SafeRV : Shakti Functional Safety Architecture

Building Blocks for Safety Critical RISC-V Systems

Paul George & Vipul Vaidya

10/10/2019

SHAKTI Group | CSE Dept | RISE Lab | IIT Madras

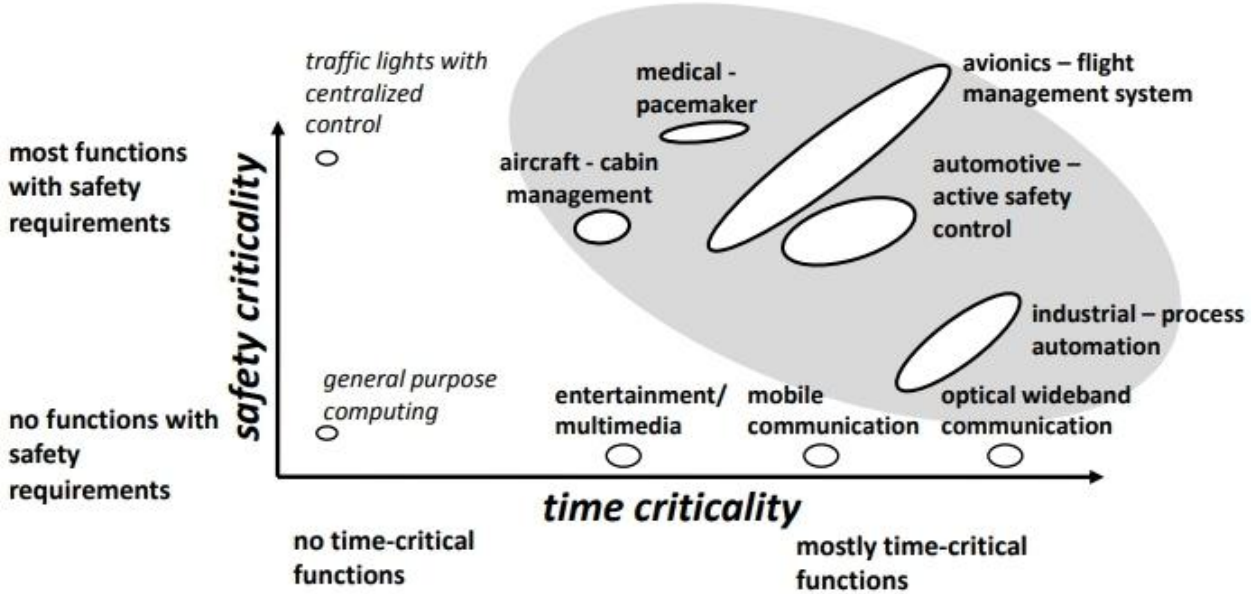
Contributors

- Neel Gala
- Vipul Vaidya
- Rishabh Jain
- Paul George
- Rahul Bodduna
- Anmol Sahoo

Outline

- Motivation
- Shakti - Thales Safe-RV Avionics
- C-Class Core Enhancements
- System Enhancements for Timing Critical/Certainty Applications
- Fault Tolerance Primer
  - Shakti C-Class TLS
- Network-on-Chip for Mixed Critical Systems

Motivation: Mixed Critical x Real time x Fault-Tolerant



- Enabling applications in automotive, industrial control, avionics & space.

[Fig.] S. Tobuschat, "Predictable and Runtime-Adaptable Network-On-Chip for Mixed-Critical Real-time Systems" ,TU Braunschweig, 2019

SHAKTI

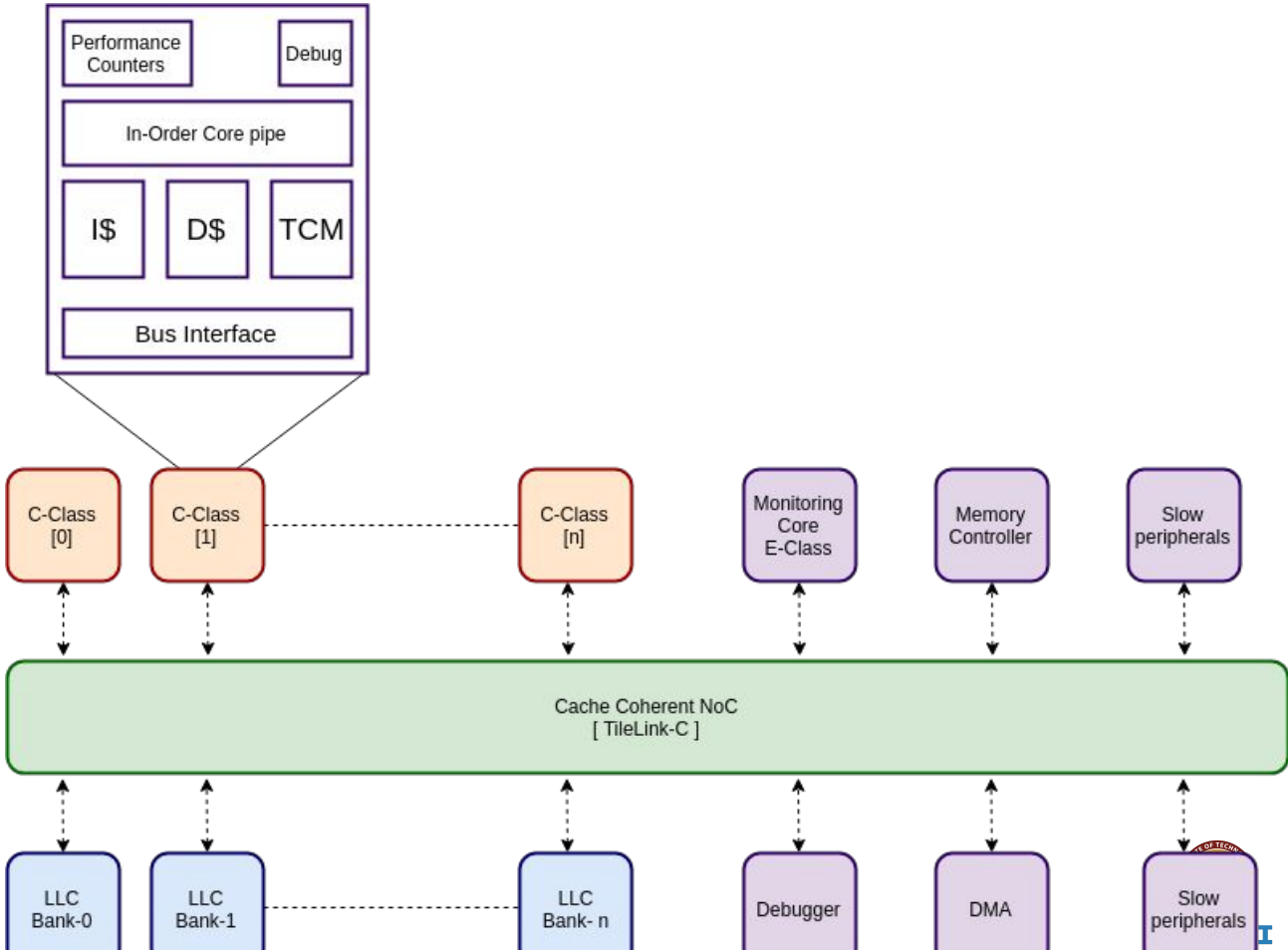
Shakti - Thales Safe RV Mixed Critical SoC

8 x Fault Tolerant C-Class RV64GC - Dedicated L1D/I

1x E-Class RV64IMC - Monitor Core

Criticality Aware - Latency Characterised Interconnect NOC

Banked distributed shared L2 Cache . MSI/MESI - Directory Based Cache Coherence protocol



Time Bound Solutions

Tightly-coupled Memory (TCM)

aka Tightly-integrated Memory (TIM)

- Per Core
- Will be implemented independent of Caches
- Accessed simultaneously with the Caches (separate address range)
- Separate Scratchpad for Instruction and Data

- Cache locking supported in addition
  - Lock up to 2 out of 4 ways
- Cache replacement policy
  - Deterministic! No random! (WCET)
  - Round-Robin chosen for all caches

System Counters

System

- Events / Registers
  - Interconnect read/write filtered by
    - source,
    - target and
    - address range
  - Same for Memory controller
  - Voltage
  - Temperatuure

- System events monitorable from Monitoring Core only
- Per-CPU counters are readable from Monitoring core without impacting that core's execution speed (WIP)

Performance Counters

Per-Core

- Needed for Analysis (WCET), but also for runtime-monitoring
- > 25 events defined per core, e.g. count the number of
  - L1-I or L1-D cache misses and accesses
  - Conditional branches and unconditional jumps
  - Atomic instructions
  - CSR operations
  - Exceptions and Interrupts
  - Accessible without performance costs with the C-Class Daisy chained CSR file.

Core Functional Timing Predictability

- Timing predictability:
  - Shakti Multiplier was originally an „early-out“ multiplier with 1-8 cycles depending on input
  - Replaced by a constant-cycle multiplier

Fault Tolerance

Fault Models

Fault Model = type of error x bit error rate x number of simultaneous errors allowed.

- Type of Errors
  - Single Event Upsets
  - Single Event Transients
  - Permanent Faults - Latchup - Analog sensors required.
- Number of simultaneous errors allowed = 1
- Bit Error Rate ::
  - Depends on Process Technology and operating conditions (derating factors).

Reliability Target.

- MAX :: 100 FIT :: 100 Failures in 10^9 Hours. (Terrestrial - w/o Altitude Derating)

The Solutions will be restricted to micro architectural changes only .No Assumptions assumptions wrt. Cell Level Strategies , Rad Hard Cells , Timing and Clock Tree vulnerability Mitigation with C-Elements , Clock Source/Tree Diversity made.

Architectural Vulnerability Factor - (S. Mukherjee Intel)

Architectural vulnerability analysis is one of the key techniques to identify candidate hardware structures that need protection from soft errors.

The AVF of every hardware structure on a chip is also necessary to compute the full-chip FIT rates.

$$\text{AVF}_{\text{structure}} = \frac{\sum_{i=0}^{N} (\text{bitwise AVF})_i}{N}$$

$$\text{AVF}_{\text{structure}} = \frac{\frac{\sum_{i=0}^{N} \text{ACE cycle}_i}{\text{Total cycles}}}{N} = \frac{\text{Average number of ACE bits in a structure in a cycle}}{\text{Total number of bits in a structure}}$$

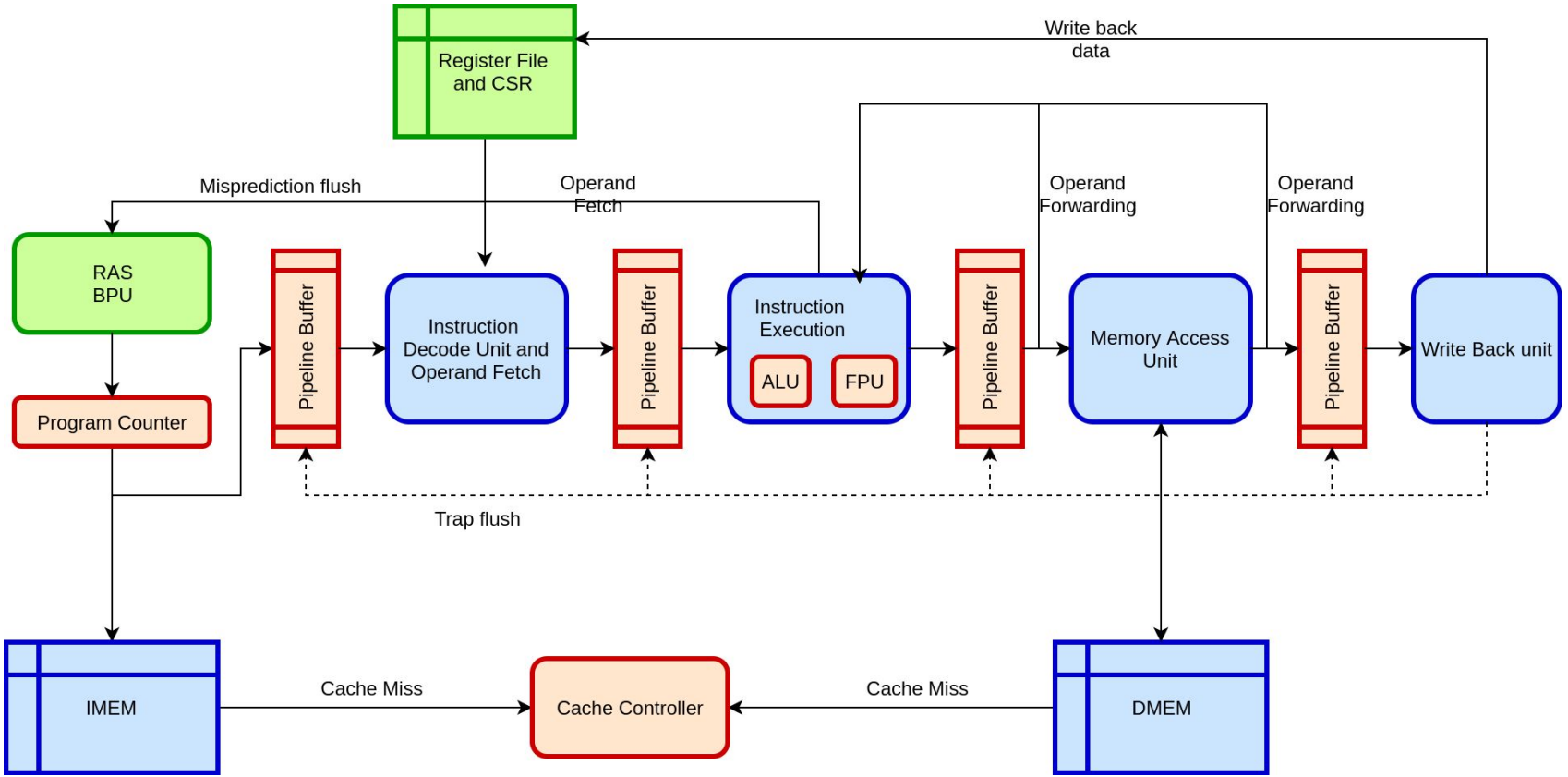
FIT = AVF * Intrinsic Failure Rate of Process Technology * Environmental Deration Factor

AVF ~= Average System Bandwidth used by Architecturally Correct (& Relevant) Execution

Fault Tolerance : TLS Cores

Baseline Solution - Core Complex

Vanilla C-Class Embedded Core



SHAKTI

C-Class TLS :: Initial Draft

Micro Architectural Features :

- Triplicated Core Pipeline
  - Voted Outputs
  - Lazy State Monitoring (minimize Latent Errors)
  - ISR based resynchronization
- Architecturally Enhanced Imem, Dmem , Cache Controller for FT.
  - SECDED ECC on all Storage Elements.
  - Parity on all inflight Data.
  - One Hot Encoded FSMs with Fail Traps and Timeouts.
- Core local Reliability Root Node enabling strategic handling of faults.

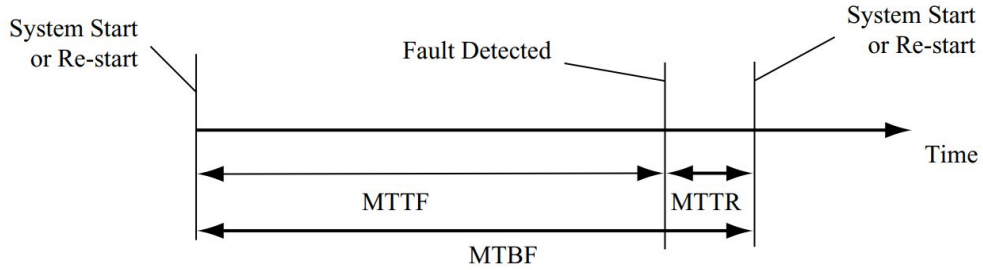
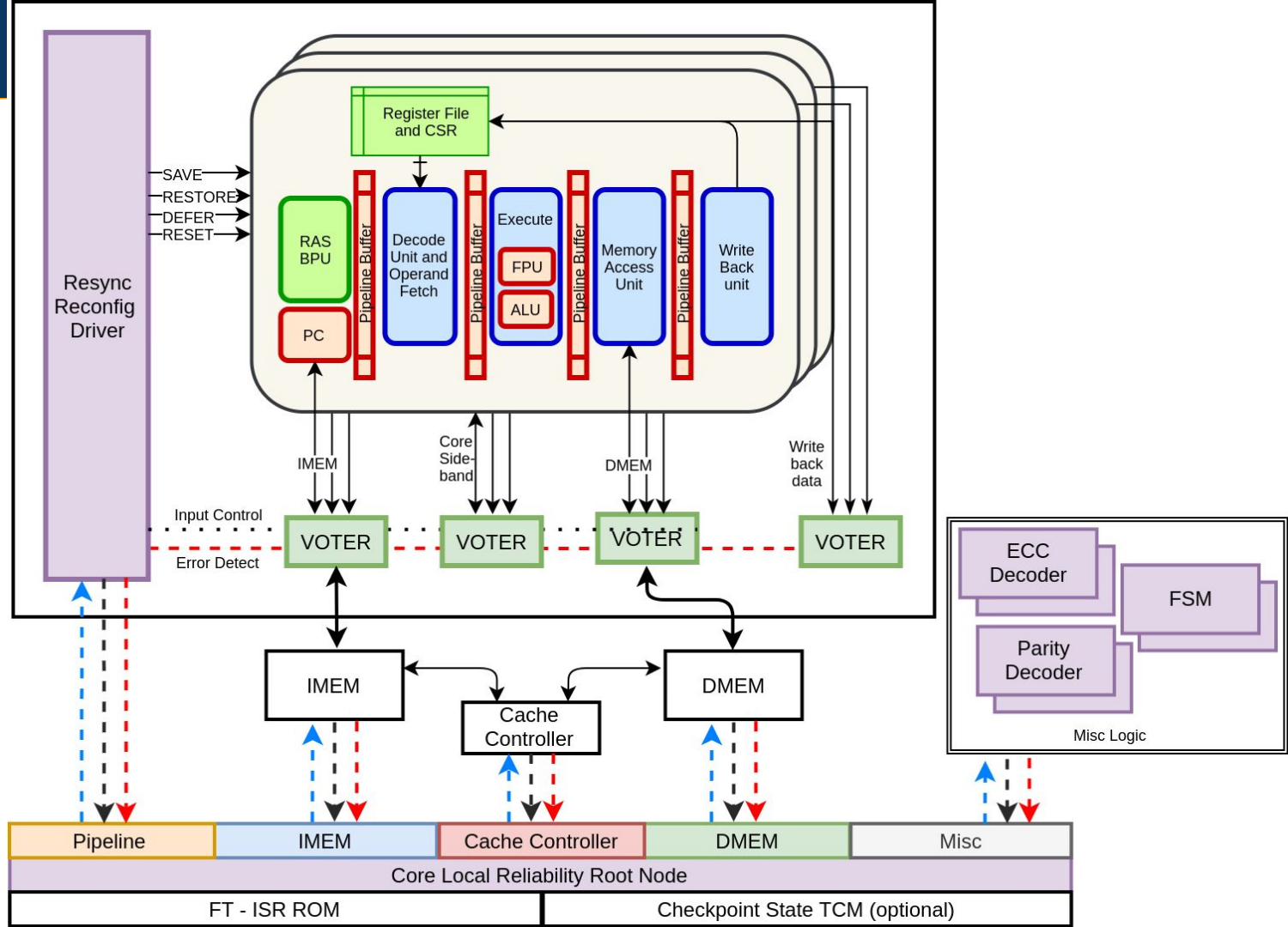


FIGURE 1.5 Relationship between MTTF, MTTR, and MTBF.

SHAKTI

TLS

Triple
Lock-Stepped
C- Class
Core Complex



TLS :: ERROR DETECTION

TLS Core Complex Error Sources

- Pipeline to Imem
- Pipeline to Dmem
- Pipeline to Sideband
- Pipeline Interface Control Signal Mismatch and Timeout
- Pipeline to Lazy Monitoring
- Imem Internal State
- Dmem Internal State
- Cache Controller Internal State
- Core Complex - Interconnect Faults
- Core Complex - State Corruption

Core Local Reliability Root Node

- Allows system software to perform a strategic role in recovery from and diagnosis of hardware errors.
- All Corrected, deferred, and uncorrected errors are logged. Enabled/Non-Maskable errors raise Machine Check exceptions.
  - Corrected :: Self Corrected by ECC
  - Deferred :: Error Detected , Resynchronization pending , Core in degraded resilience
  - Uncorrected :: Non Recoverable State Loss.
- Reconfigurable Event Specific Resynchronization and Scrubbing Routines.
- Configurable Error Thresholding.
- Events banked by source
- Communicates fatal errors to the monitor core.

TLS :: Pipeline Resynchronization

Error Detection at the pipeline output voters requires resynchronization for continued high resilience operation. It is achieved with the following ISRs.

- SAVE :: Creates an image of the Core Architectural State (GPRs, FPRs , CSRs , PC) . Reuses Pipeline Voters by performing store ops to Core Local TCM.
- RESTORE :: Restore Core Architectural State from Core Local TCM.

Additional Behavior :: (Depending on the nature of the error / Configuration)

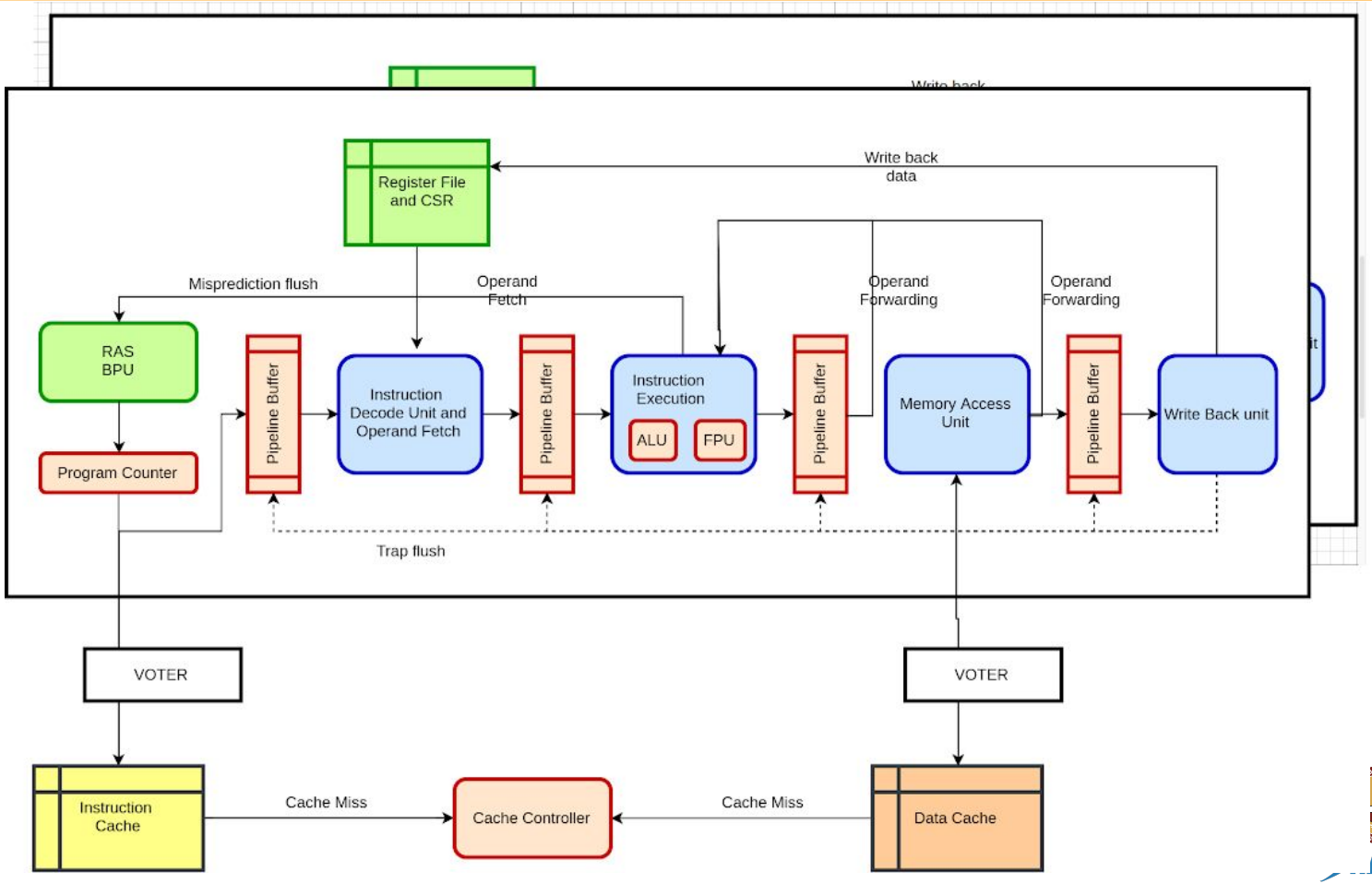
- RESET :: Checkpointed pipelines can optionally be reset to flush errors in unreachable micro-architectural state variables.
- DEFER :: Disable a faulty Pipeline and operate in degraded resiliency mode.

Fault Tolerance Verification Strategy

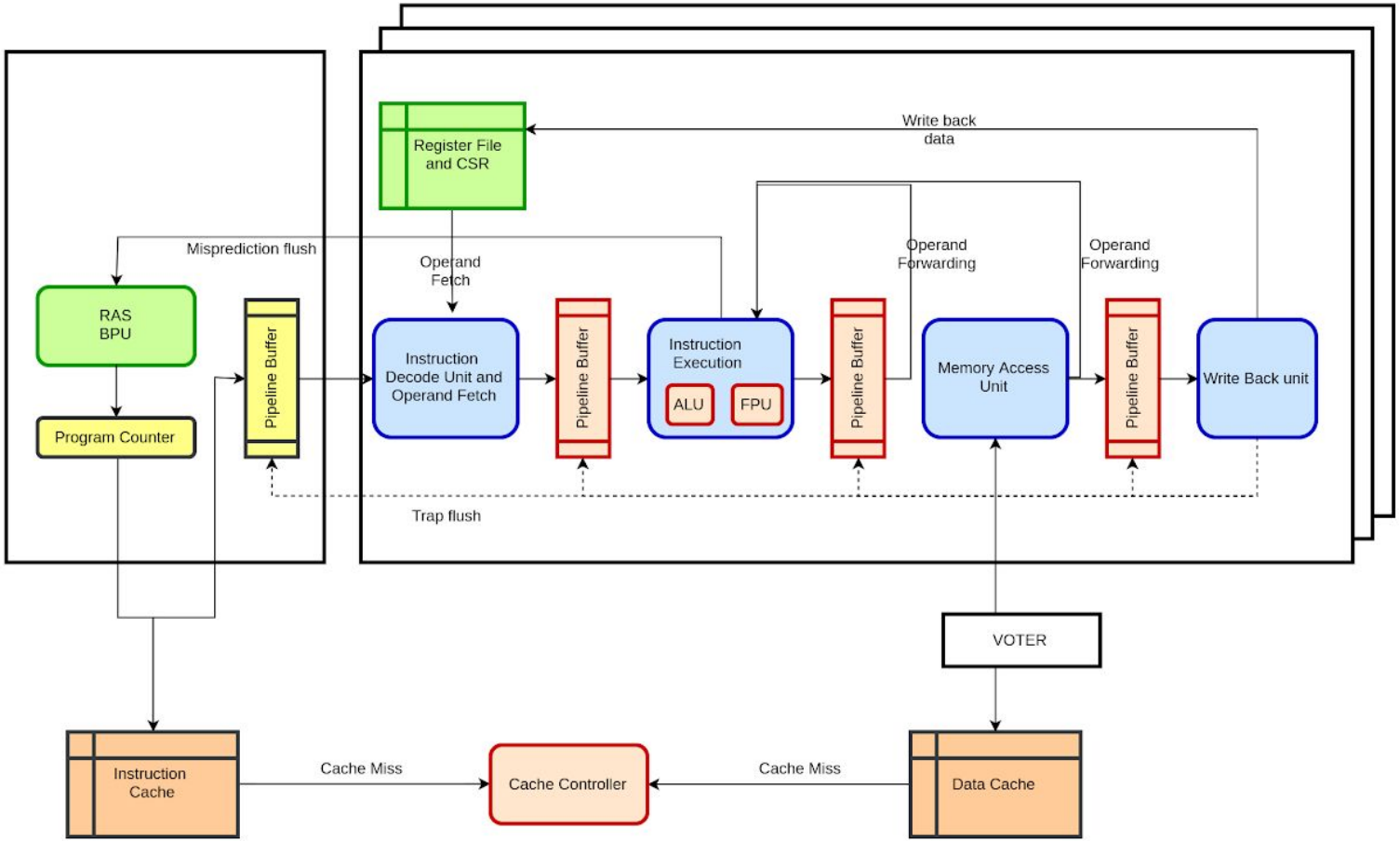
Functional Verification Through Statistical fault injection on Elaborated Verilog Netlist.

Results drive AVF estimation enabling redundancy optimization to meet performance and reliability requirements.

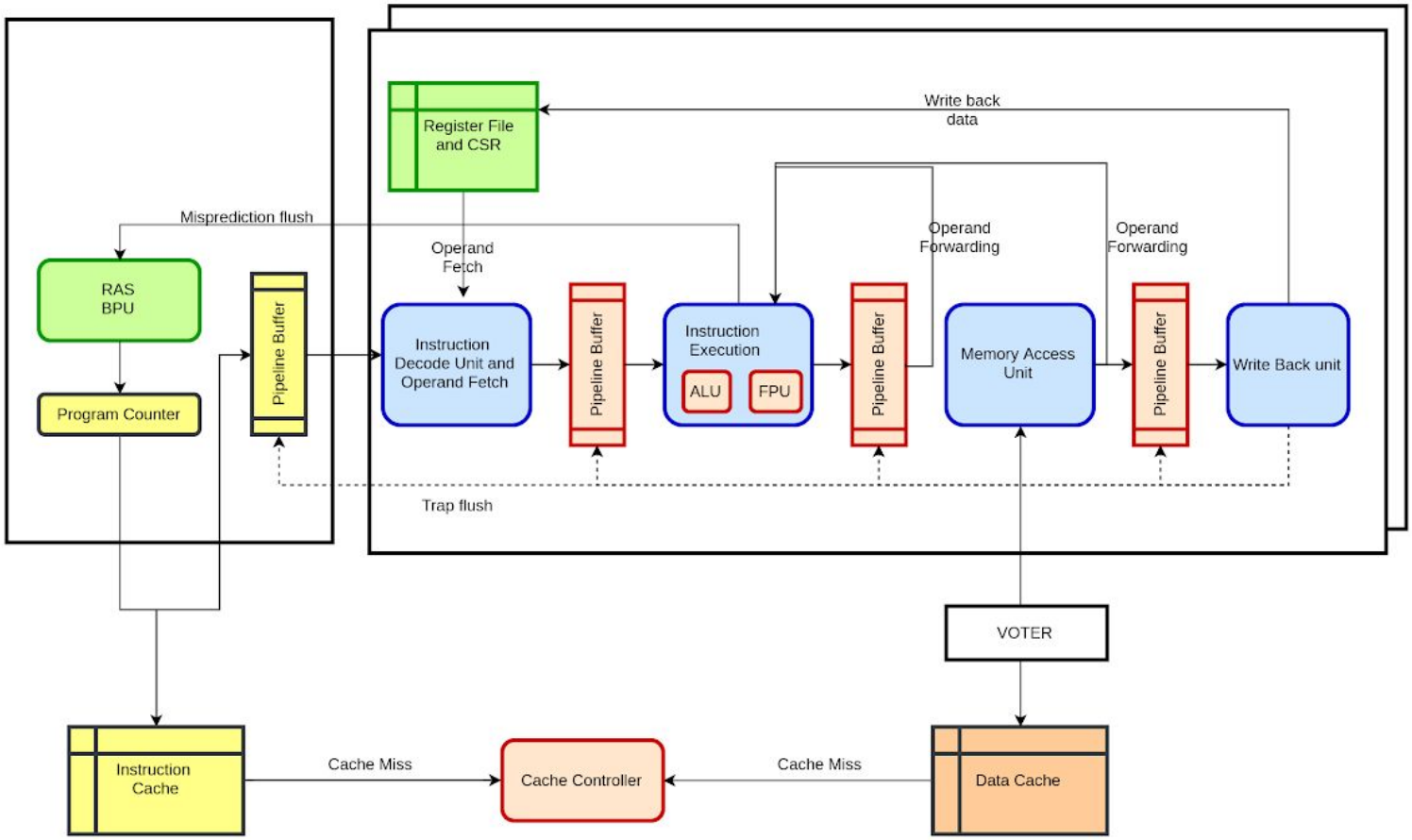
Full DMR - C-Class Pipeline



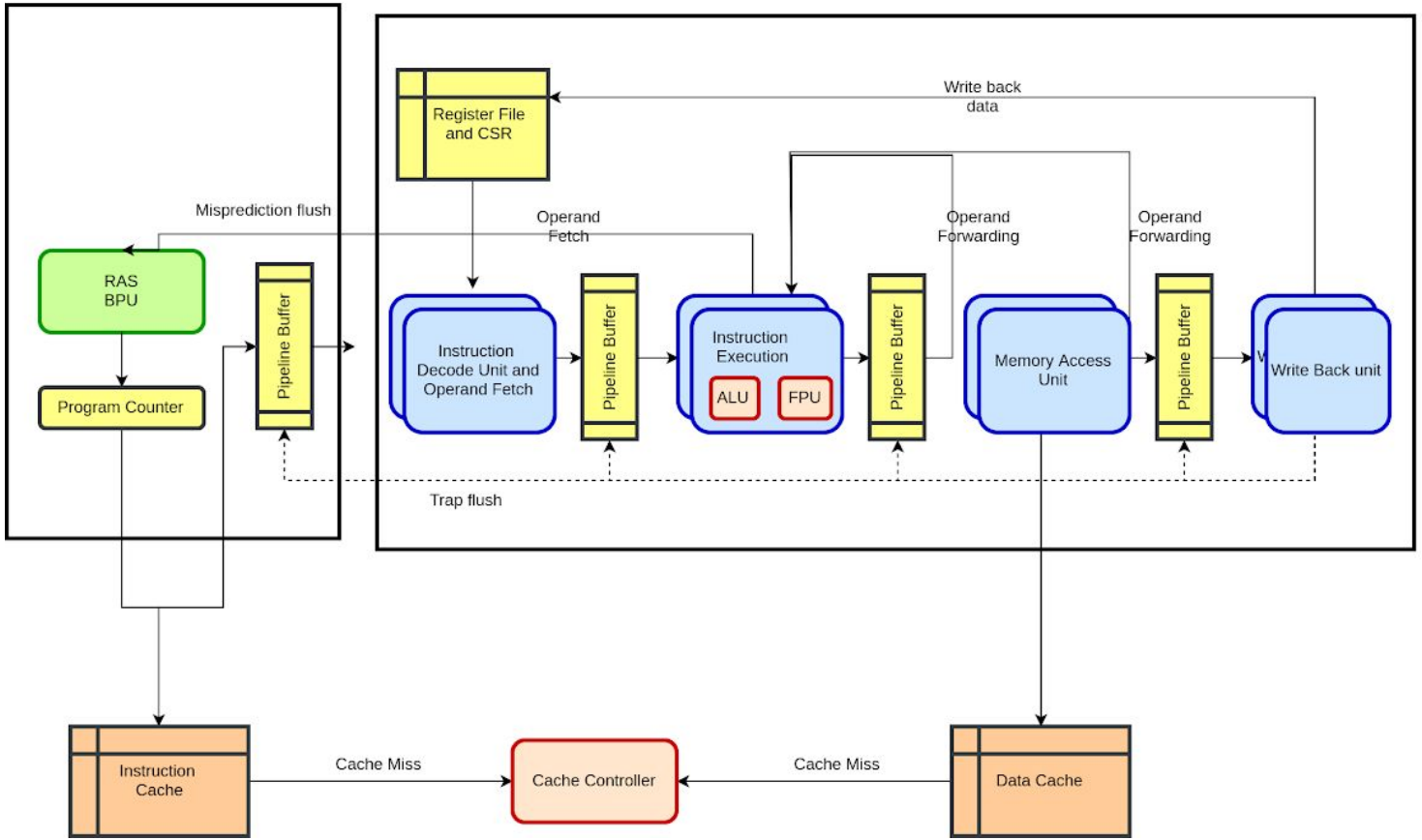
Hardened Frontend - TMR - Exe - Mem -WC Cluster



Hardened Frontend - Double Modular Redundant Exe-Mem-WB Cluster

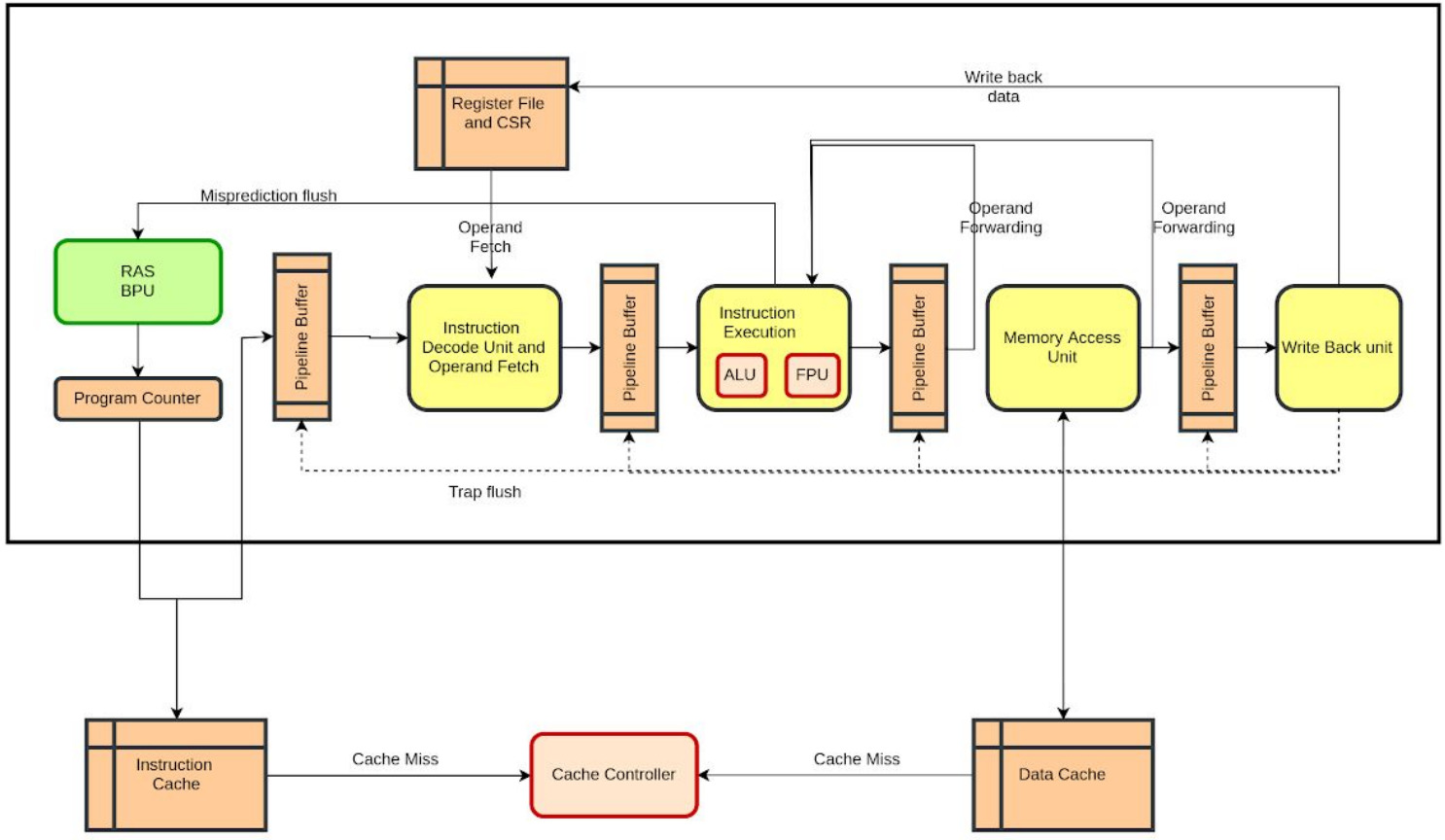


Hardened Frontend - Double Modular Functional Units in Exe - Mem - WB cluster



Fully Hardened C-Class Core Pipeline

Parity Predicted Functional units



References :: Fault Tolerance : TLS Cores

[1] Mukherjee S. - **Architecture Design for Soft Errors**

[2] Sorin D. - **Fault Tolerant Computer Architecture -** Synthesis Lectures on Computer Architecture #5.

[3] Xabier Iturbe et al. - **The Arm Triple Core Lock-Step (TCLS) Processor -** ACM Transactions on Computer Systems, Vol. 36, No. 3, Article 7. Publication date: June 2019.

[4] AMD inc. - **BKDG: Bios Kernel Developer's Guide -** AMD_Family_15h_Models_70h-7Fh_BKDG

[5] Federal Aviation Administration – **Single Event Effects Mitigation Techniques Report** - DOT/FAA/TC-15/62

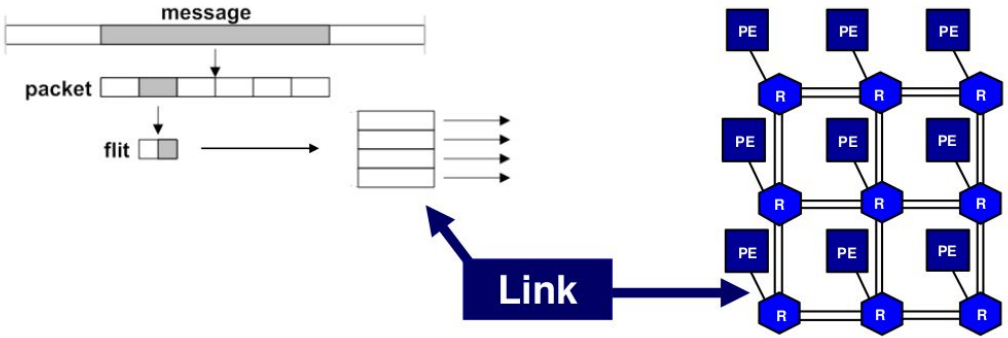
SHAKTI

NoC for Mixed Critical Systems (MCS)

- MCS are characterized by levels of criticality among the applications running on them, Typically :- High Critical & Low critical.
- It is required that the High Critical tasks are provided bounded latencies while the Low critical ones can tolerate some degradation in doing so.
- Apart from the core, the interconnection fabric should also be **deterministic**, meaning it should provide an **upper bound** on the latencies of packets from critical tasks.
- Aim is to extend an NoC with wormhole flow control to incorporate notion of criticality.
- We borrow ideas from the paper **WPMC Flood** [1] :- It models the latency of flows over an enhanced wormhole NoC and performs WCET and schedulability analysis.
- They also perform simulations to correlate their results with the analysis.

[1] L. S. Indrusiak, J. Harbin, and A. Burns, “Average and worst-case latency improvements in mixed-criticality wormhole networks-on-chip,”

Packetization in an NoC



- Network Interface breaks a message into **packets** consisting of fixed sized **flits.**
- Link width determines flit size, packet can contain variable no. of flits :-

  Control message - 1 flit , Data message - 5 flits

- The header flit has information like route info, VC ID.
- Wormhole flow control :The body flits follow the head in pipelined fashion, the header can perform a hop without other flits being present at the input port buffer.

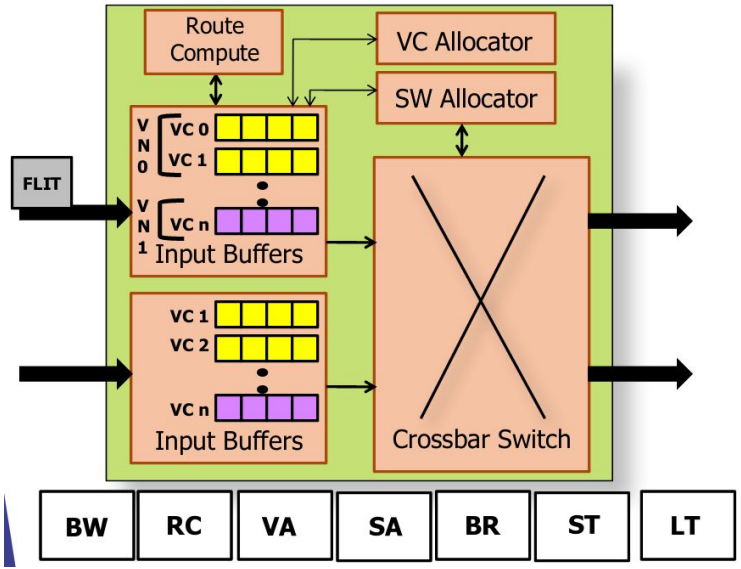
[Fig. : Dr. Leandro Indrusiak, ECRTS 2017]

SHAKTI

Wormhole NoC

- Typically each input port of router will have multiple buffers called **Virtual Channels** to store incoming flits.
- The VCs can be grouped into Virtual Nets (Vnet), each for a particular message class.
- Necessary to avoid protocol level deadlocks.

- A conventional Router has following stages
  - Buffer Write
  - Route Computation
  - VC Allocation
  - Switch Allocation
  - Buffer Read
  - Switch Traversal
  - Link Traversal



[Reference : Dr. Tushar Krisha, 2017 gem5 workshop, ARM Research Summit]

Notion of Criticality in the Router

Adding criticality to the VCs

- We assign criticalities to each VCs, namely HI (high critical) and LO(low critical).
- The Network interface while injecting makes sure that packets occupy VCs corresponding to their criticality.
- Assignation of criticalities can be programmable by using 1-bit registers corresponding to each VC.

An input port having say 4 VCs would look like following :-

VC0 (HI)

VC1 (HI)

VC2 (LO)

VC3 (LO)

Notion of Criticality in the Router

Criticality of a Network Link/Router

- The criticality of link affects how input & output port arbitration is done.
- Each flow is characterized by parameters like **injection rate** , **message size** etc.
- High critical flows have predefined values for these parameters (HI-crit definition) to determine criticality of the link. [2]
- A link would be by default in LO-crit mode.
- If high critical flow is within its HI-crit definition, then link stays in LO-crit mode. If not, the routers and its links switch to HI-crit mode.
- This information for mode change is embedded in the header so that the router is notified & this info propagates as the flit travels.

[2] A. Burns, J. Harbin, L.S. Indrusiak, "A Wormhole NoC Protocol for Mixed Criticality Systems", RTSS

Notion of criticality in Router

Arbitration

- In LO-crit mode, a round-robin policy is used among all VCs irrespective of criticality.
- In HI-crit mode, priority is given to the HI-crit VCs over LO-crit ones, while using RR among the HI-crit VCs.
- Similar policy is used at outport arbitration.
- If HI-crit packets are blocked due to unavailability of buffers at downstream router, LO-crit packets would make progress.

Credit Management

- Separate credit management for HI-crit & LO-crit VCs.
- This ensures HI-crit packets flow only through HI-crit VCs.

NoC Modeling - Gem5

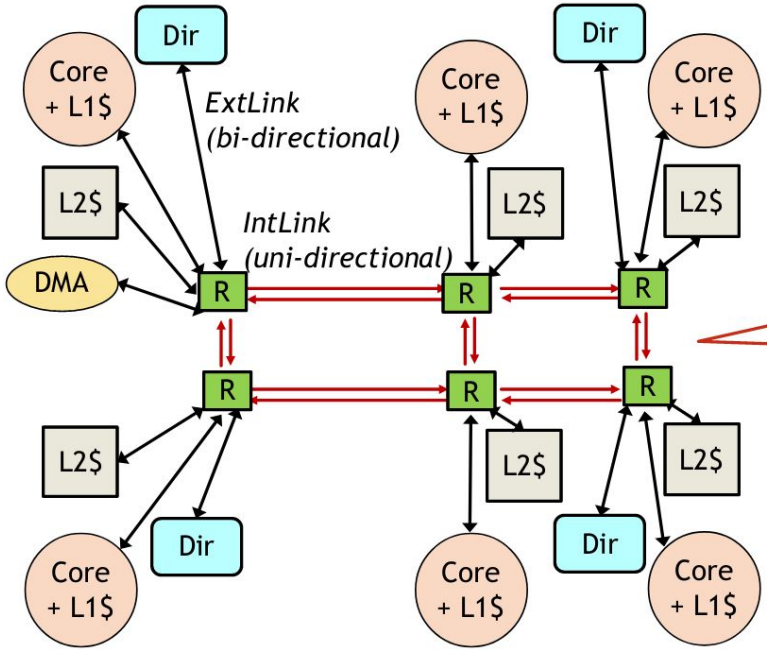
Garnet2.0

- Detailed interconnection model in Gem5 for simulating, modelling and exploring Network-On-Chips.
- Provides a cycle-accurate router and link model with parameterizable latency.
- Able to model various topologies and routing algorithms.
- Parameterizable over -
  - Number of Virtual Channels
  - Depth of Virtual Channel buffers
  - Link width etc.
  - Injection rate etc.

et al. Agarwal, 2009: "GARNET: A detailed on-chip network model inside a full-system simulator"

Garnet2.0

- The garnet system model comprises of -
  - Routers
  - Data & Credit Links
  - Directory Controller
  - Core + Private L1 Cache
  - Distributed Shared L2 Cache
  - Other controllers (Eg: DMA)
- Synthetic Traffic Generator :-
  - Uniform Random
  - Other Synthetic Patterns



System Overview Garnet

[Reference : Dr. Tushar Krisha, 2017 gem5 workshop, ARM Research Summit]

Observation and Results:

- Average latency and maximum latency comparison:
- VNET2 is a message class, where packet size is 5 flit (data message).
- Criticality assignment would be as following :-

An input port having say 4 VCs (VNET2)

VC8 (HI)

VC9 (HI)

VC10 (LO)

VC11 (LO)

	baseline(VNET2)	Scheme(VC 8-9) High Critical	Scheme(VC 10-11) Low Critical
Average latency	33.90	20.19	36.19
Maximum latency	366	81	416

Garnet – Modelling a NoC

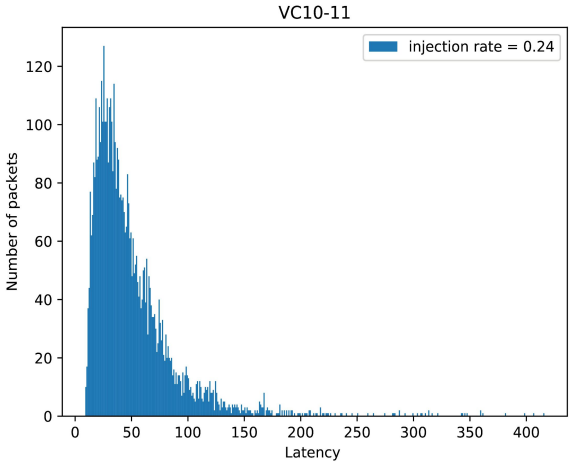
Network Configuration :

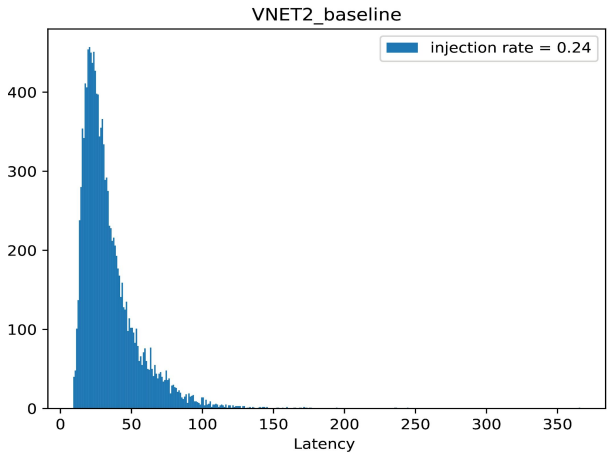
- 1-cycle Router
- Topology : 4*4 Mesh
- Routing : XY (DOR)- Deterministic in nature.
- Simulation cycles : 10,000
- Flow control : Wormhole with VCs
- Injection rate : 0.24
- Traffic pattern : uniform_random
- Link criticality : High

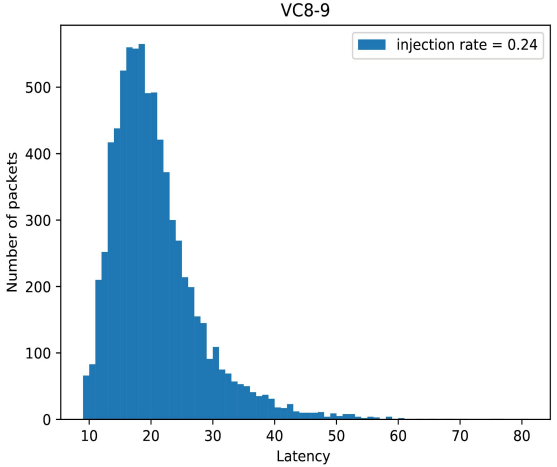
Assigning criticality to VCs in each VNET :

- Facilitates criticality usage in each message class.
- Eg:- a high critical Load request from core to memory.
- Similarly, the response will travel from memory to core in HI-crit VCs only.

Results: Priority to HI-Crit VCs







- Compared to VNET2_baseline, tail latency of VCs 8-9 has come down drastically.
- Tail latency of VCs 10-11 has degraded w.r.t baseline case.

NoC Generator : OpenSMART

- Can generate RTL for a network configuration like number of VCs, routing algorithm, topology, router pipeline stages, etc.

Language	Bluespec, Chisel
Flow Control	Wormhole with VCs
Topology	Mesh, Look-ahead XY routing
Buffer Management	Credit
Router Microarchitecture	1 cycle/2 cycle , SMART
Packet size	1 flit
Traffic Generator support/Stress Test	Uniform_random, Bit_complement

- Interface between NI and core will be via TileLink transactors.
- Implemented FPGA synthesizable LFSR modules for traffic generation to stress test NoC.

et al. H Kwon, 2017: "OpenSMART: Single-cycle multi-hop NoC generator in BSV and Chisel"

WIP/Future Work

- Modelling traffic in Gem5 to create hot modules to increase interference.
- Implementing Multi-flit packet support in OpenSmart.
- Incorporating criticality in OpenSMART routers.
- Modifying NI to insert packets into appropriate VCs.
- Run-time monitoring of flow parameters at NI to determine link criticality.
- Preventing degradation of low critical packets by methods such as resetting link criticality, slack aware arbitration [3].
- Performance counters.

[3] S. Tobuschat, R. Ernst, "Efficient latency guarantees for mixed-criticality networks-on-chip", RTAS 2017

Cache Coherence

- Using ProtoGen [4] to generate cache and directory controller state machine.
- ProtoGen takes a Stable State Protocol (SSP) and generates concurrent, safety and deadlock free state machine.
- By default, it supports generation of MSI, MESI & MOSI protocols.
- We plan to explore time predictable coherence protocols and use Protogen to generate the controllers.

[4] et al. N. Oswald, ProtoGen:Automatically Generating Directory Cache Coherence Protocols from Atomic Specifications, ISCA 2018

Thank you

Repositories

Gem5: https://gitlab.com/shaktiproject/tools/shakti-gem5

OpenSMART: https://gitlab.com/shaktiproject/uncore/OpenSMART

Shakti TLS : https://gitlab.com/shaktiproject/cores/c-class/tree/136-c-class-tls

SHAKTI