCU118 *GFE-SVF*, which consists of the following hardware and software components:

- *SVF* block instantiated in the *GFE* hardware: extracts TV 2.0 traces (see file trace-protocol.pdf) in the GFE repository generated by the P{1,2,3} cores executing RISC-V code and writes the traces to a file on the host.
- PCIe endpoint: provides high-speed one-way transfer of trace files to the host.
- *Tandem Verifier*: checks the TV 2.0 trace against a golden reference model and reports discrepancies.
- *Cissr*: the default (C-based) RISC-V reference model for generating TV-trace comparison results.

Traces from the PCIe link into the host are stored as binary files by the executable *TV-trace_writer* running on the host.

The stored traces contain all the information necessary for the executable *TV-trace_checker_rv{32,64}* (the *Tandem Verifier*) to initialize its copy of system memory and call *Cissr* to execute instructions and report processor state updates. The *Tandem Verifier* then reports any differences between *Cissr* and *GFE* processor program flow and processor state.

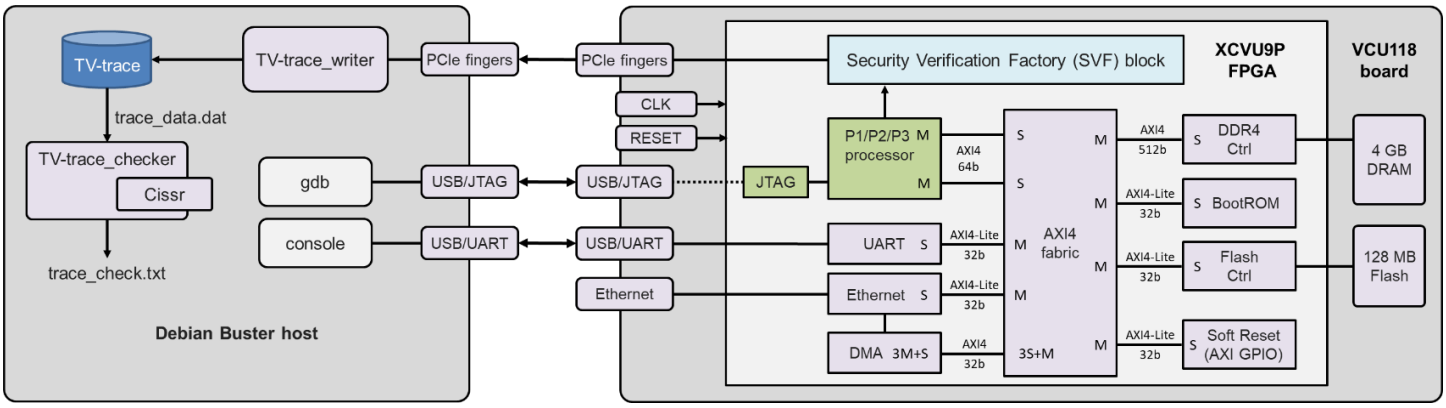


Figure 4: VCU118 GFE operating with *Security Verification Factory* (*SVF*) components

10. *GENERATING AND CHECKING TV TRACE FILES USING THE GFE-SVF*

There are two sets of executable programs in the *TV-hostside* directory (one for RV32 and one for RV64) for creating and checking a TV trace on the host:

- *TV-trace_writer*: stores TV trace transmitted from VCU118 to binary file
- *TV-trace_checker_rv{32,64}*: *Tandem Verifier* checks GFE processor trace files against *Cissr* model

TV-trace_writer requires a license, which is supplied under the Bluespec license.

1. EXAMPLE: GENERATE AND CHECK TV TRACE FOR AN ISA TEST

Step 1: Load the bitstream for chisel_p2 or bluespec_p2 using the instructions in README.md in the GFE repository. Then from your top level GFE directory:

```
$ cd TV-hostside     # directory set up to write trace (includes scemi.params file)

$ ./TV-trace_write   # start the program to store TV-trace from the VCU118
```

Step 2: Load and run the rv64ui-p-add ISA test on the P2 (again, using the instructions in top-level README.md).

Step 3: When then test completes, kill the process running the *TV-trace_writer* program to stop writing trace into the *trace_data.dat* binary file. If the processor had not been halted by the test program (e.g. by reaching a breakpoint), this eventually stalls the CPU, requiring a VCU118 reload before running a another program. Even if this is not necessary, you should always give the "bluenoc reset" command before starting another run.

Step 4: Run the *Tandem Verifier* on the *trace_data.dat* file:

```
$ ./TV-hostside/TV-trace_checker_rv64  trace_data.dat > trace.txt
```

Step 5: Inspect the results in the *trace.txt* output file (snippets in Figure 2):

```
 1  Opened file 'trace_data.dat' for reading trace_data.
 2  Loading BOOT ROM from file 'boot_ROM_RV64.memhex'
 3  ISA = RV64IMAFDCUS
 4  Cissr: v2018-01-31 (RV64)
 5  ------
 6  Cissr: reset
 7  Tandem verifier is: Cissr
 8  Trace configuration: XLEN=64                                                    MLEN=64          FLEN=0
 9  { STATE_INIT[mem_req addr 0x80000000 STORE 32b data 0x4c0006f] }
10  { STATE_INIT[mem_req addr 0x80000004 STORE 32b data 0x34202f73] }
11  { STATE_INIT[mem_req addr 0x80000008 STORE 32b data 0x800f93] }
12  { STATE_INIT[mem_req addr 0x8000000c STORE 32b data 0x3ff0a63] }
13  { STATE_INIT[mem_req addr 0x80000010 STORE 32b data 0x900f93] }
```

```
      #Test program load followed by thousands of 0xaaaaaaaa loads generated by gdb/openocd …
2055  { STATE_INIT[mem_req addr 0x80001ff8 STORE 32b data 0xaaaaaaaa] }
2056  { STATE_INIT[mem_req addr 0x80001ffc STORE 32b data 0xaaaaaaaa] }
2057  { STATE_INIT[dpc 80000000] }
2058  { STATE_INIT[mem_req addr 0x80000040 STORE 32b data 0x0] }
2059  { STATE_INIT[dcsr 4000b053] }
2060  { [pc 8000004c][instr 04c0006f] } inum 1
      #P2 and Cissr start with different PCs(acceptable. After reporting a discrepancy the verifier
      adopts P2's PC and continues, producing a single non-fatal error.
2061  COMPARE ERROR: PC: dut 0x8000004c                                                    tv 0x400
2062  Verifier architecture state ================================
2063    instret 0  PC:00000400
2064    Cur Priv Mode: M
2065    GPRs:
2066     x0/ zero               0 x1/ ra              0 x2/ sp              0 x3/ gp              0
2067     x4/ tp                 0 x5/ t0              0 x6/ t1              0 x7/ t2              0
      #… more lines reporting that the rest of the architectural state matches
2088      mip MIP{ eips:0000 tips:0000 sips:0000}
2089      mcycle 0  minstret 0
      #end of COMPARE ERROR
2090  { [pc 80000050][instr f1402573][a0(x10) 0] } inum 2
2091  { [pc 80000054][instr 00051063] } inum 3
2092  { [pc 80000058][instr 00002297][t0(x5) 80002054] } inum 4
2093  { [pc 8000005c][instr fac28293][t0(x5) 80002000] } inum 5
      #... hundreds of matching instructions …
2591  { [pc 80000004][instr 00000073][priv 3][mstatus a00000000][mcause 8][mepc 80000628][mtval 0] } inum 473
2592  { [pc 80000008][instr 34202f73][t5(x30) 8] } inum 474
2593  { [pc 8000000c][instr 00800f93][t6(x31) 8] } inum 475
2594  { [pc 80000040][instr 03ff0a63] } inum 476
2595  { STATE_INIT[mem_req addr 0x80000040 STORE 32b data 0x0] } # end of trace
```

Figure 2: *Tandem Verifier* output for rv64ui-p-add ISA test trace generated on the VCU118

2. DISABLING TV TRACE

The SVF block disables TV trace transmission if there is no active PCIe link. This also eliminates back pressure to the processors.

Warning:

- If the link is enabled and *TV-trace_writer* is not running on the host, the SVF block will still transmit TV traces and eventually wedge the GFE processor with back pressure.
- If the program is running for a long time, or for example, if the CPU gets stuck in a loop, the file will get very large very quickly.

To avoid these problems, disable TV trace transmission using one of the following approaches:

- Physically disconnect the VCU118 from PCIe. This should only be done if a PCIe cable is used that can be easily disconnected from the host. Removing the VCU118 from a PCIe slot and re-inserting is high-

ly discouraged.

- Reprogram the FPGA but do not execute "bluenoc_hotswap". The PCIe link will not be enabled, and accordingly the SVF block will disable TV trace transmission.

A future version of the SVF will provide a register for programmatically enabling and disabling TV trace generation.

The GFE is built with the SVF block instantiated. This has a small impact on GFE processor LUT count (about 7K), but it could have a larger impact on the FPGA build time. If build time becomes a significant productivity issue, an option can be provided to remove the SVF block from the build.