FRONTGRADE Gaisler

RVMC/NOEL3
A new RISC-V microcontroller CPU core

2025-04-02

Agenda

1 Overview

2 NOEL3 configurations

3 IP cores ecosystem

4 Fault tolerance features

5 Software ecosystem

RVMC – NOEL3

- Developing microcontroller as part of RISC-V Hardware/Software Ecosystem under the name of RVMC, RISC-V Microcontroller
- RVMC will be available as part of ESA IP core portfolio
- Frontgrade Gaisler will make a commercially available version under product name NOEL3

FRONTGRADE Gaisler

NOEL3 – RISC-V processor IP core

Characteristics:

- RISC-V processor core (32-bits)
  - Barrel architecture
- Deterministic, in-order pipeline
- Fault tolerance features
- Small area footprint, suitable also for very small FPGAs
- RISC-V software and tool support, plus our own ecosystem and toolchains

Primary features:

- RV32IMAFCB
  - Suitable for RTOS
  - Configurable number of threads
  - Internal tightly-coupled memories
- AHB and deterministic bus support

Performance

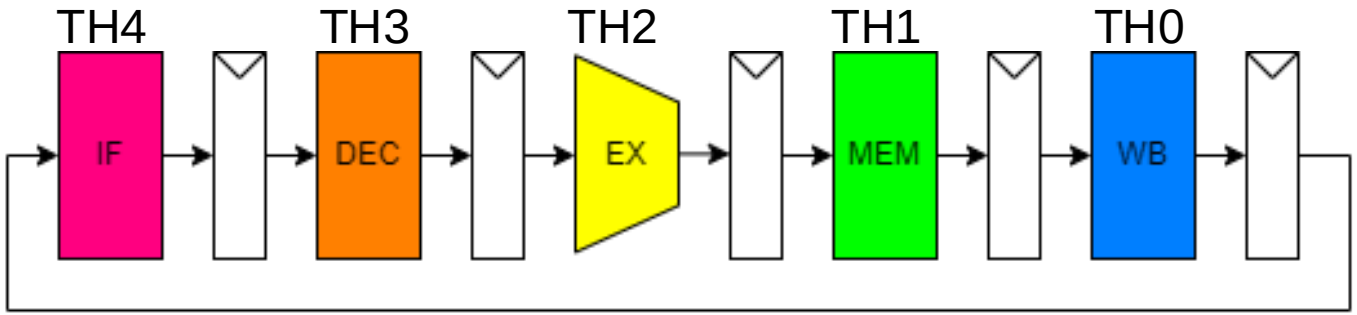
- CoreMark*/MHz (per thread): 0.9**

** Performace may change
* -O3 -finline-functions --param max-inline-insns-auto=20 --param inline-min-speedup=10 -funswitch-loops -funroll-all-loops -fgcse-after-reload -fpredictive-commoning -fipa-cp-clone -falign-jumps=8 -falign-functions=8 --param=l1-cache-line-size=32 --param=l1-cache-size=62 -mcmodel=medany -static -std=gnu99 -ffast-math -ffreestanding -fno-common -fno-builtin-printf -fno-tree-loop-distribute-patterns -march=rv32i_zmmul_zicsr -mabi=ilp32 -nostartfiles -lm -lgcc -T ../riscv64-baremetal/link.ld -I../riscv64-baremetal -I. -DFLAGS_STR=\""-O3 -finline-functions --param max-inline-insns-auto=20 --param inline-min-speedup=10 -funswitch-loops -funroll-all-loops -fgcse-after-reload -fpredictive-commoning -fipa-cp-clone -falign-jumps=8 -falign-functions=8 --param=l1-cache-line-size=32 --param=l1-cache-size=62 -mcmodel=medany -static -std=gnu99 -ffast-math -ffreestanding -fno-common -fno-builtin-printf -fno-tree-loop-distribute-patterns -march=rv32i_zmmul_zicsr -mabi=ilp32 -nostartfiles -lm -lgcc -T ../riscv64-baremetal/link.ld -Wno-implicit-int -Wno-implicit-function-declaration

NOEL3 – Barrel processor

Barrel architecture?

- Fine-grained multithreading processor
- Only one instruction per thread is executed in the pipeline at the same time
- Number of stages = Number of threads



Cycle	Thread retiring instruction
0	Thread 0

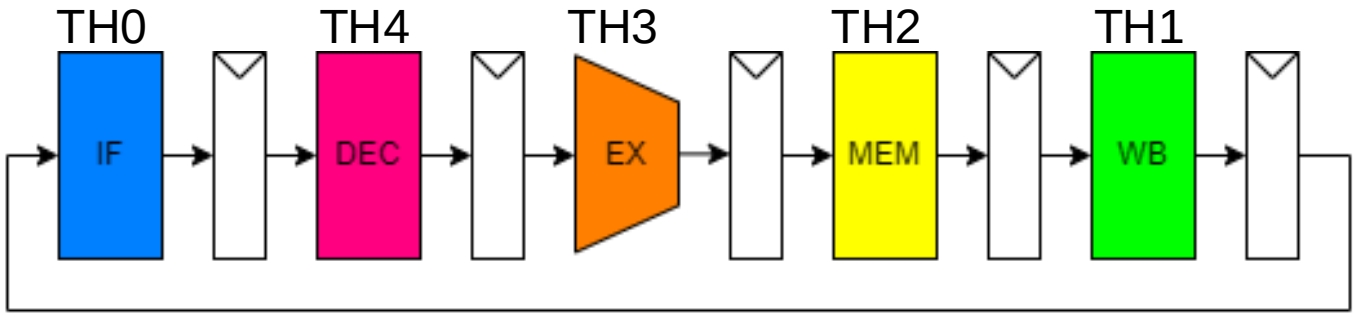
Why barrel architecture?

- Small area
  - No data dependencies/forwarding, no branch prediction, no speculation
- Easier verification
  - Mostly 1-iteration instructions
  - In-order pipeline, instruction retires before next one starts
- High multithreading performance

NOEL3 – Barrel processor

Barrel architecture?

- Fine-grained multithreading processor
- Only one instruction per thread is executed in the pipeline at the same time
- Number of stages = Number of threads



Cycle	Thread retiring instruction
0	Thread 0
1	Thread 1

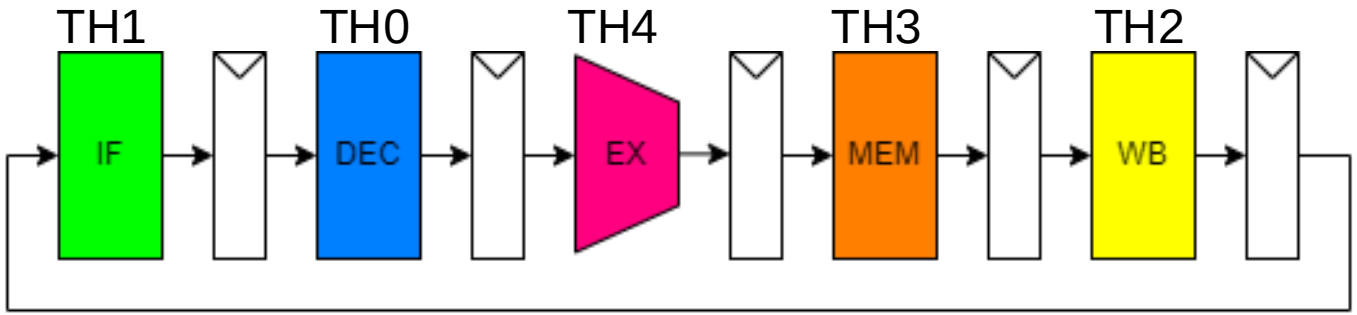
Why barrel architecture?

- Small area
  - No data dependencies/forwarding, no branch prediction, no speculation
- Easier verification
  - Mostly 1-iteration instructions
  - In-order pipeline, instruction retires before next one starts
- High multithreading performance

NOEL3 – Barrel processor

Barrel architecture?

- Fine-grained multithreading processor
- Only one instruction per thread is executed in the pipeline at the same time
- Number of stages = Number of threads



Cycle	Thread retiring instruction
0	Thread 0
1	Thread 1
2	Thread 2

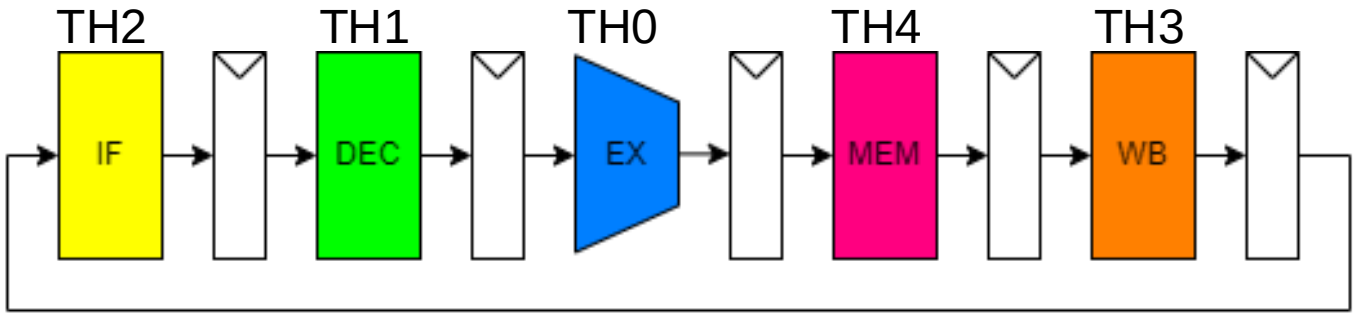
Why barrel architecture?

- Small area
  - No data dependencies/forwarding, no branch prediction, no speculation
- Easier verification
  - Mostly 1-iteration instructions
  - In-order pipeline, instruction retires before next one starts
- High multithreading performance

NOEL3 – Barrel processor

Barrel architecture?

- Fine-grained multithreading processor
- Only one instruction per thread is executed in the pipeline at the same time
- Number of stages = Number of threads



Cycle	Thread retiring instruction
0	Thread 0
1	Thread 1
2	Thread 2
3	Thread 3

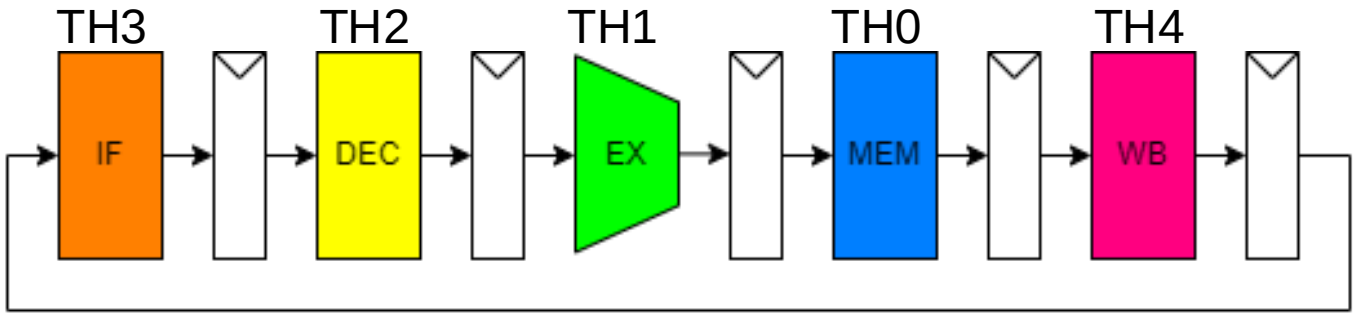
Why barrel architecture?

- Small area
  - No data dependencies/forwarding, no branch prediction, no speculation
- Easier verification
  - Mostly 1-iteration instructions
  - In-order pipeline, instruction retires before next one starts
- High multithreading performance

NOEL3 – Barrel processor

Barrel architecture?

- Fine-grained multithreading processor
- Only one instruction per thread is executed in the pipeline at the same time
- Number of stages = Number of threads



Cycle	Thread retiring instruction
0	Thread 0
1	Thread 1
2	Thread 2
3	Thread 3
4	Thread 4

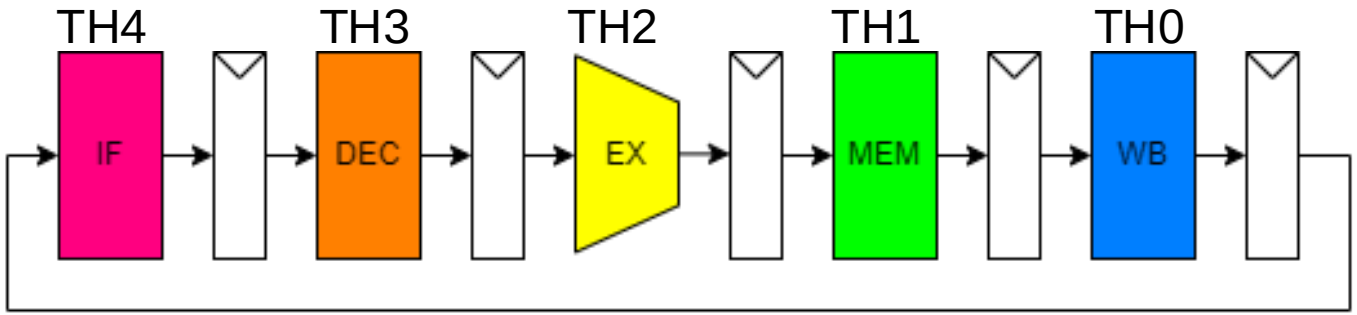
Why barrel architecture?

- Small area
  - No data dependencies/forwarding, no branch prediction, no speculation
- Easier verification
  - Mostly 1-iteration instructions
  - In-order pipeline, instruction retires before next one starts
- High multithreading performance

NOEL3 – Barrel processor

Barrel architecture?

- Fine-grained multithreading processor
- Only one instruction per thread is executed in the pipeline at the same time
- Number of stages = Number of threads



Cycle	Thread retiring instruction
0	Thread 0
1	Thread 1
2	Thread 2
3	Thread 3
4	Thread 4
5	Thread 0

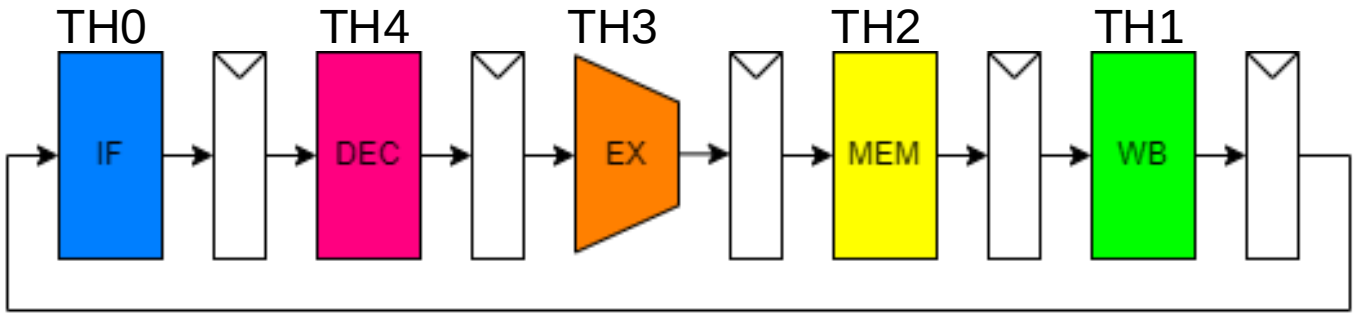
Why barrel architecture?

- Small area
  - No data dependencies/forwarding, no branch prediction, no speculation
- Easier verification
  - Mostly 1-iteration instructions
  - In-order pipeline, instruction retires before next one starts
- High multithreading performance

NOEL3 – Barrel processor

Barrel architecture?

- Fine-grained multithreading processor
- Only one instruction per thread is executed in the pipeline at the same time
- Number of stages = Number of threads



Why barrel architecture?

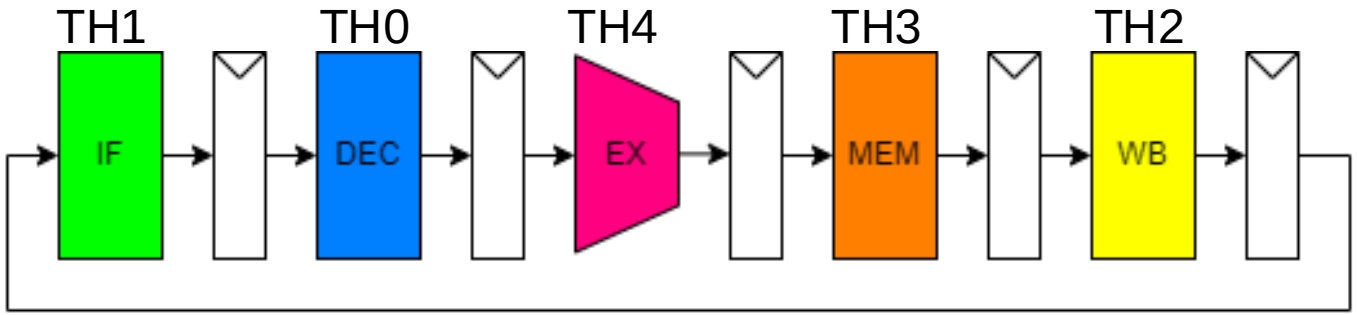
- Small area
  - No data dependencies/forwarding, no branch prediction, no speculation
- Easier verification
  - Mostly 1-iteration instructions
  - In-order pipeline, instruction retires before next one starts
- High multithreading performance

Cycle	Thread retiring instruction
0	Thread 0
1	Thread 1
2	Thread 2
3	Thread 3
4	Thread 4
5	Thread 0
6	Thread 1

NOEL3 – Barrel processor

Barrel architecture?

- Fine-grained multithreading processor
- Only one instruction per thread is executed in the pipeline at the same time
- Number of stages = Number of threads



Why barrel architecture?

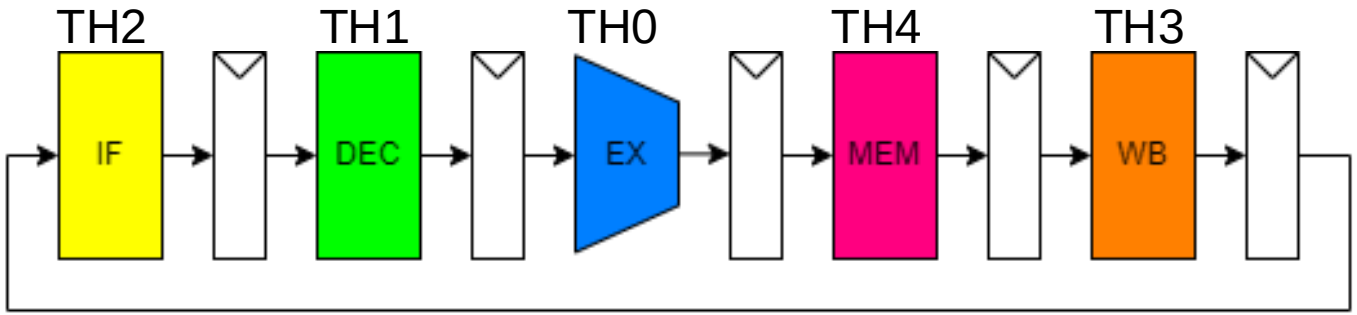
- Small area
  - No data dependencies/forwarding, no branch prediction, no speculation
- Easier verification
  - Mostly 1-iteration instructions
  - In-order pipeline, instruction retires before next one starts
- High multithreading performance

Cycle	Thread retiring instruction
0	Thread 0
1	Thread 1
2	Thread 2
3	Thread 3
4	Thread 4
5	Thread 0
6	Thread 1
7	Thread 2

NOEL3 – Barrel processor

Barrel architecture?

- Fine-grained multithreading processor
- Only one instruction per thread is executed in the pipeline at the same time
- Number of stages = Number of threads



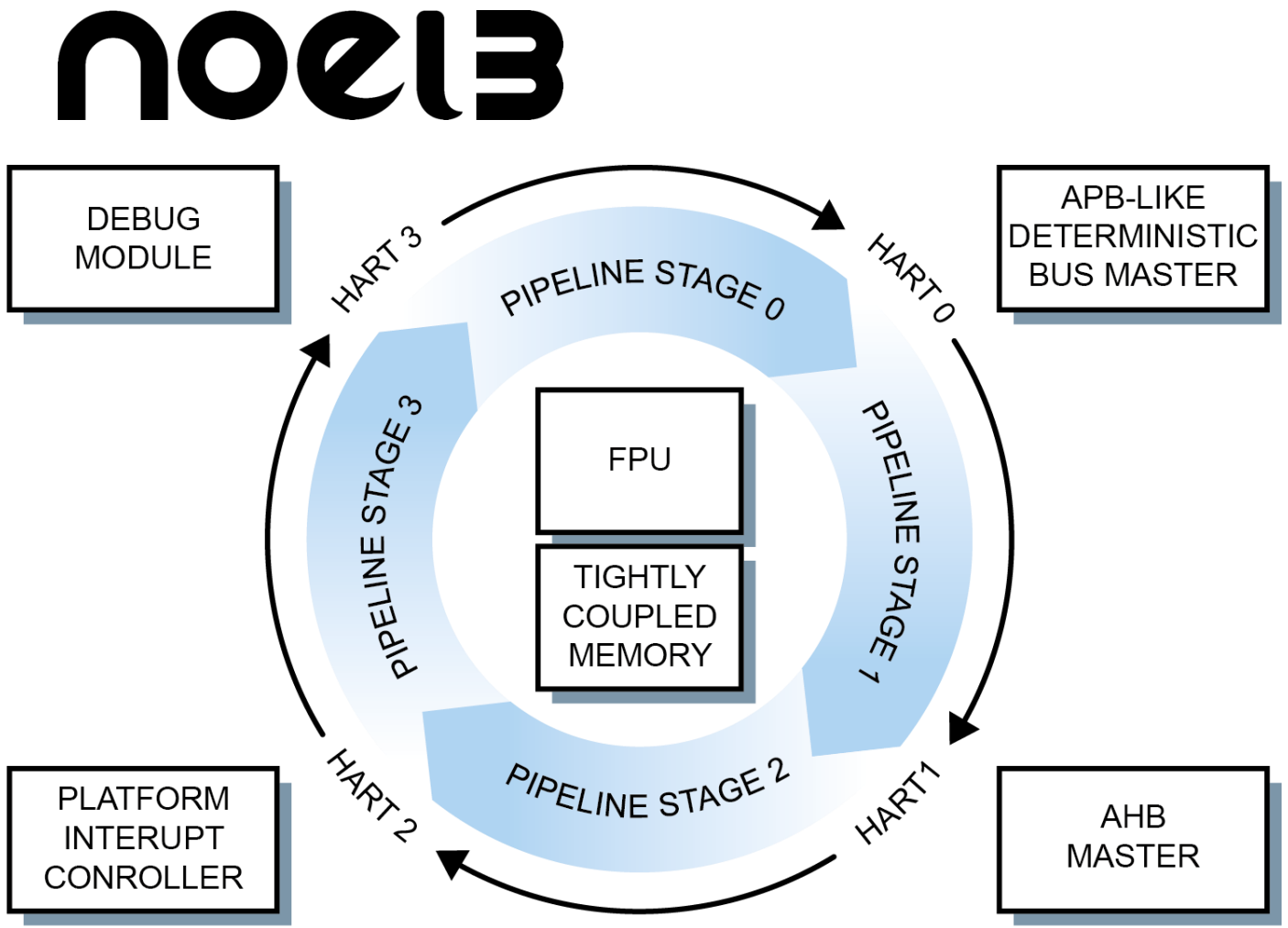
Why barrel architecture?

- Small area
  - No data dependencies/forwarding, no branch prediction, no speculation
- Easier verification
  - Mostly 1-iteration instructions
  - In-order pipeline, instruction retires before next one starts
- High multithreading performance

Cycle	Thread retiring instruction
0	Thread 0
1	Thread 1
2	Thread 2
3	Thread 3
4	Thread 4
5	Thread 0
6	Thread 1
7	Thread 2
8	Thread 3

NOEL3 primary features

NOEL3 – Primary features



- In-order pipeline, barrel architecture
- Deterministic execution
- Fault tolerance
- Configurable number of threads
- Internal tightly-coupled memories
  - Configurable size and architecture
  - Accessible from the outside
- AHB interface
- Deterministic bus interface
- RISC-V standard debug module (GRMON and 3rd party RISC-V debuggers)
- Custom deterministic accelerator interface
- CSR in BRAM
- CV-X-IF (not planned for first release)
- https://www.gaisler.com/products/noel3

NOEL3 – Determinism

Why?

- Real-time systems
- System predictability

What?

- Accurate, predictable execution time regardless of what other threads are doing
- Threads do not affect each other execution-wise
- Access to deterministic bus peripherals

What?

- Accesses to external memory
  - TCMs should suffice
- Interrupts
  - Response time is bounded
  - Possibility to allocate one of the threads to only service interrupts
- AHB accesses

How?

- TCMs
- Deterministic bus interface
- Time slotted accelerator interface

NOEL3 comparison

Architecture	RV32I	RV32IZmmul	RV32IMAFCBZfhZfa
Hardware execution threads	3+1	3+1	6+1
Tightly Coupled Memory	32+ 8 KB	256+256 KB	256+256 KB
Floating point	No	No	Yes
Multiply/Divide	No	Multiply	Yes
Atomics	No	No	Yes
Max frequency	208 MHz	202 MHz	TBD
CoreMark/MHz (for each thread)	0.35	0.9	TBD
LUTs	1.5k	1.8k	TBD
BRAMs	32	134	TBD

Example implementation figures for a Kintex Ultrascale FPGA. The figures have been extracted for the current development stage and are subject to change.

More NOEL3 configurations will be available.

TCM – Tightly Coupled Memory. Directly connected to the CPU for low latency and predictable timing

Compressed – RISC-V 16-bits instructions to reduce the size of compiled programs

Atomics – RISC-V instructions for atomic operations, crucial for multi-threaded and multi-core processing

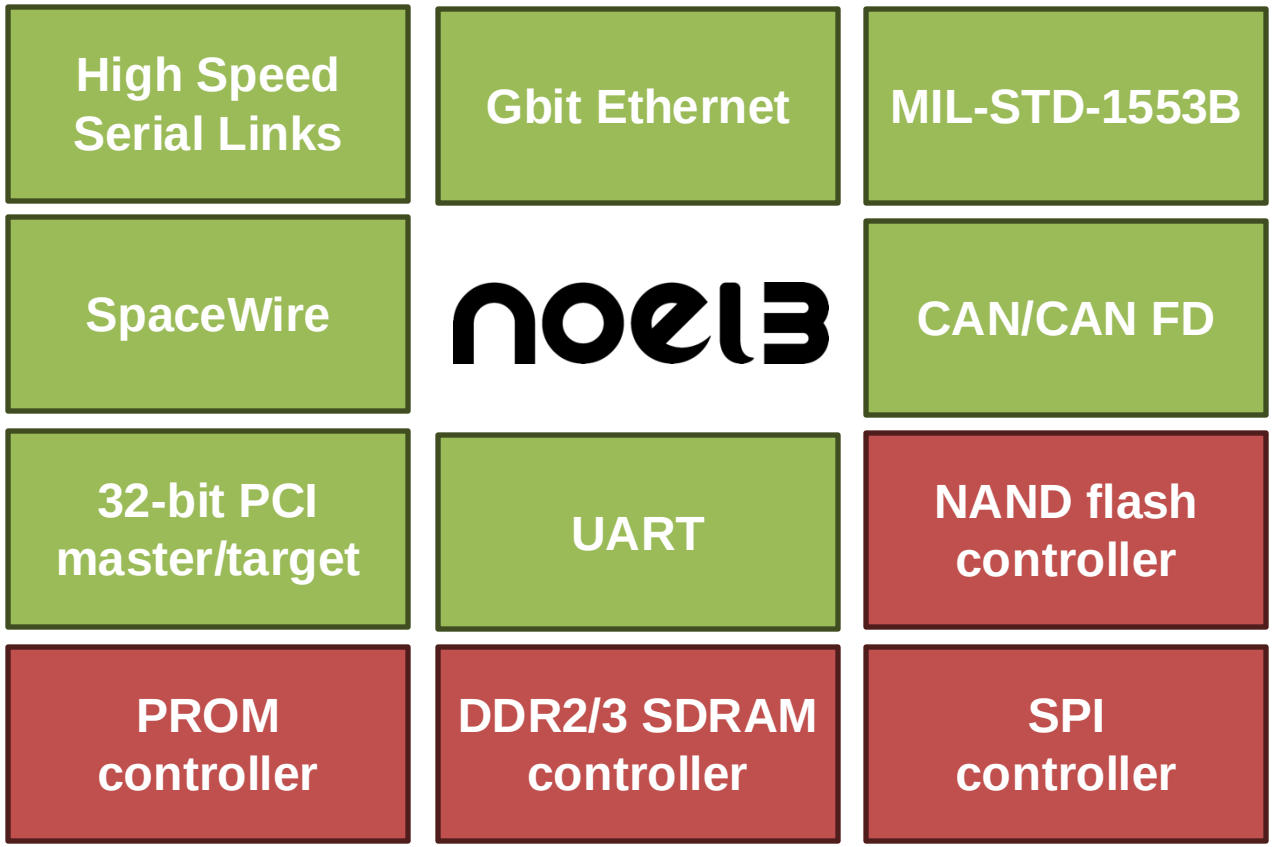
Bit Manipulation – RISC-V instructions to directly manipulate individual bits or groups of bits, improving performance and code density

IP cores ecosystem

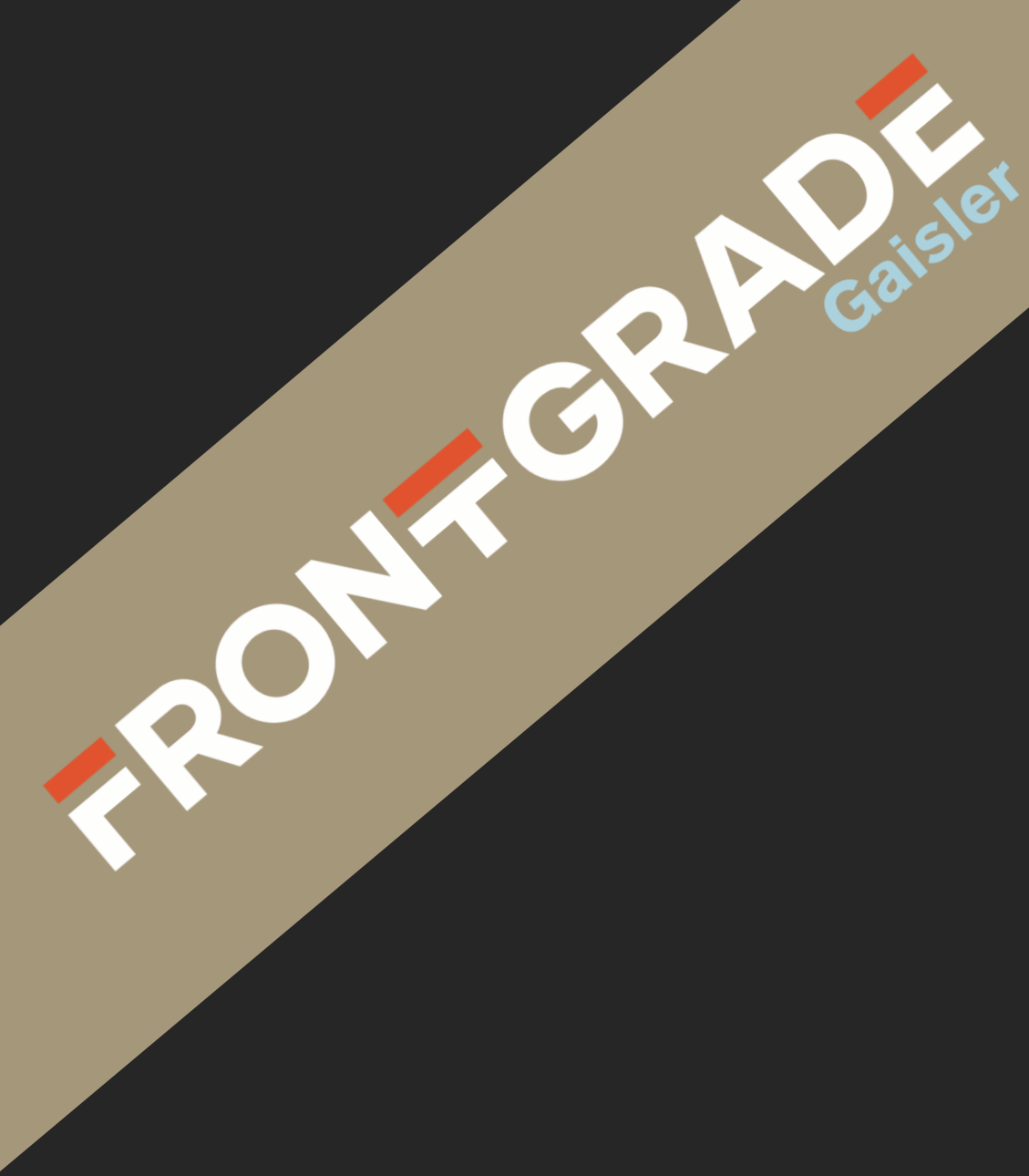
NOEL3 – IP ecosystem

- **Memory controllers (with EDAC):**
  - PROM/IO/SRAM/SDRAM
  - SINGLE/DUAL/QUAD SPI
  - DDR2/DDR3 SDRAM
  - NAND Flash memory controller

- **Peripherals:**
  - UART
  - SpaceWire (controller/router)
  - High Speed Serial Links (SpaceFibre, Wizardlink)
  - 32-bit master/target PCI interface
  - MIL-STD-1553B
  - CAN, CAN FD
  - Gbit Ethernet
  - FPGA supervisor
  - …and many more, see https://www.gaisler.com/grlib-ip-library



GRLIB
SYSTEM-ON-CHIP IP LIBRARY

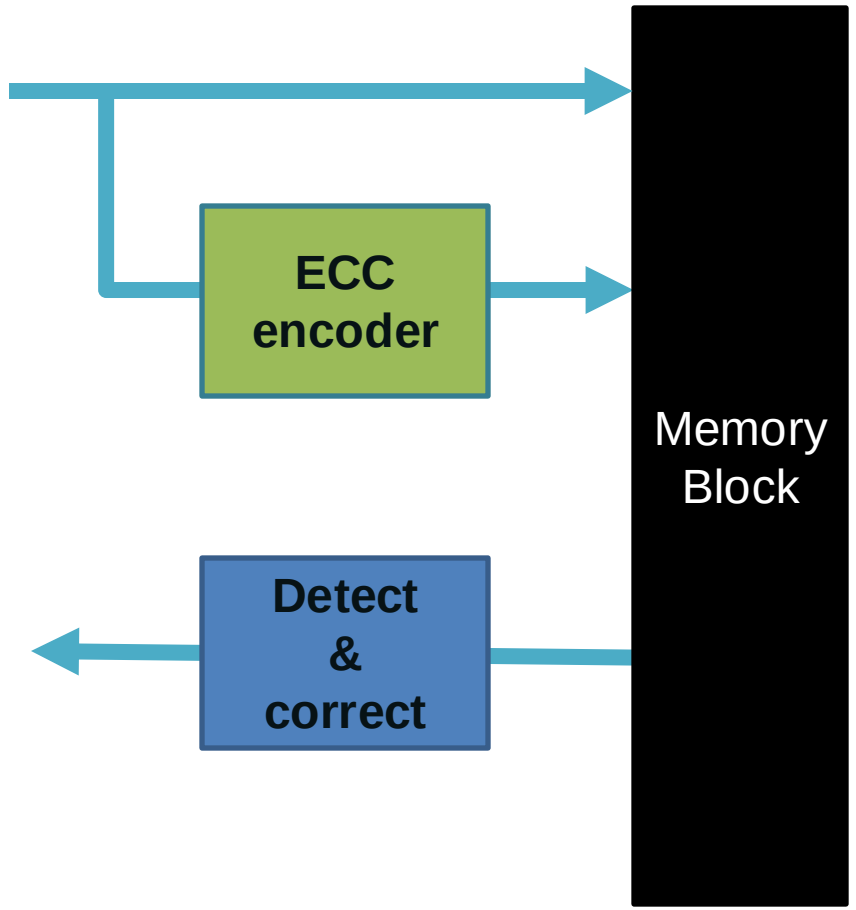


Fault tolerance features

NOEL3 Fault tolerance

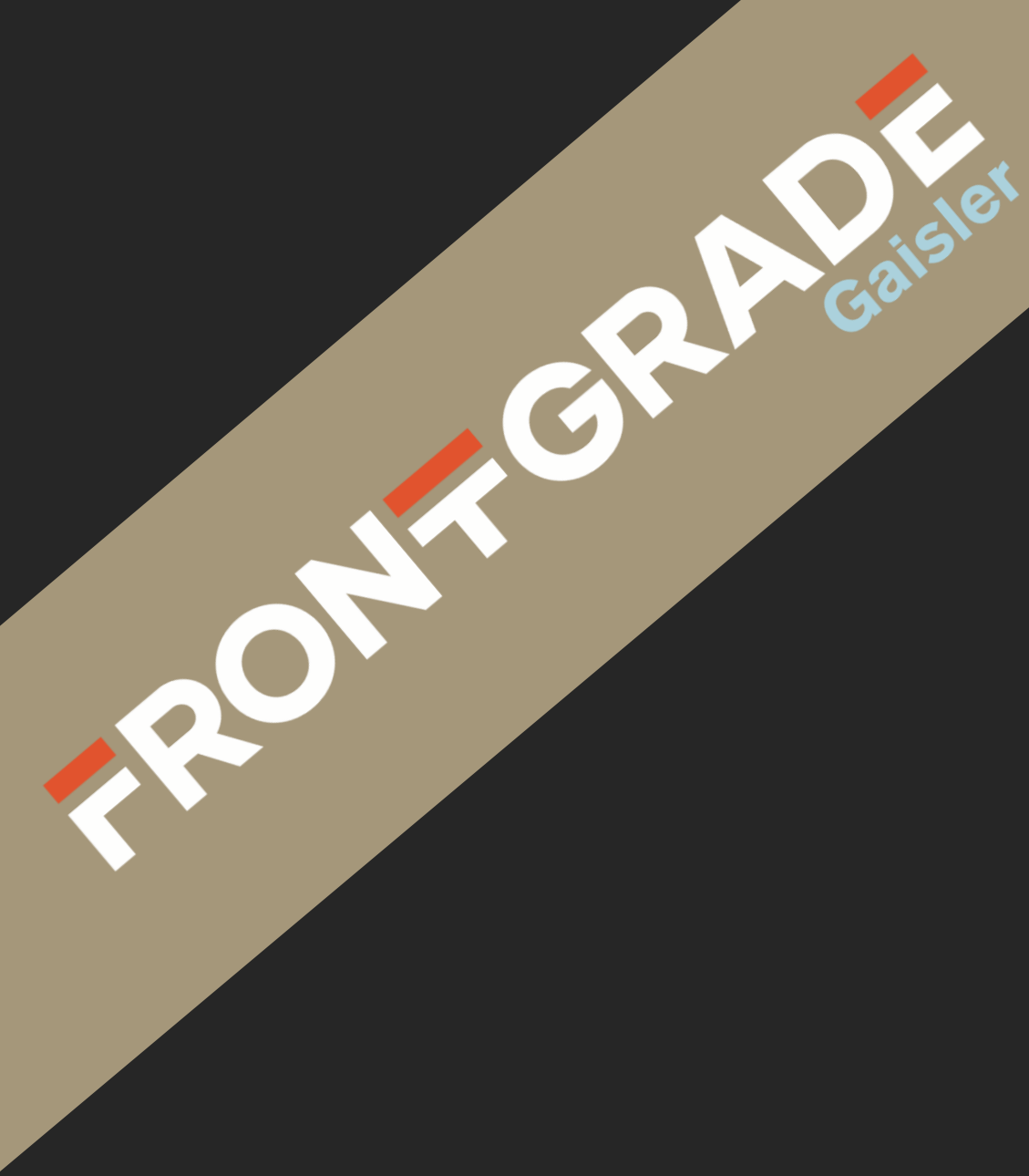
SECDED ECC:

- Protection of memory blocks (tightly coupled memories, register files) using error-correcting codes
- Two means of protection are supported:
  - Native ECC in FPGA fabric (Frontgrade Certus, AMD 7-series/Ultrascale/Versal, Microchip RTG4/Polarfire, NanoXplore)
  - Generic RTL-based ECC (ASIC, other technologies)
- Protection of flip-flops and logic is left to the underlying technology
  - Rad-hard FPGAs and ASIC libraries
  - TMR for SRAM-based FPGAs



Scrubbing:

- TCMs hardware scrubber
- Register file hardware scrubber



Software ecosystem

NOEL3 Software ecosystem

Hardware debugger:

- RISC-V standard specification
- GRMON4

Target software:

- Planned examples for software to take advantage of deterministic multithreading
- Planned RTEMS
- Planned Zephyr ports

NOEL3 Availability

- First implementation of RV32IZmmul running software
- Development continues with planned efforts on verification and feature additions
- Target release of RV32IMAFCBZfhZfa is Q3 2025

- Sign up to our newsletter to receive updates, www.gaisler.com/newsletter