ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

STMicroelectronics

A Fine-Grained Multithreaded RISC-V Core for Efficient and Predictable Real-Time Microcontrollers

Arbi Gunbardhi*[1]*, *Riccardo Tedeschi*[1]*,*
Filippo Grillotti[2]*, Fabio De Ambroggi*[2]*, Elio Guidetti*[2]*,*
Luca Benini[1,3]*, Davide Rossi*[1,4]

University of Bologna, Italy[1] STMicroelectronics, Italy[2]
ETH Zürich, Switzerland[3] Fondazione Chips-IT[4]

Motivation

Context

Microcontroller Unit (**MCU**) designs must adhere to **strict constraints**:

- **Silicon Area**
- **Power Consumption**
- **Determinism**

Motivation

Context

Microcontroller Unit (**MCU**) designs must adhere to **strict constraints**:

- **Silicon Area**
- **Power Consumption**
- **Determinism**

Problem

Tight constraints preclude complex **mitigation logic**, making the pipeline remains susceptible to:

- **Data dependencies**
- **Control Hazards**

Motivation

Context

Microcontroller Unit (**MCU**) designs must adhere to **strict constraints**:

- **Silicon Area**
- **Power Consumption**
- **Determinism**

Problem

Tight constraints preclude complex **mitigation logic**, making the pipeline remains susceptible to:

- **Data dependencies**
- **Control Hazards**

Solution

FGMT can address both in **efficient way!**

Contribution

1 **Dual-Threaded Fine-Grained Multithreading (FGMT) on a RISC-V Pipeline**

Contribution

1. **Dual-Threaded Fine-Grained Multithreading (FGMT) on a RISC-V Pipeline**
2. **Thread Forwarding (TFW) Enhancement to FGMT**

Contribution

1. Dual-Threaded Fine-Grained Multithreading (FGMT) on a RISC-V Pipeline
2. Thread Forwarding (TFW) Enhancement to FGMT
3. Enhanced Temporal Predictability

Contribution

1. Dual-Threaded Fine-Grained Multithreading (FGMT) on a RISC-V Pipeline
2. Thread Forwarding (TFW) Enhancement to FGMT
3. Enhanced Temporal Predictability
4. Physical Implementation & PPA Analysis in commercial technology

Fine Grained Multithreading (FGMT)

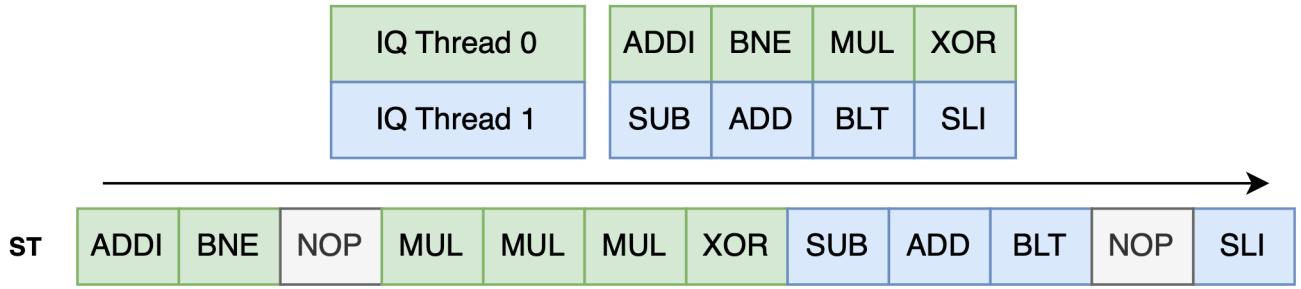
✅ **Fine-grained multithreading:** A hardware technique that maximizes overall throughput at the cost of individual thread speed.

Fine Grained Multithreading (FGMT)

✅ **Fine-grained multithreading:** A hardware technique that maximizes overall throughput at the cost of individual thread speed.

Idea

Multiple workloads to execute

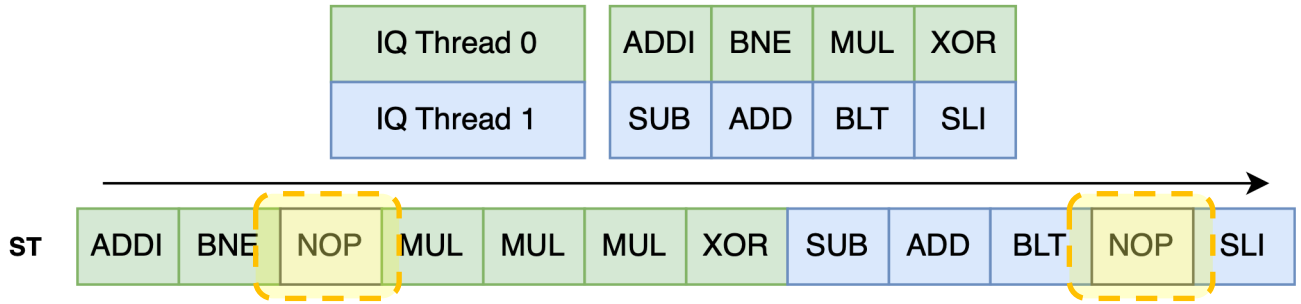


Fine Grained Multithreading (FGMT)

✅ **Fine-grained multithreading:** A hardware technique that maximizes overall throughput at the cost of individual thread speed.

Idea

Multiple workloads to execute



Fine Grained Multithreading (FGMT)

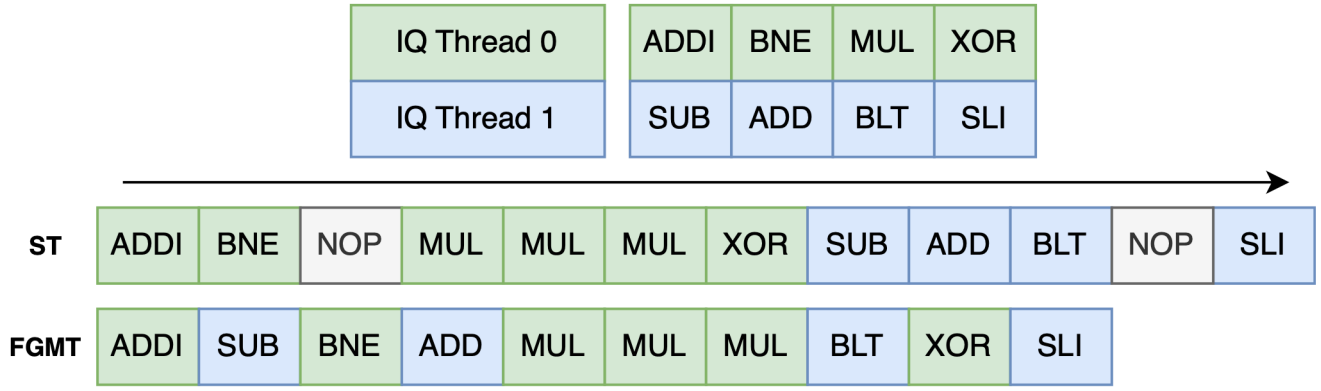
✅ **Fine-grained multithreading:** A hardware technique that maximizes overall throughput at the cost of individual thread speed.

Idea

Multiple workloads to execute

⬇

Each cycle, fetch and execute **preemptively** in **round robin**



Fine Grained Multithreading (FGMT)

✅ **Fine-grained multithreading:** A hardware technique that maximizes overall throughput at the cost of individual thread speed.

Idea

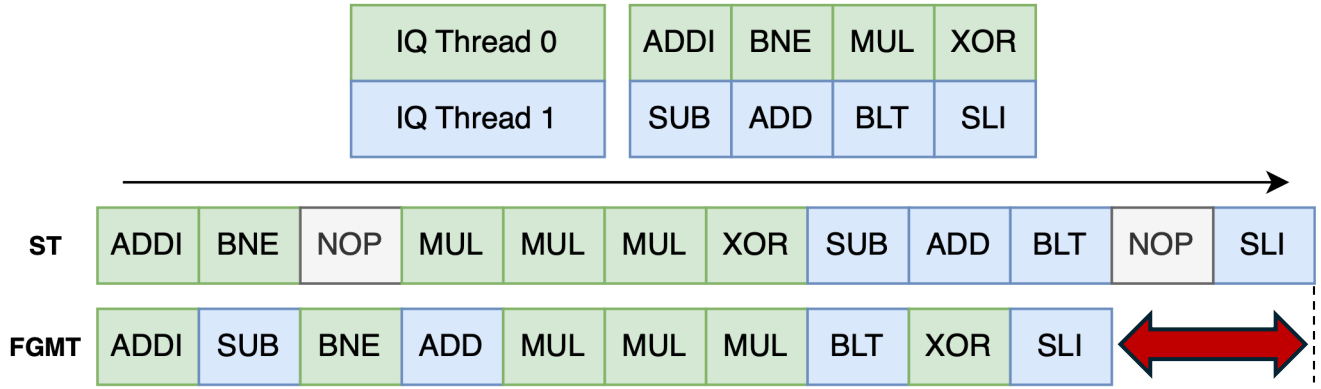
Multiple workloads to execute

↓

Each cycle, fetch and execute **preemptively** in **round robin**

↓

Hide hazards by executing instructions from other threads **instead of** having pipeline **bubbles**



Fine Grained Multithreading (FGMT)

✅ **Fine-grained multithreading:** A hardware technique that maximizes overall throughput at the cost of individual thread speed.

Idea

Multiple workloads to execute

↓

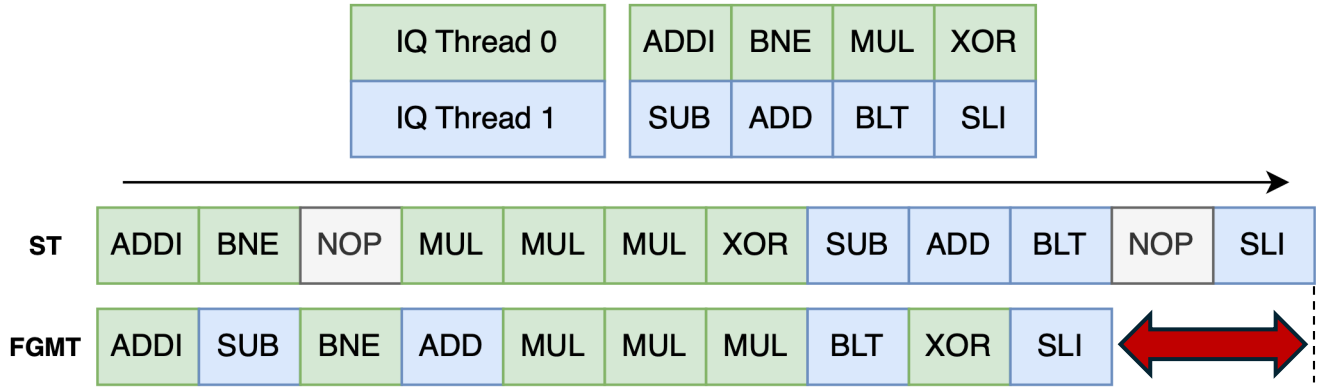
Each cycle, fetch and execute **preemptively** in **round robin**

↓

Hide hazards by executing instructions from other threads **instead of** having pipeline **bubbles**

Solves Control Hazards

Solves Data Hazards



Fine Grained Multithreading (FGMT)

✅ **Fine-grained multithreading:** A hardware technique that maximizes overall throughput at the cost of individual thread speed.

Idea

Multiple workloads to execute

↓

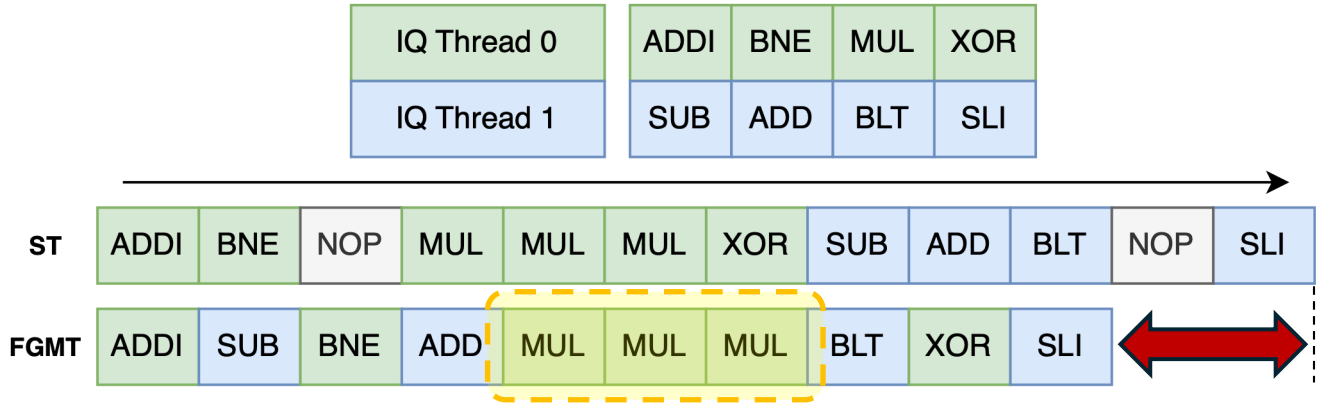
Each cycle, fetch and execute **preemptively** in **round robin**

↓

Hide hazards by executing instructions from other threads **instead of** having pipeline **bubbles**

Solves Control Hazards

Solves Data Hazards



Solves NOT Structurale Hazards

Thread Forwarding (TFW)

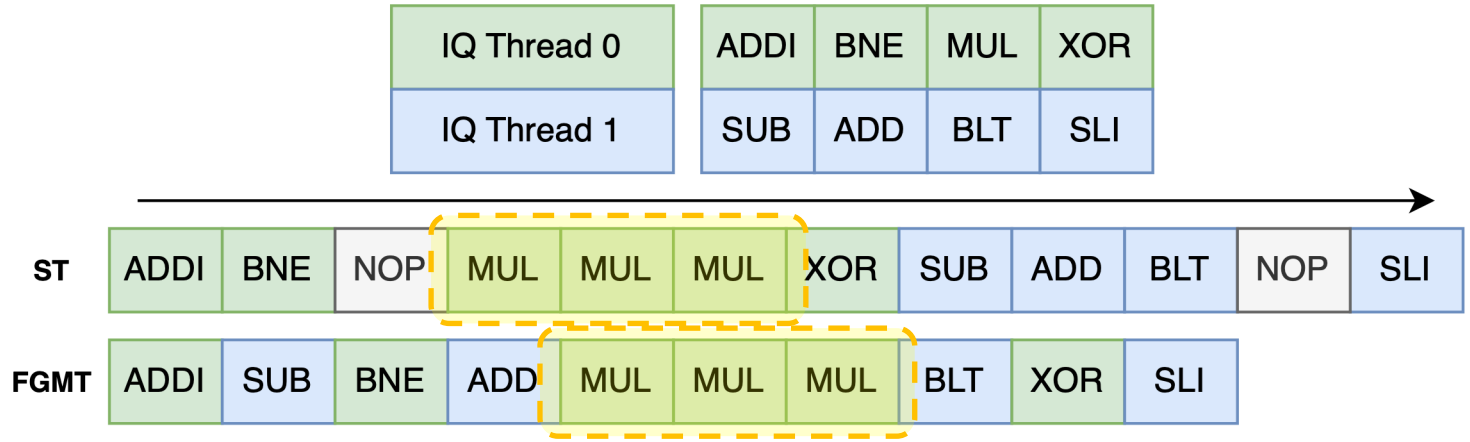
✅ **Thread Forwarding :** An FGMT enhancement designed to mitigate throughput stalls caused by multicycle instructions

Thread Forwarding (TFW)

✅ **Thread Forwarding :** An FGMT enhancement designed to mitigate throughput stalls caused by multicycle instructions

Idea

While a **multicycle** is in execution, the **entire pipeline is stalled**



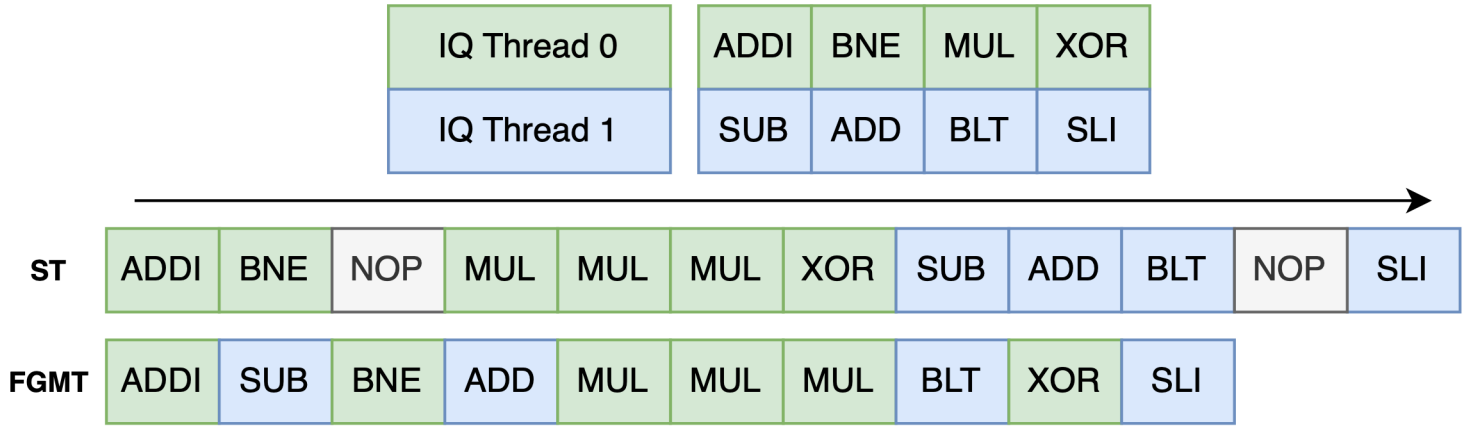
Thread Forwarding (TFW)

✅ **Thread Forwarding :** An FGMT enhancement designed to mitigate throughput stalls caused by multicycle instructions

Idea

While a **multicycle** is in execution, the **entire pipeline is stalled**

We have some **idle functional units**



Thread Forwarding (TFW)

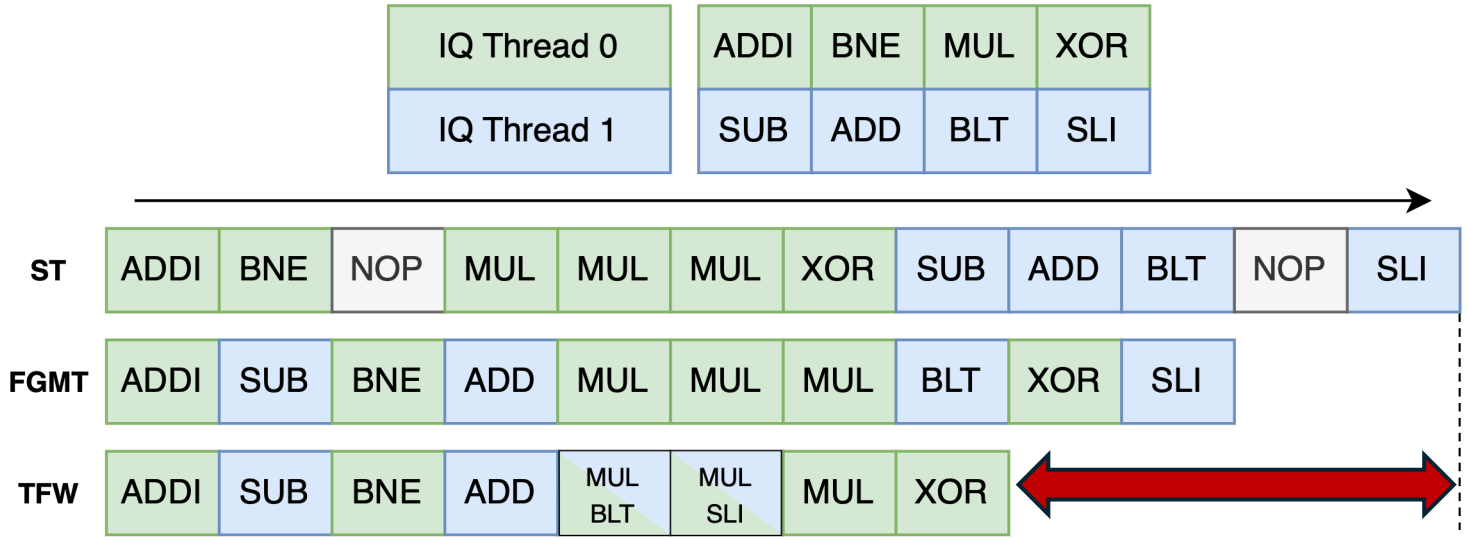
✅ **Thread Forwarding :** An FGMT enhancement designed to mitigate throughput stalls caused by multicycle instructions

Idea

While a **multicycle** is in execution, the **entire pipeline is stalled**

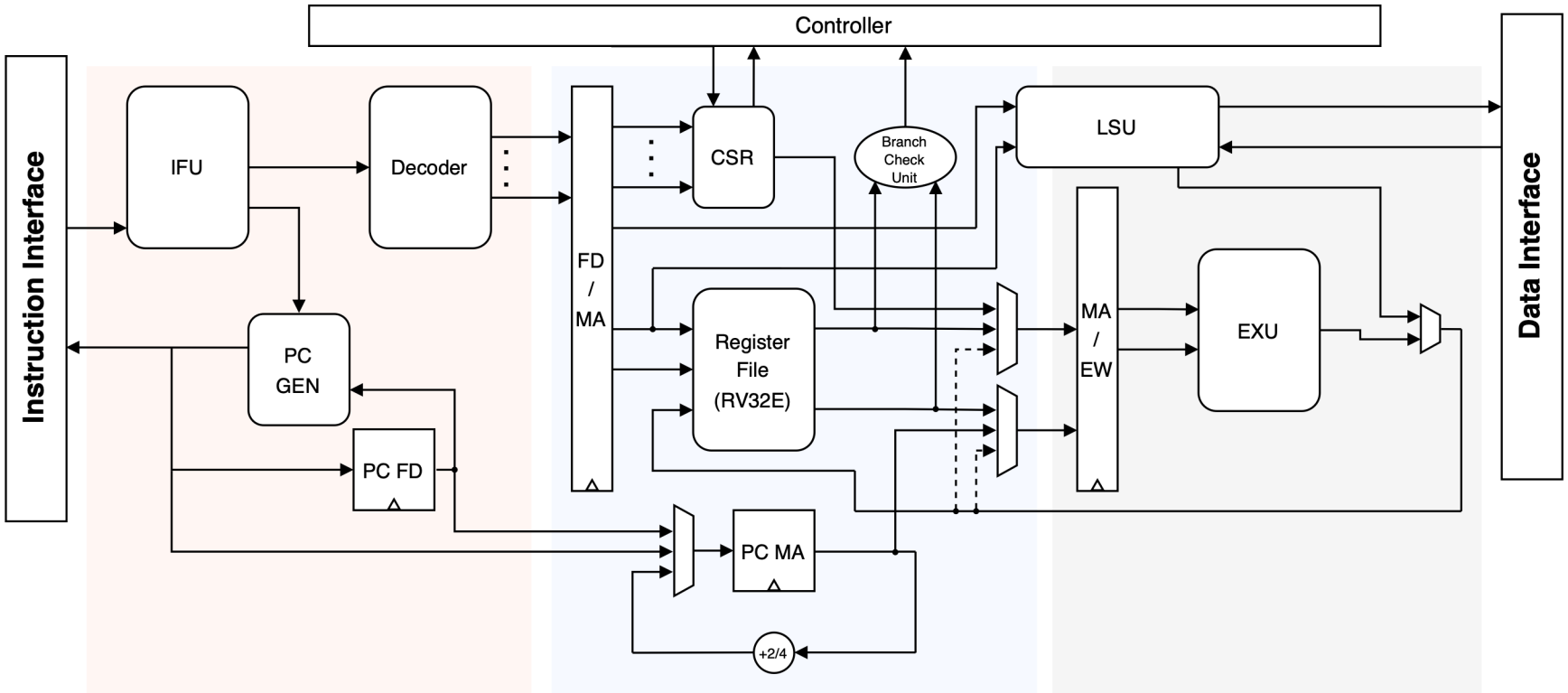
We have some **idle functional units**

The **alternative thread** can **utilize** the **idle** functional units

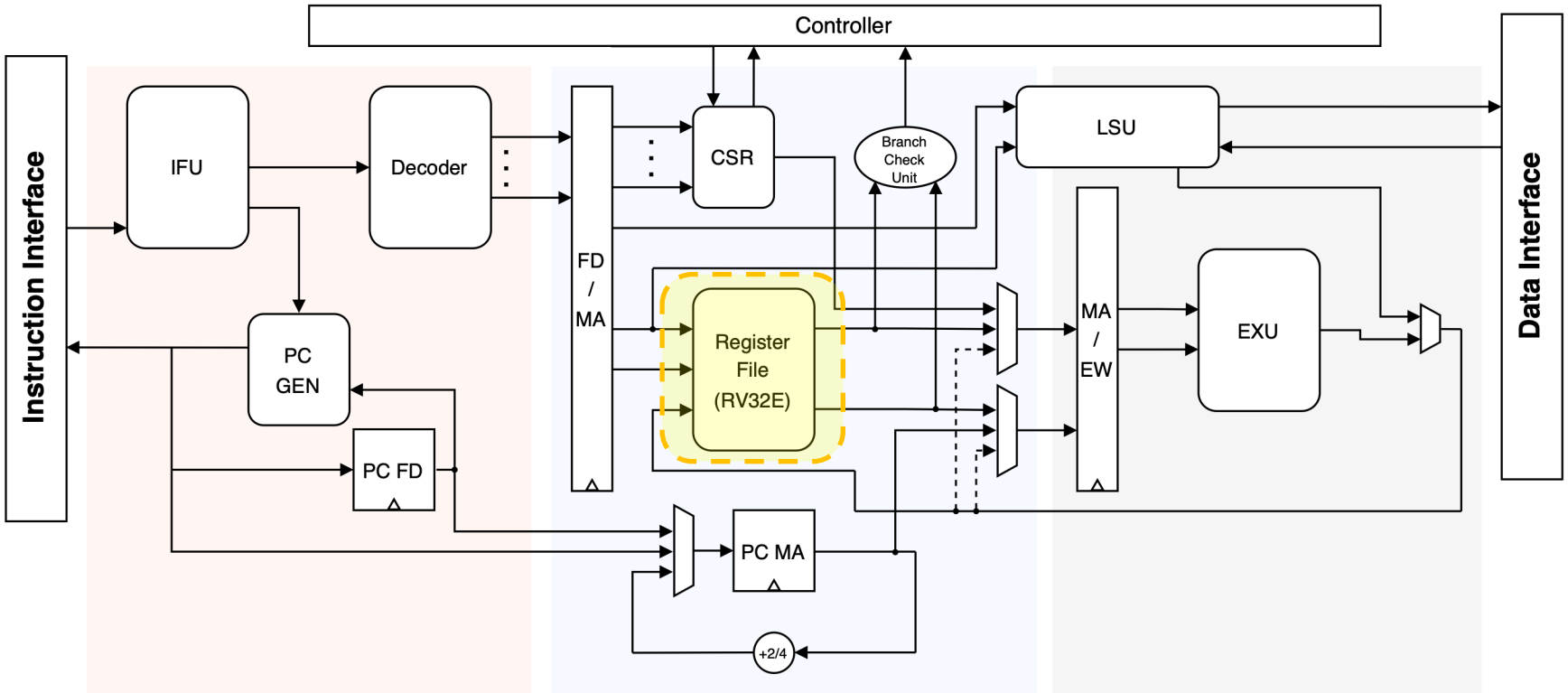


Baseline core: STRIVE-L

STRIVE-L, an industrial **RISC-V** baseline optimized for low-area and low-power **embedded applications**



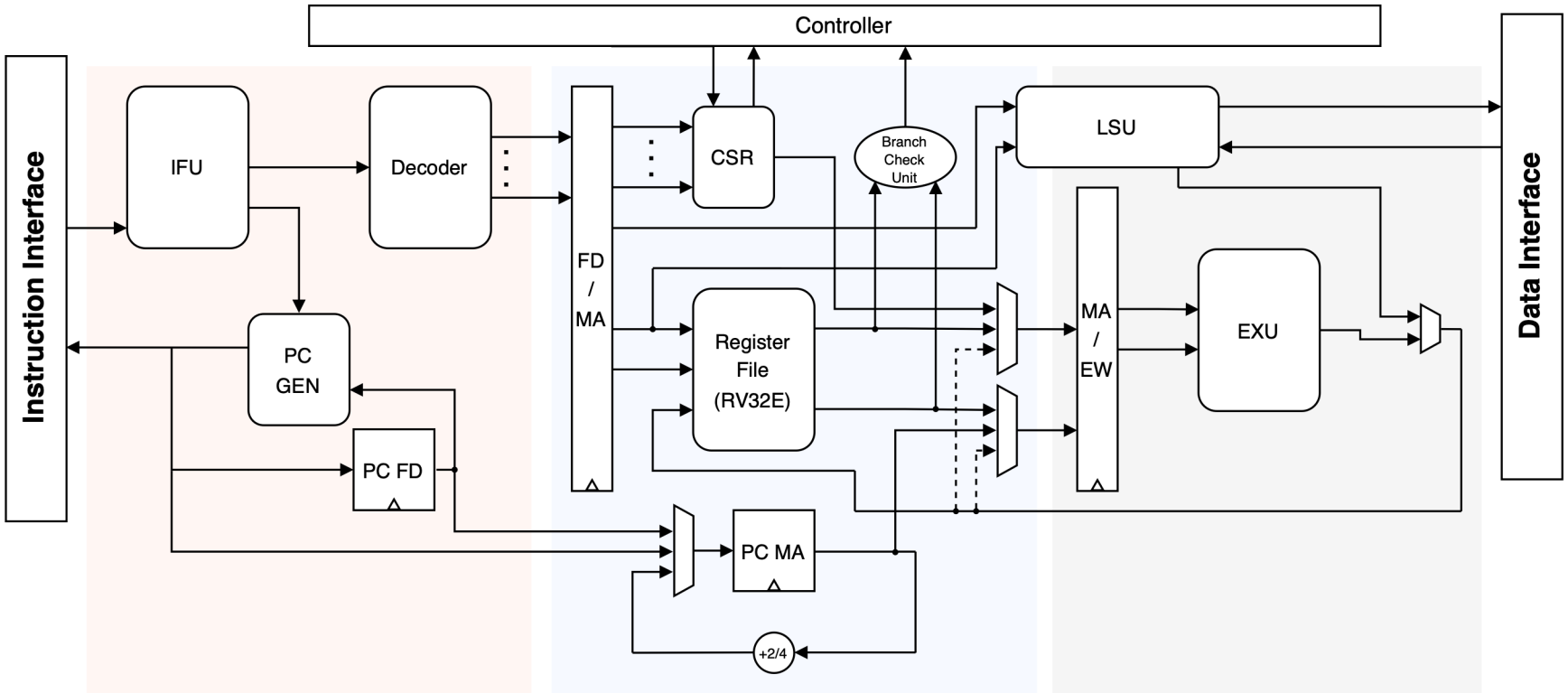
Baseline core: STRIVE-L



STRIVE-L, an industrial **RISC-V** baseline optimized for low-area and low-power **embedded applications:**

- **Architecture:** *32-bit RV32E with **C** and **M***

Baseline core: STRIVE-L



FD
Fetch-Decode

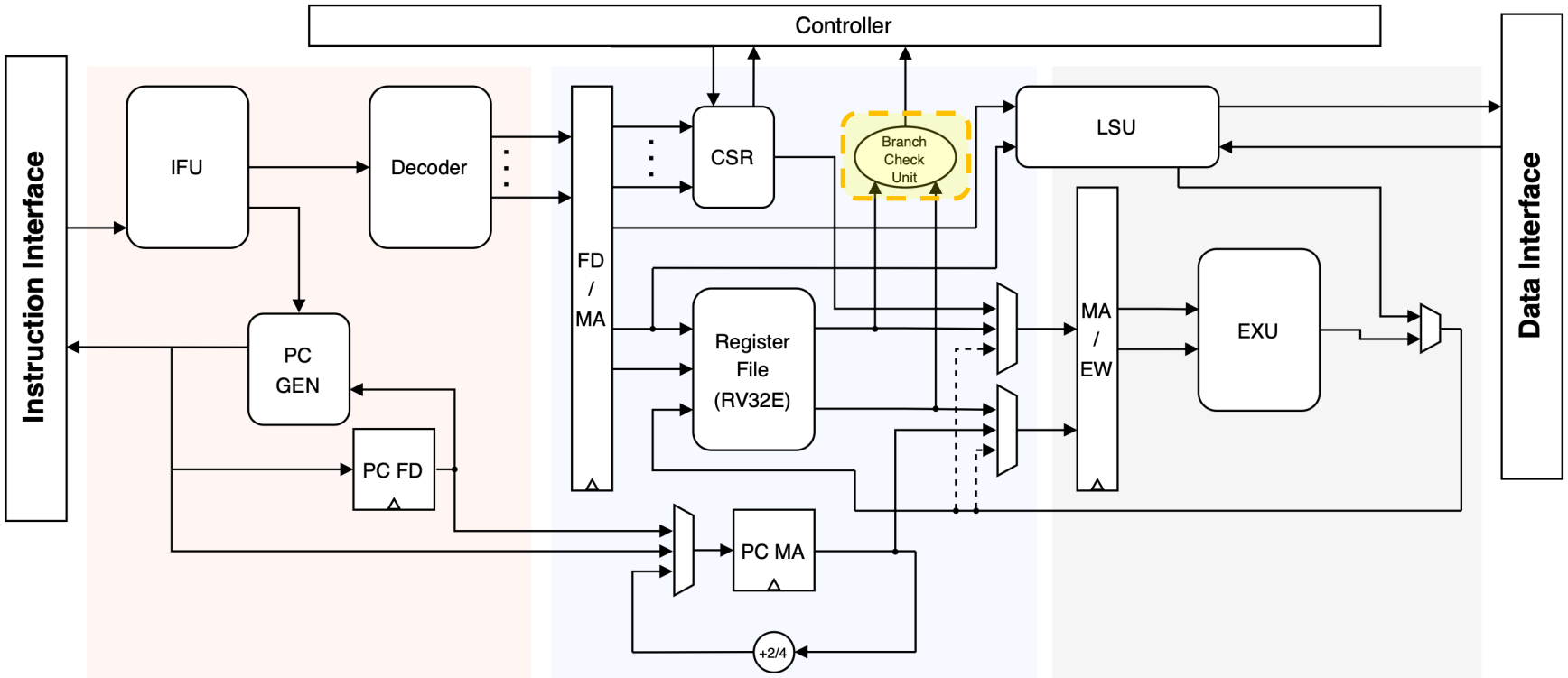
MA
Memory-Access

EW
Execute-Write

STRIVE-L, an industrial **RISC-V** baseline optimized for low-area and low-power **embedded applications:**

- **Architecture:** *32-bit RV32E with **C** and **M***
- **Pipeline:** ***3-stage** in-order*

Baseline core: STRIVE-L



FD
Fetch-Decode

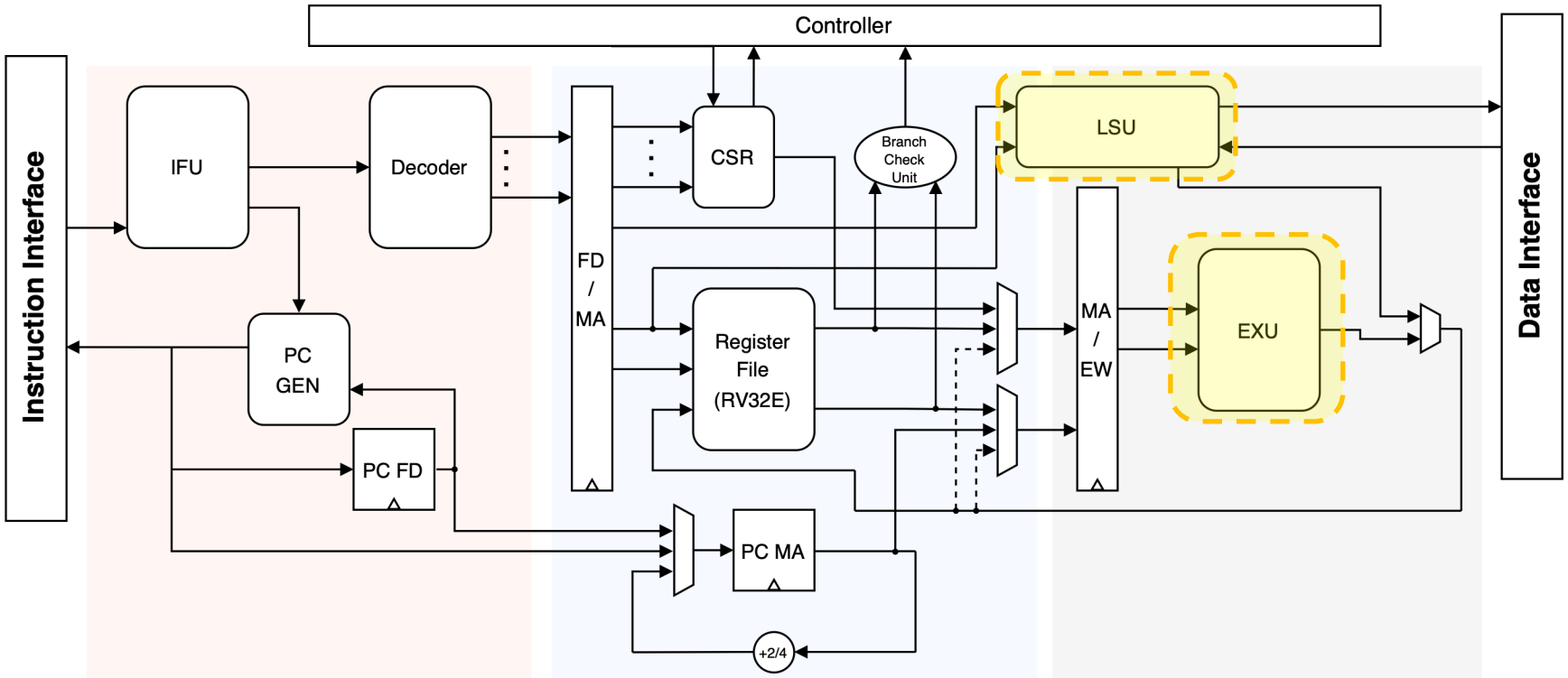
MA
Memory-Access

EW
Execute-Write

STRIVE-L, an industrial **RISC-V** baseline optimized for low-area and low-power **embedded applications:**

- **Architecture:** *32-bit RV32E with **C** and **M***
- **Pipeline:** ***3-stage** in-order*
- **Control Flow:** ***Static** branch **prediction**, resolved in the **MA stage → 2 cyles latency on misspredicts***

Baseline core: STRIVE-L



FD
Fetch-Decode

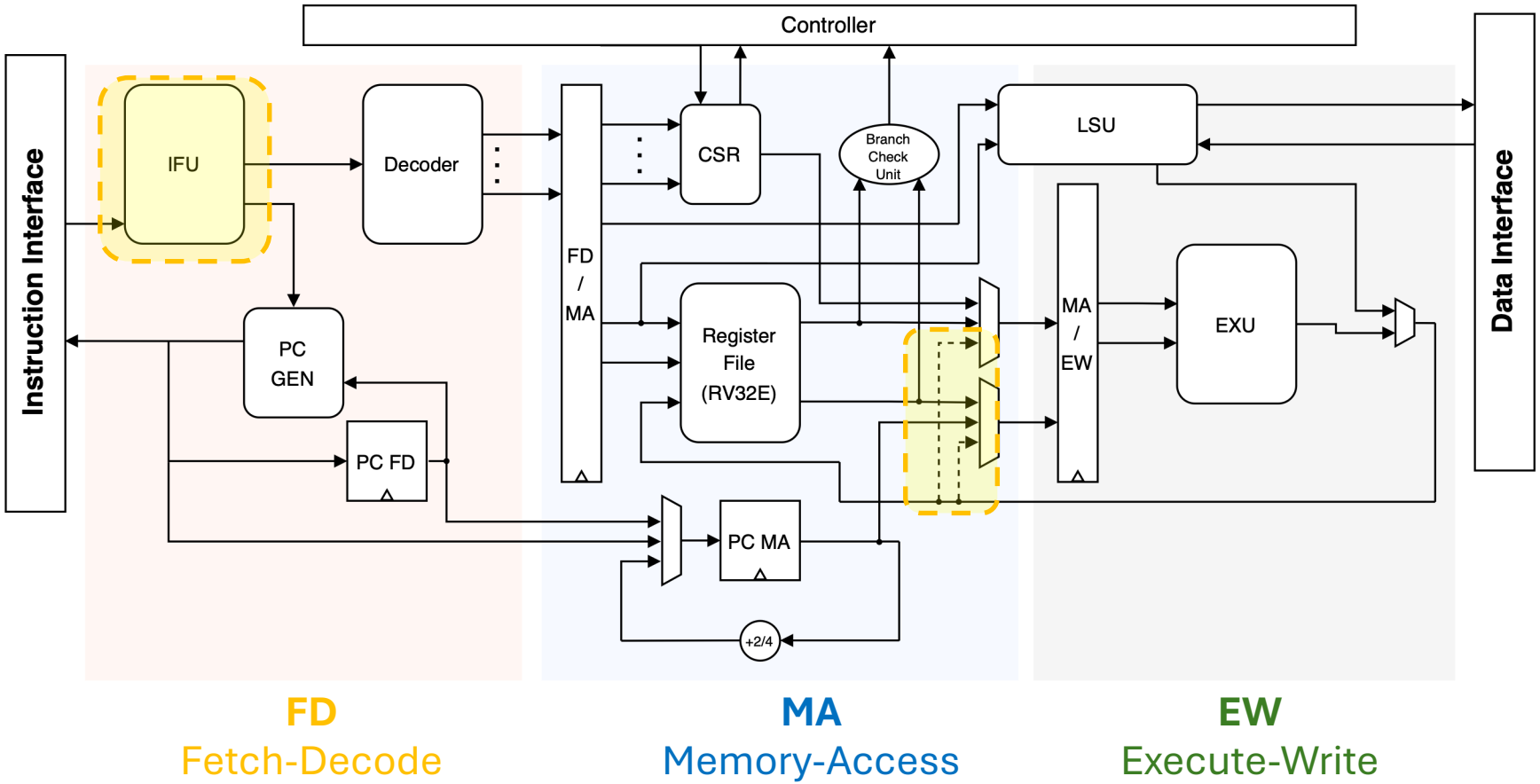
MA
Memory-Access

EW
Execute-Write

STRIVE-L, an industrial **RISC-V** baseline optimized for low-area and low-power **embedded applications:**

- **Architecture:** *32-bit RV32E with **C** and **M***
- **Pipeline:** ***3-stage** in-order*
- **Control Flow:** ***Static** branch **prediction**, resolved in the **MA stage → 2 cyles latency on misspredicts***
- **Execution Latency**
  - *3-cycle **mul***
  - *1-cycle **load-use***

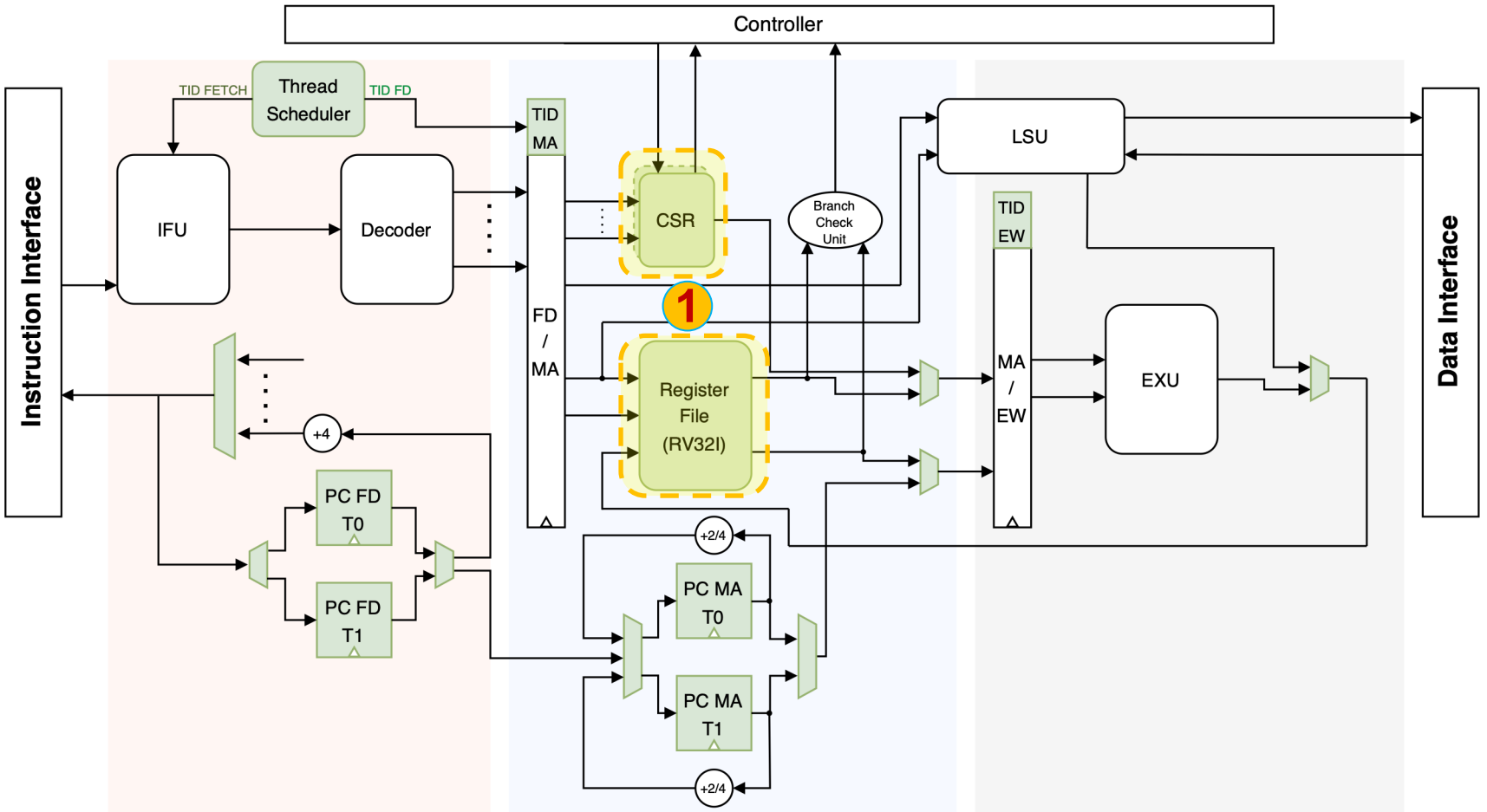
Baseline core: STRIVE-L



STRIVE-L, an industrial **RISC-V** baseline optimized for low-area and low-power **embedded applications:**

- **Architecture:** *32-bit RV32E with **C** and **M***
- **Pipeline:** ***3-stage** in-order*
- **Control Flow:** ***Static** branch **prediction**, resolved in the **MA stage → 2 cyles latency on mispredicts***
- **Execution Latency**
  - *3-cycle **mul***
  - *1-cycle **load-use***
- **Hazard Mitigation:** *Data **bypassing** and 8-byte **prefetch buffer***

FGMT Architecture



FD
Fetch-Decode

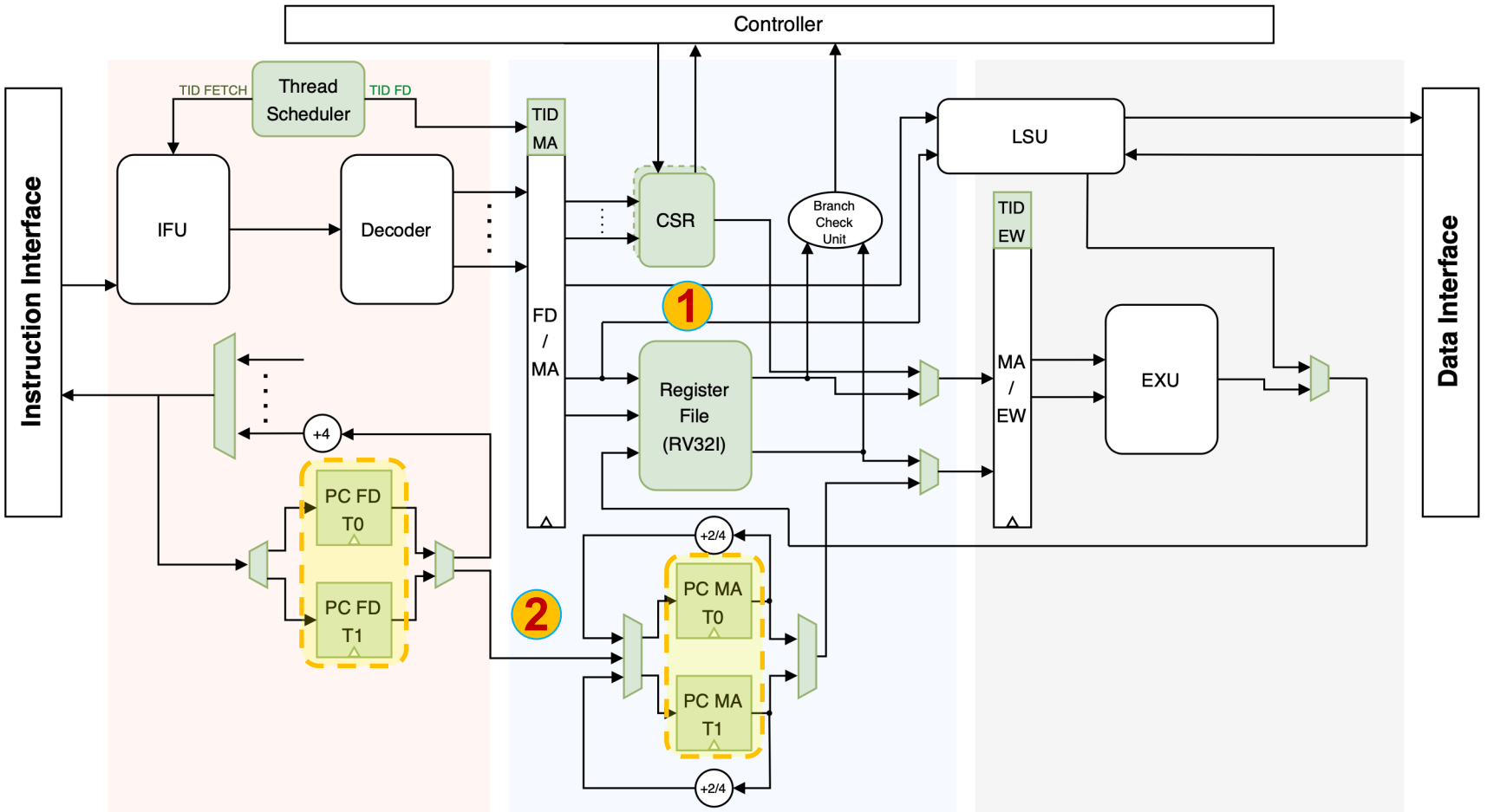
MA
Memory-Access

EW
Execute-Write

1 *Context Duplication*

*The **register file** capacity is doubled and **CSRs** are **duplicated**.*

FGMT Architecture



FD
Fetch-Decode

MA
Memory-Access

EW
Execute-Write

1 Context Duplication

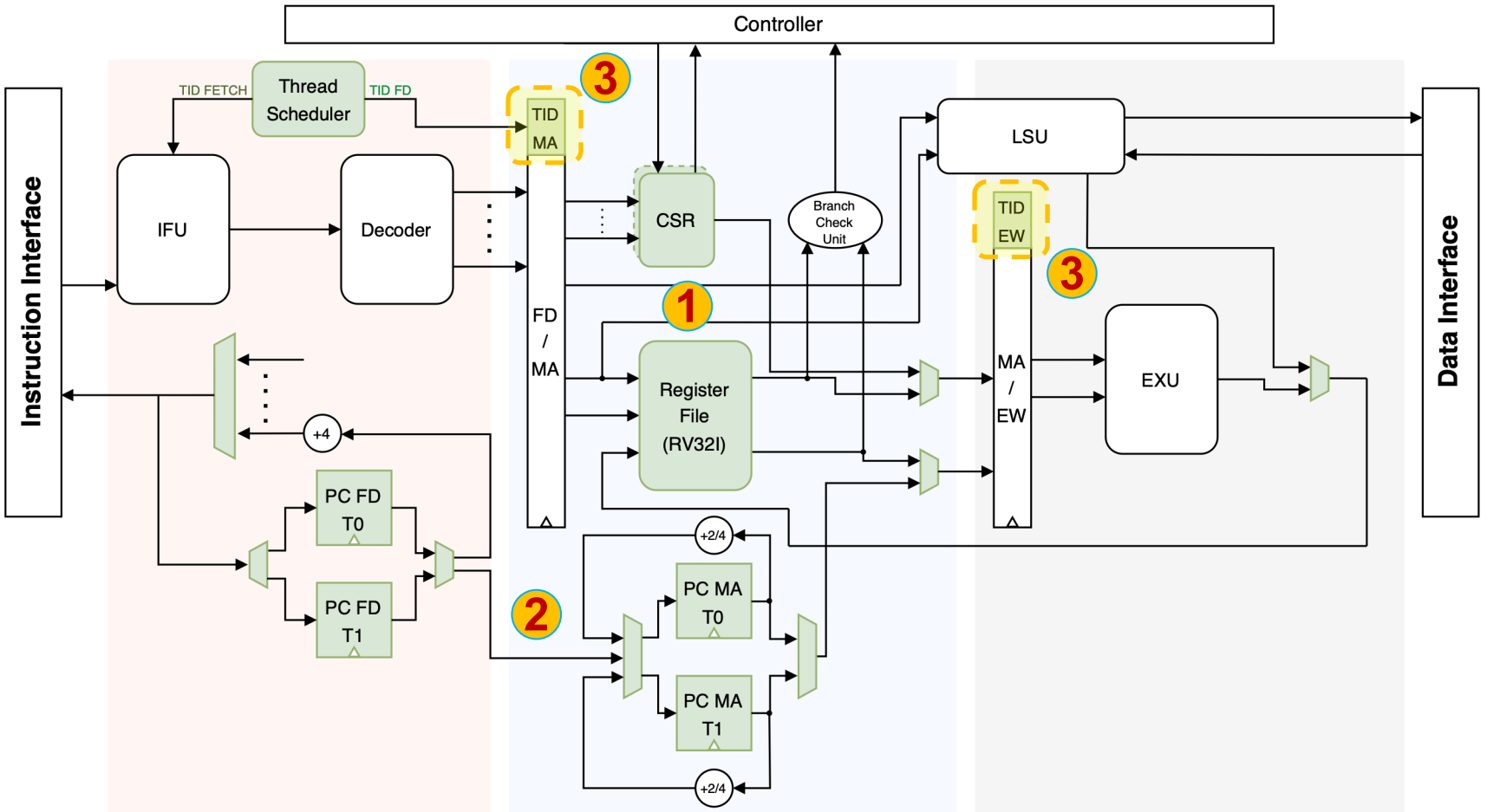
*The **register file** capacity is doubled and **CSRs** are **duplicated**.*

2 PC Duplication

*- **Primary PC** governs the **FD stage**.*

*- **Secondary PC** in the **MA stage** calculates jump targets and **PC-relative offsets**.*

FGMT Architecture



FD
Fetch-Decode

MA
Memory-Access

EW
Execute-Write

1 *Context Duplication*

*The **register file** capacity is doubled and **CSRs** are **duplicated**.*

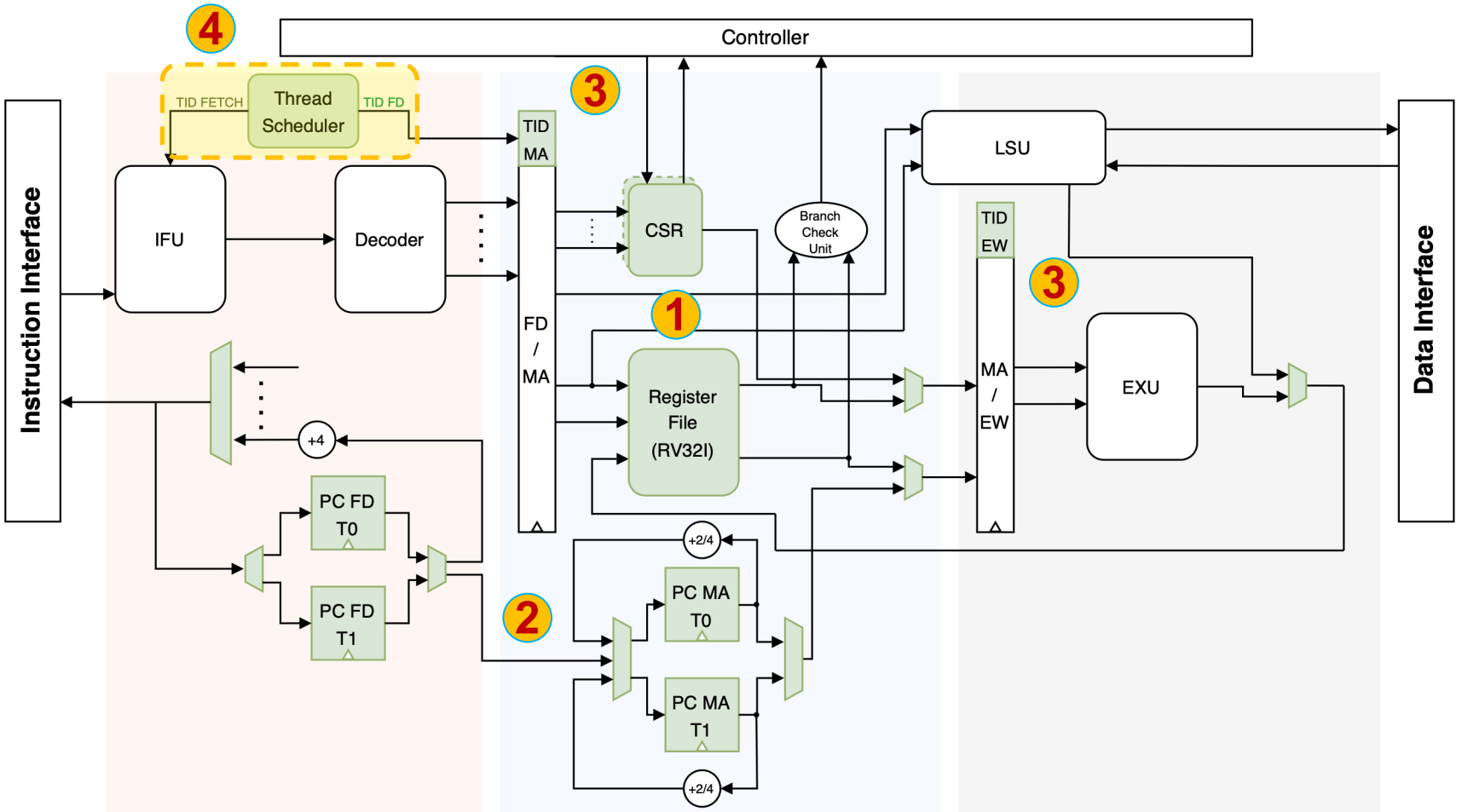
2 *PC Duplication*

*- **Primary PC** governs the **FD stage**.*
*- **Secondary PC** in the **MA stage** calculates jump targets and **PC-relative offsets**.*

3 *Thread ID* (TID)

*- Acts as the **msb** to **partition the register** file in two.*
*- **Regulates access** to **shared resources**.*

FGMT Architecture



1 Context Duplication

*The **register file** capacity is doubled and **CSRs** are **duplicated**.*

2 PC Duplication

- ***Primary PC** governs the **FD stage**.*
- ***Secondary PC** in the **MA stage** calculates jump targets and **PC-relative offsets**.*

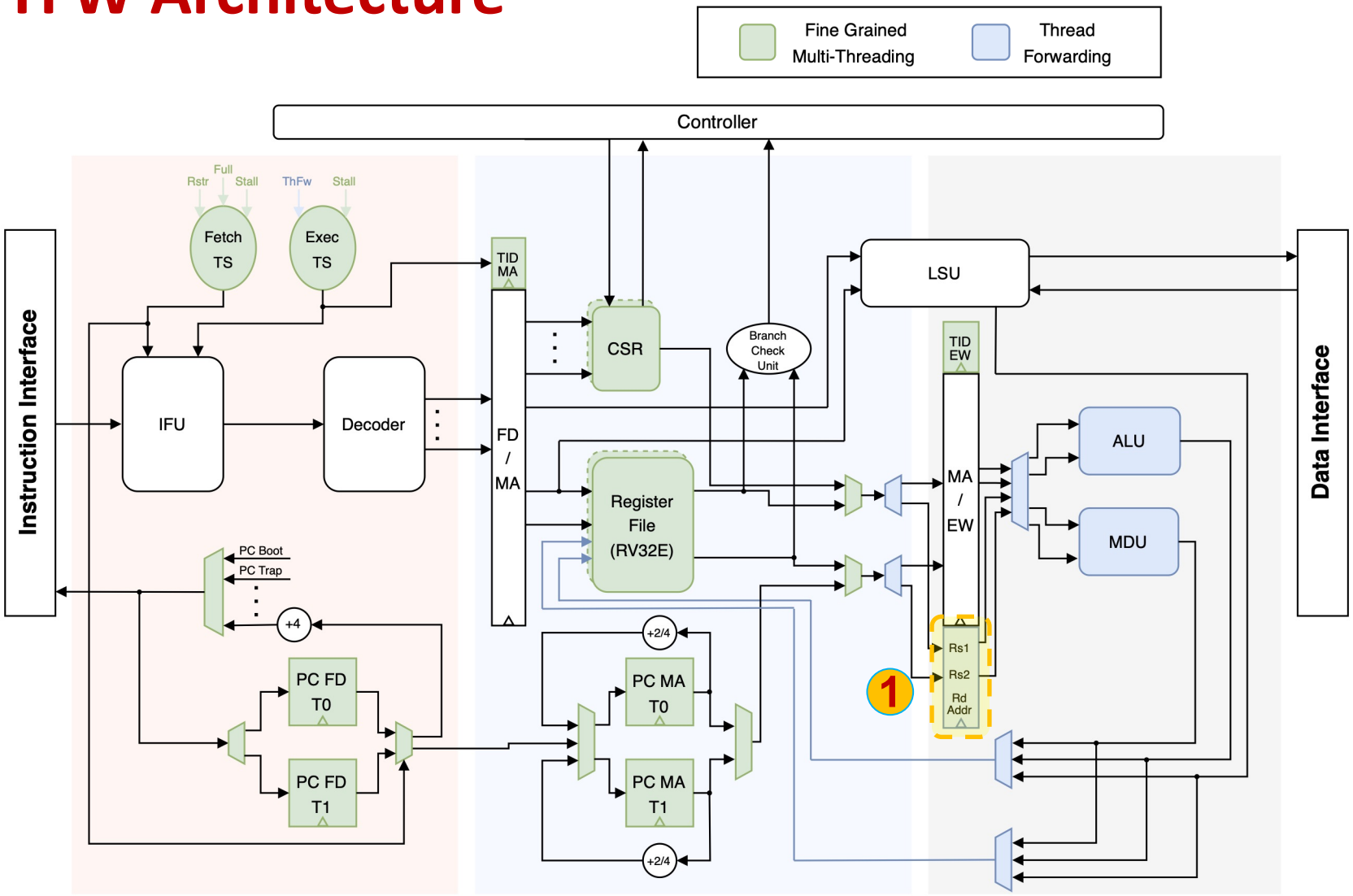
3 *Thread ID* (TID)

- *Acts as the **msb** to **partition the register** file in two.*
- ***Regulates access** to **shared resources**.*

4 *Thread Scheduler*

***Strictly alternates** the active thread every clock cycle to **interleave instructions and mask hazards**.*

TFW Architecture



FD
Fetch-Decode

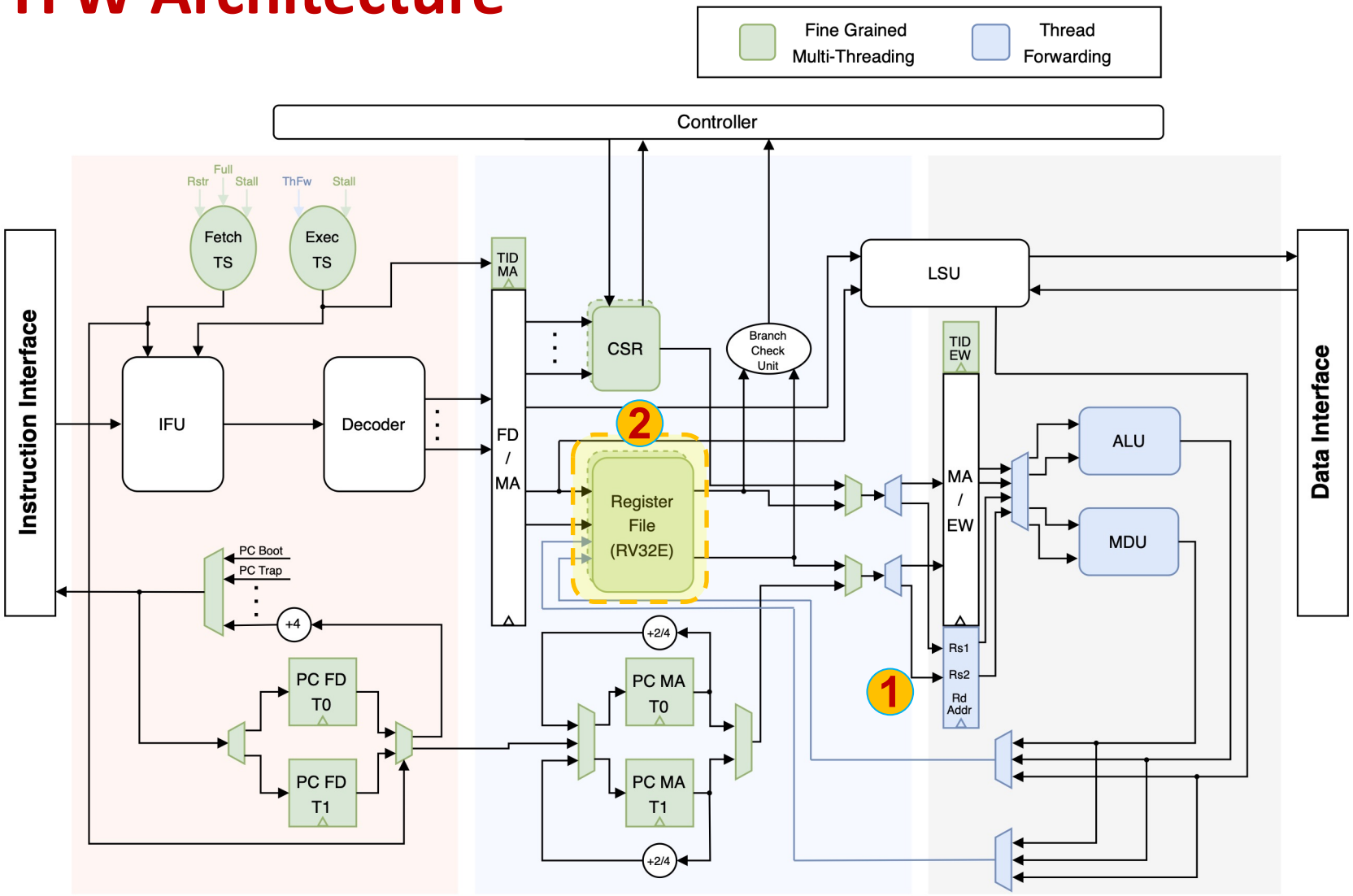
MA
Memory-Access

EW
Execute-Write

1 ***Pipeline registers***

Additional registers in the EW stage hold operands and the destination address to **preserve the state of the forwarded instruction**.

TFW Architecture



FD
Fetch-Decode

MA
Memory-Access

EW
Execute-Write

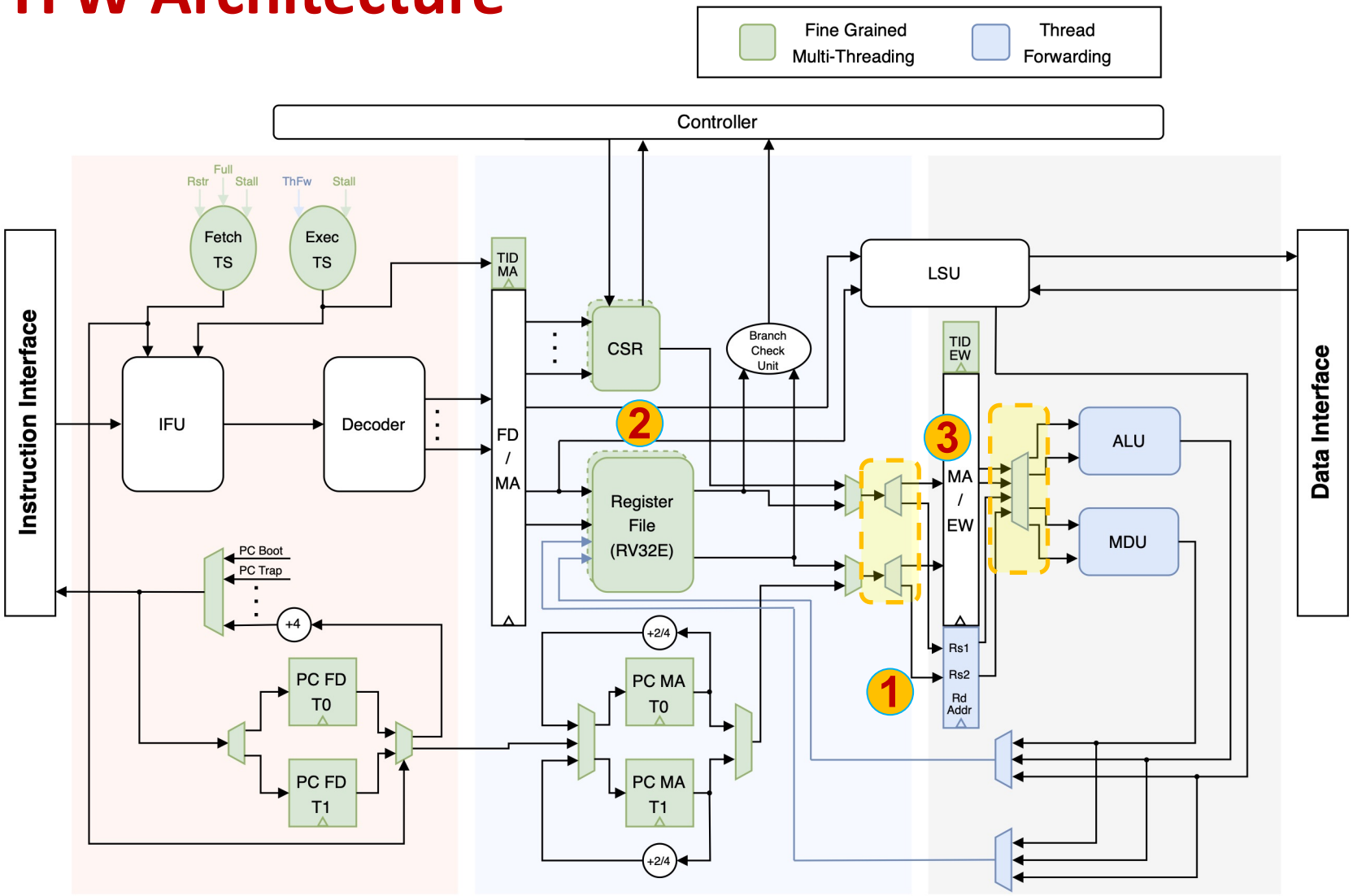
1 ***Pipeline registers***

Additional registers in the EW stage hold operands and the destination address to **preserve the state of the forwarded instruction**.

2 ***Register file cluster***

Register file is **instantiated twice**, providing independent write ports for **simultaneous writebacks**.

TFW Architecture



FD
Fetch-Decode

MA
Memory-Access

EW
Execute-Write

1 *Pipeline registers*

Additional registers in the EW stage hold operands and the destination address to **preserve the state of the forwarded instruction**.

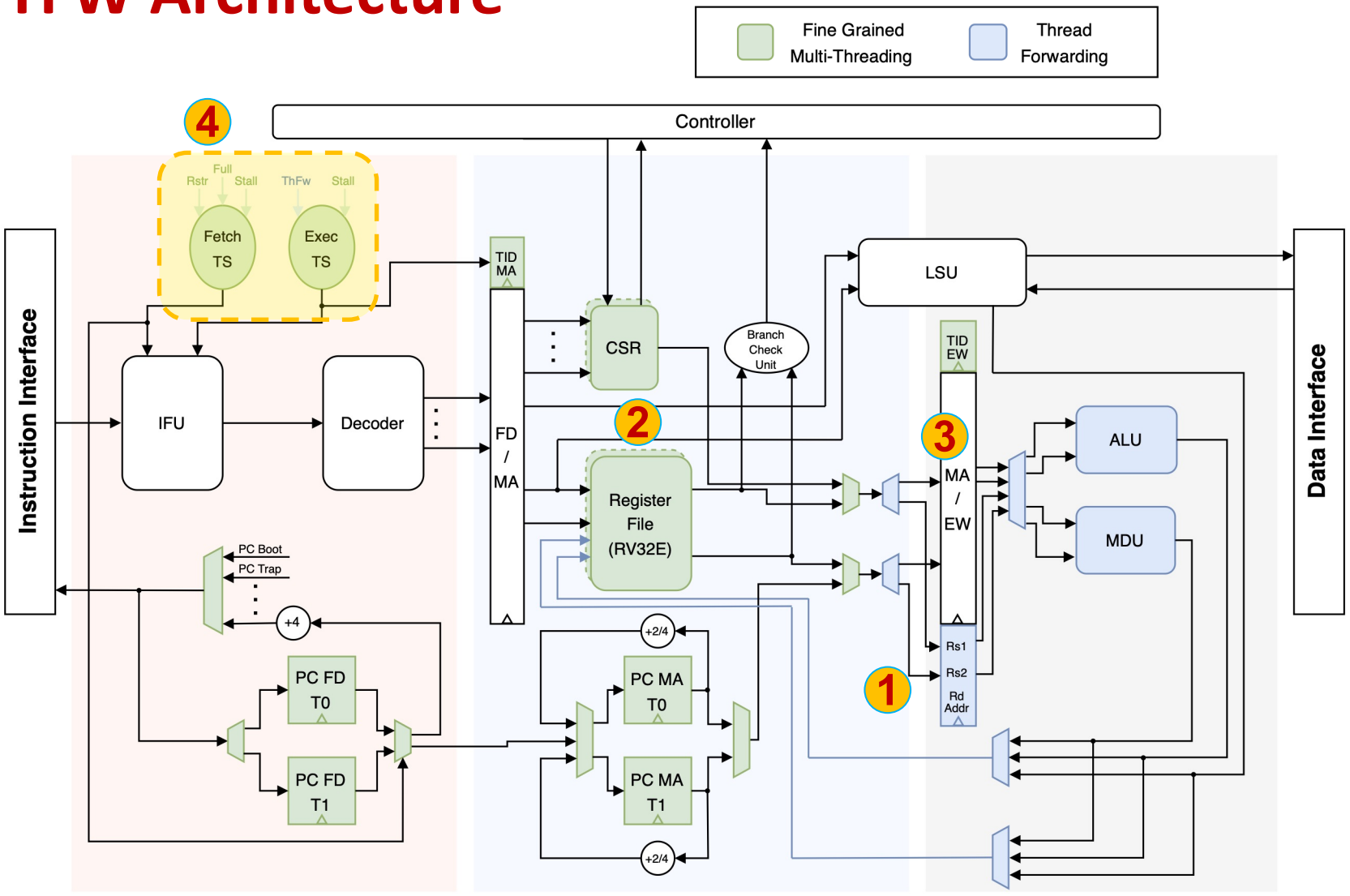
2 *Register file cluster*

Register file is **instantiated twice**, providing independent write ports for **simultaneous writebacks**.

3 *Steering logic*

Multiplexer network redirects **control signals** based on which thread currently occupies MDU.

TFW Architecture



FD
Fetch-Decode

MA
Memory-Access

EW
Execute-Write

1 *Pipeline registers*

Additional registers in the EW stage hold operands and the destination address to **preserve the state of the forwarded instruction**.

2 *Register file cluster*

Register file is **instantiated twice**, providing independent write ports for **simultaneous writebacks**.

3 *Steering logic*

Multiplexer network redirects **control signals** based on which thread currently occupies MDU.

4 *Thread Scheduler*

Suspends interleaving during **multicycle instructions** to fetch consecutive instructions from the alternate thread.

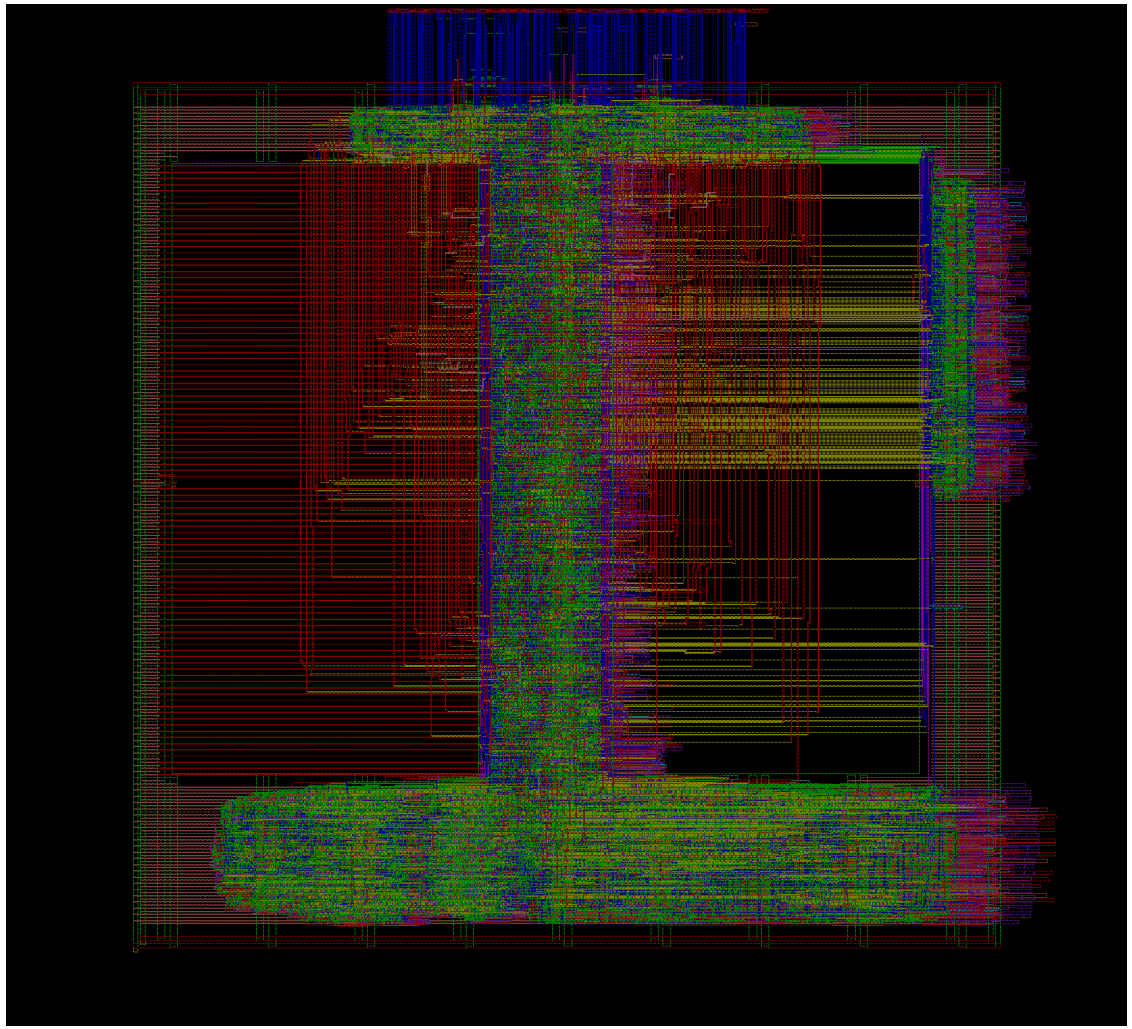
Methodology

- **Synthesis**
- **Tool:** DesignCompiler (Synopsys)
- **Frequency:** 300MHz
- **Tech:** C40
- **Corner**: SS, 1.05V, -40°C

Methodology

- **Synthesis**
- **Tool:** DesignCompiler (Synopsys)
- **Frequency:** 300MHz
- **Tech:** C40
- **Corner**: SS, 1.05V, -40°C

- **Place & Route**
- **Tool:** Innovus (Cadence)
- **Frequency:** 300MHz

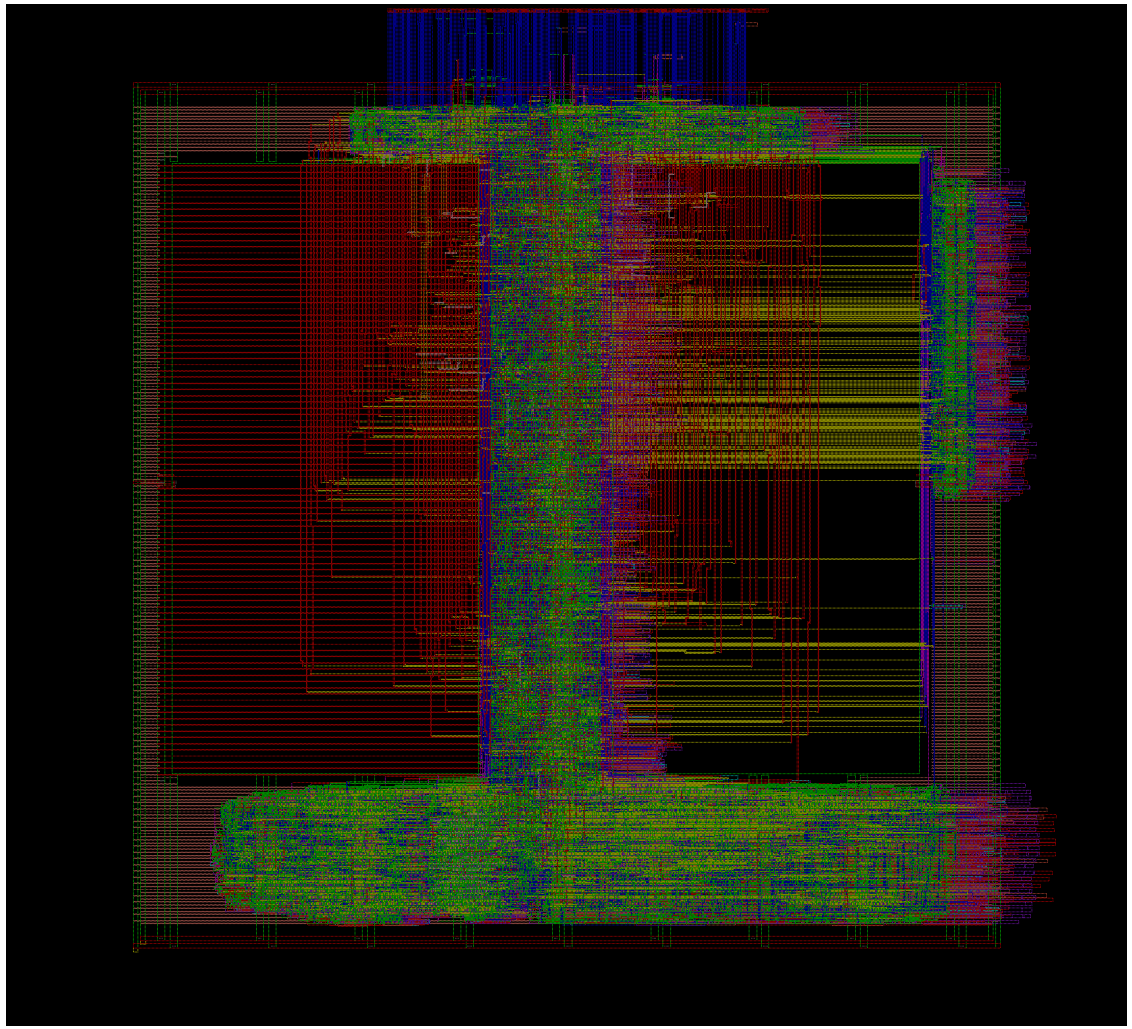


Methodology

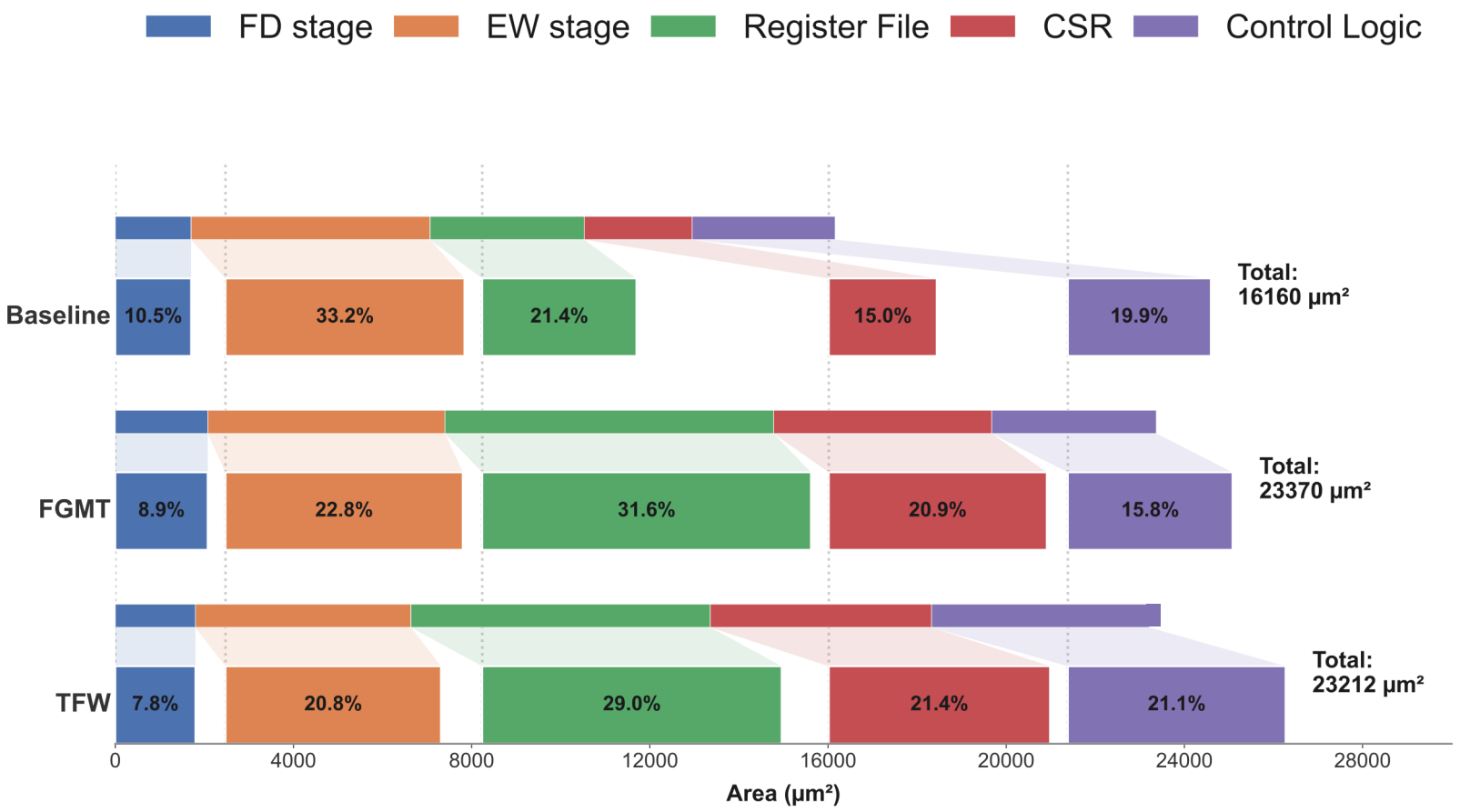
- **Synthesis**
- **Tool:** DesignCompiler (Synopsys)
- **Frequency:** 300MHz
- **Tech:** C40
- **Corner**: SS, 1.05V, -40°C

- **Place & Route**
- **Tool:** Innovus (Cadence)
- **Frequency:** 300MHz

- **Vector-driven power signoff**
- **Tool:** PrimeTime (Synopsys)
- **Corner**: TT, 1.10V, 25°C

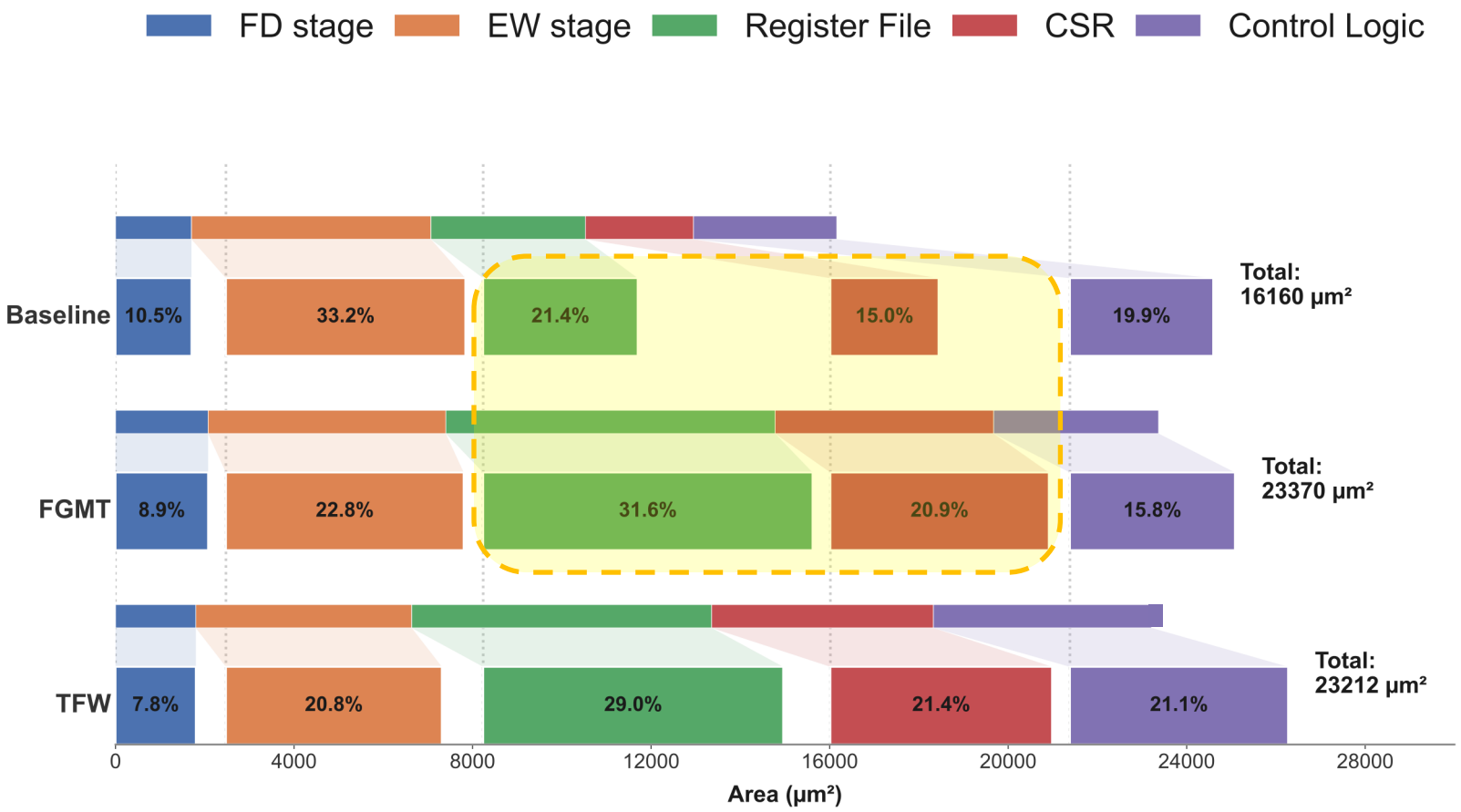


Area



- **Place & Route**
- **Frequency:** 300MHz
- **Tech:** C40
- **Corner**: SS, 1.05V, -40°C

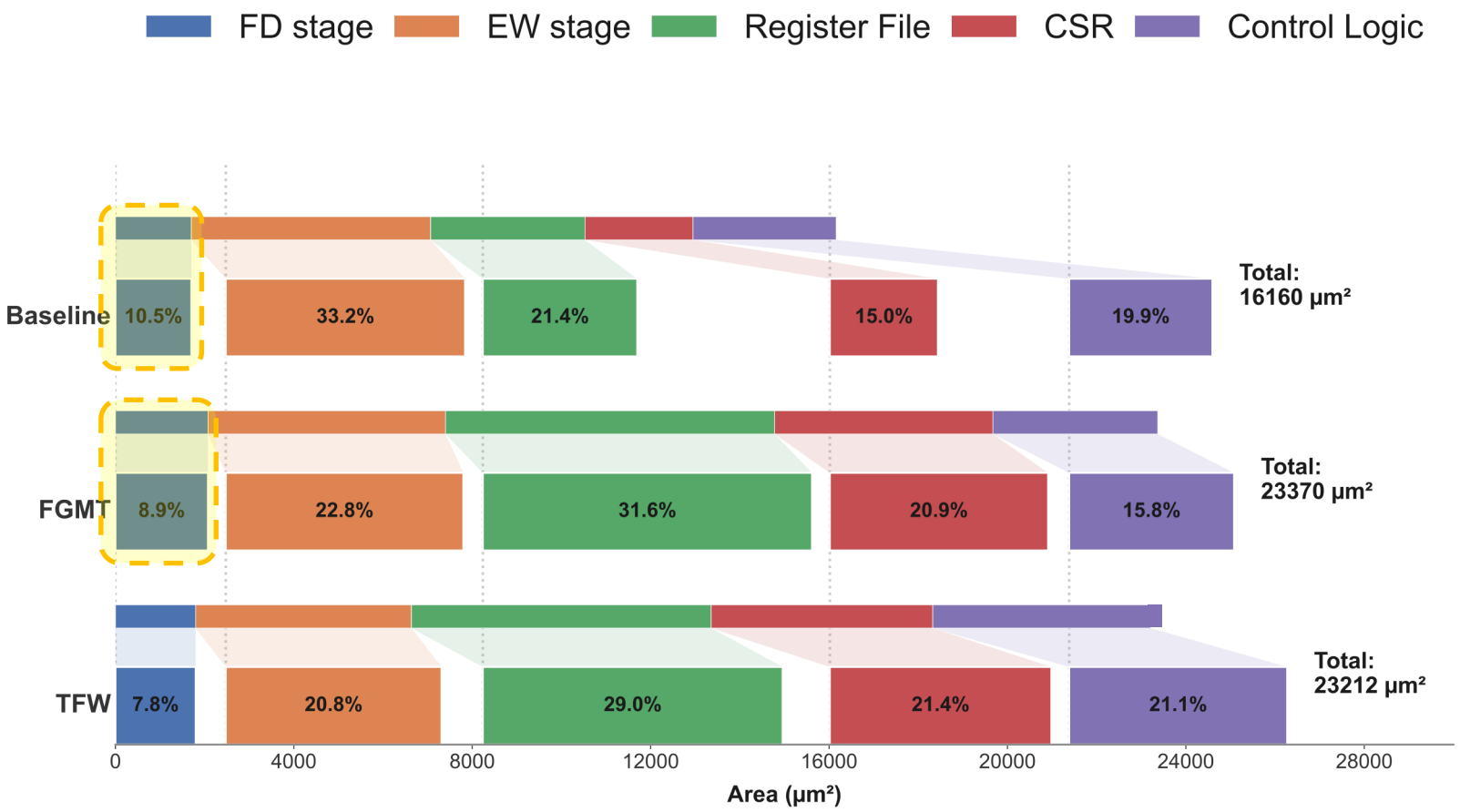
Area



- **FGMT Overhead**
  - *Duplicated **Register File***
  - *Duplicated **CSRs***

- **Place & Route**
- **Frequency:** 300MHz
- **Tech:** C40
- **Corner**: SS, 1.05V, -40°C

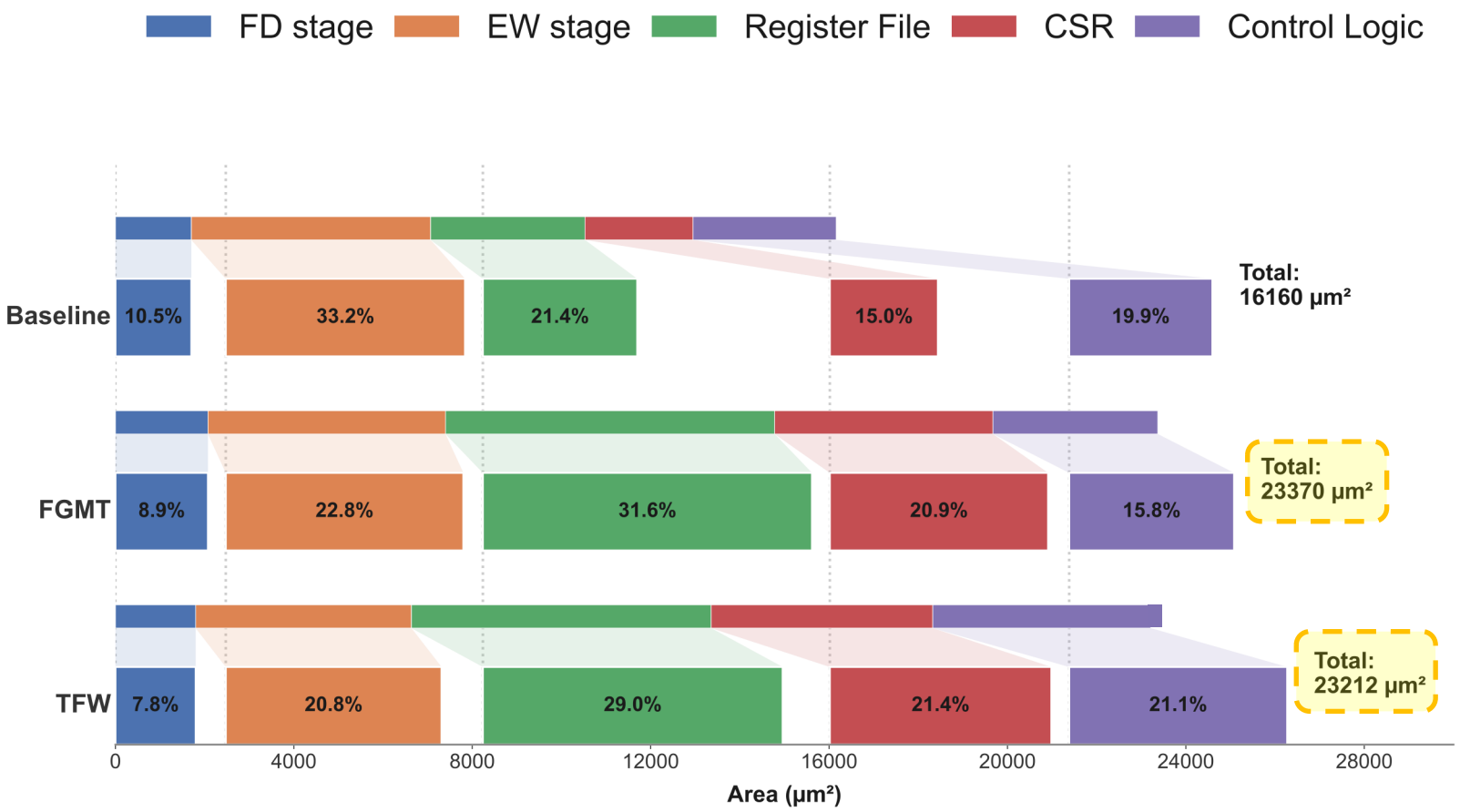
Area



- **Place & Route**
- **Frequency:** 300MHz
- **Tech:** C40
- **Corner**: SS, 1.05V, -40°C

- **FGMT Overhead**
  - *Duplicated **Register File***
  - *Duplicated **CSRs***
- **Minor Area Savings**
  - ***No** Data Dependencies → Removed **data forwarding** lines*
  - ***No** Control Hazards → Simplified **next address** fetch calculation*

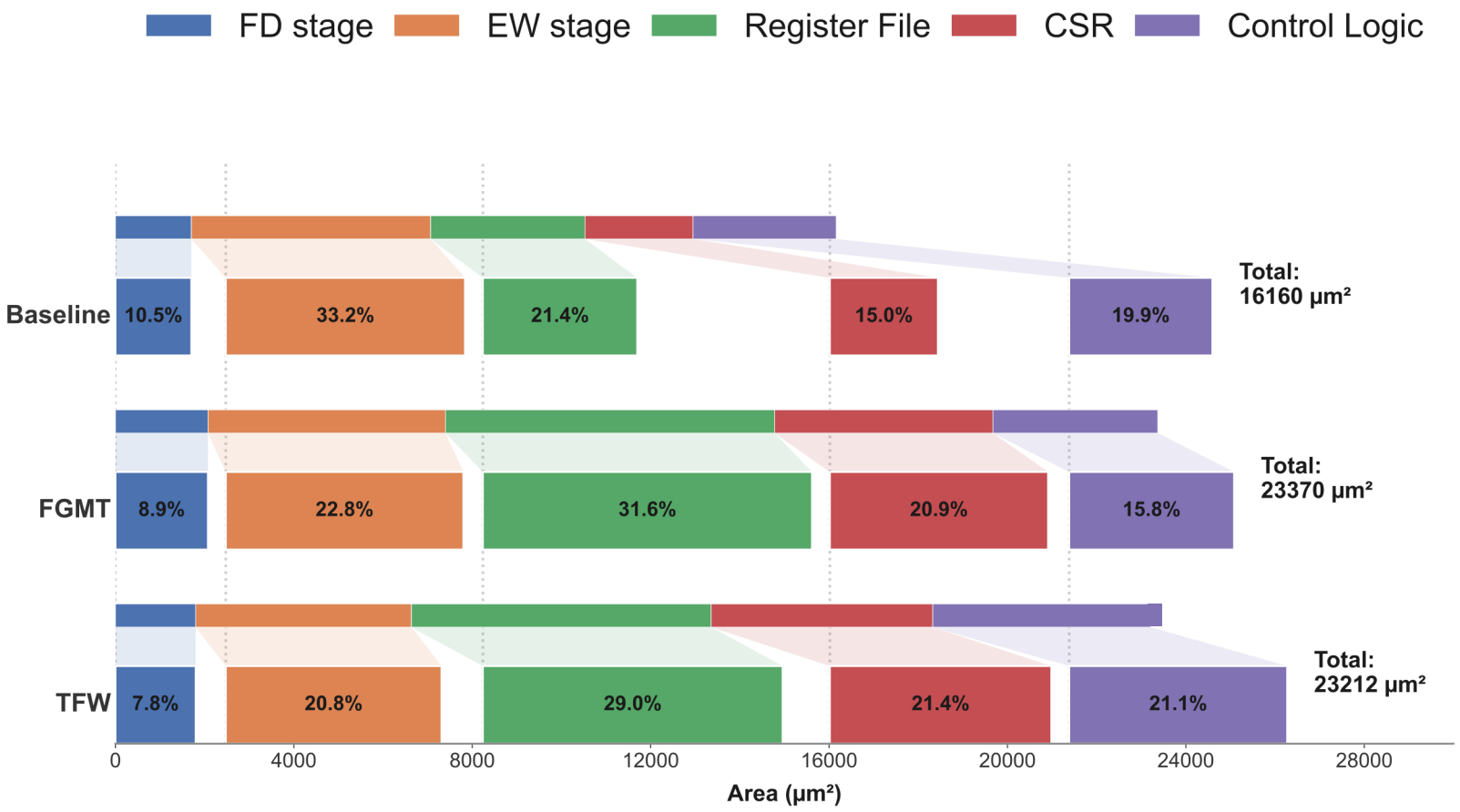
Area



- **Place & Route**
- **Frequency:** 300MHz
- **Tech:** C40
- **Corner**: SS, 1.05V, -40°C

- **FGMT Overhead**
  - *Duplicated **Register File***
  - *Duplicated **CSRs***
- **Minor Area Savings**
  - ***No** Data Dependencies → Removed **data forwarding** lines*
  - ***No** Control Hazards → Simplified **next address** fetch calculation*
- **TFW Overhead Negligible**

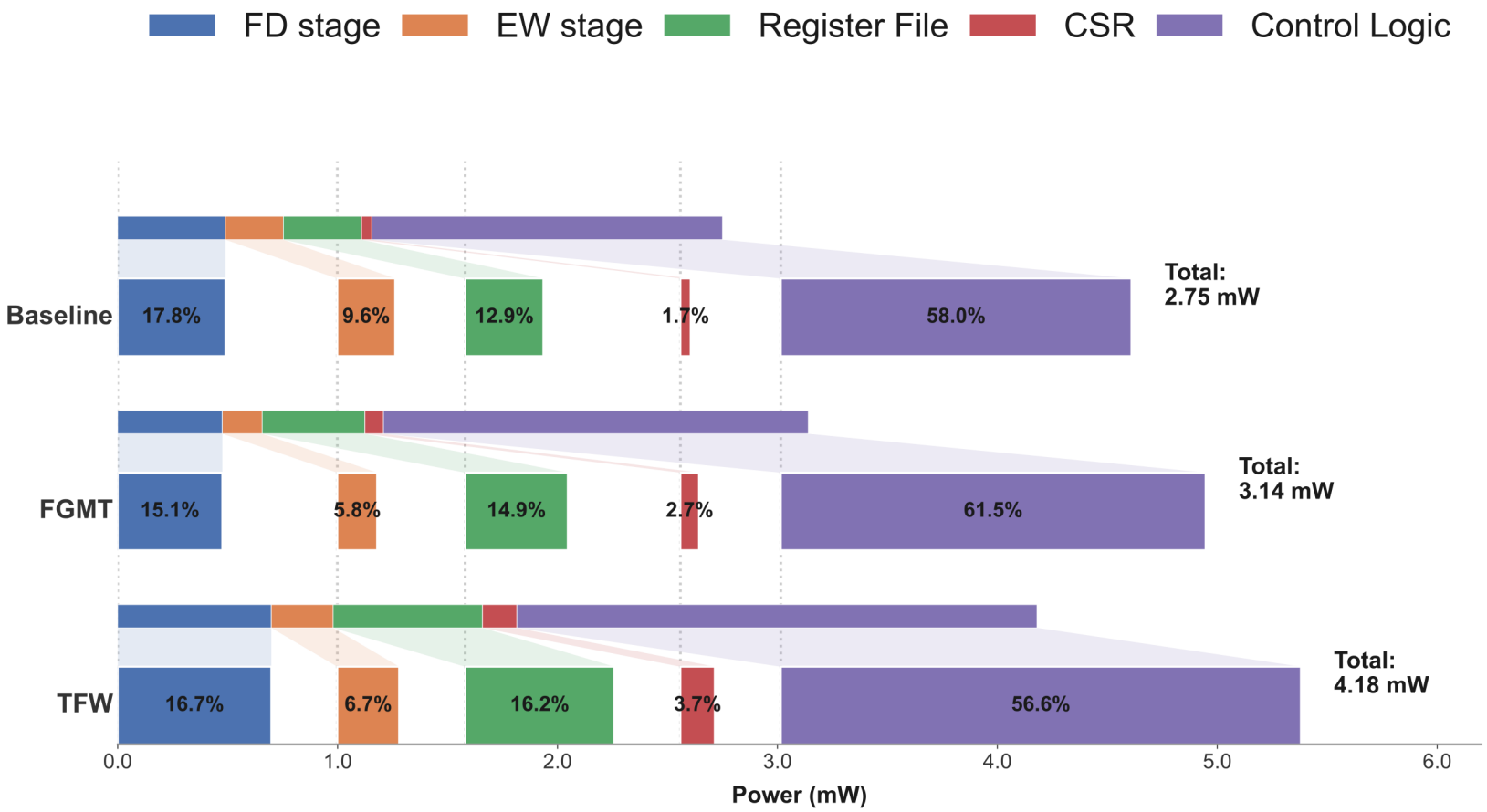
Area



- **Place & Route**
- **Frequency:** 300MHz
- **Tech:** C40
- **Corner**: SS, 1.05V, -40°C

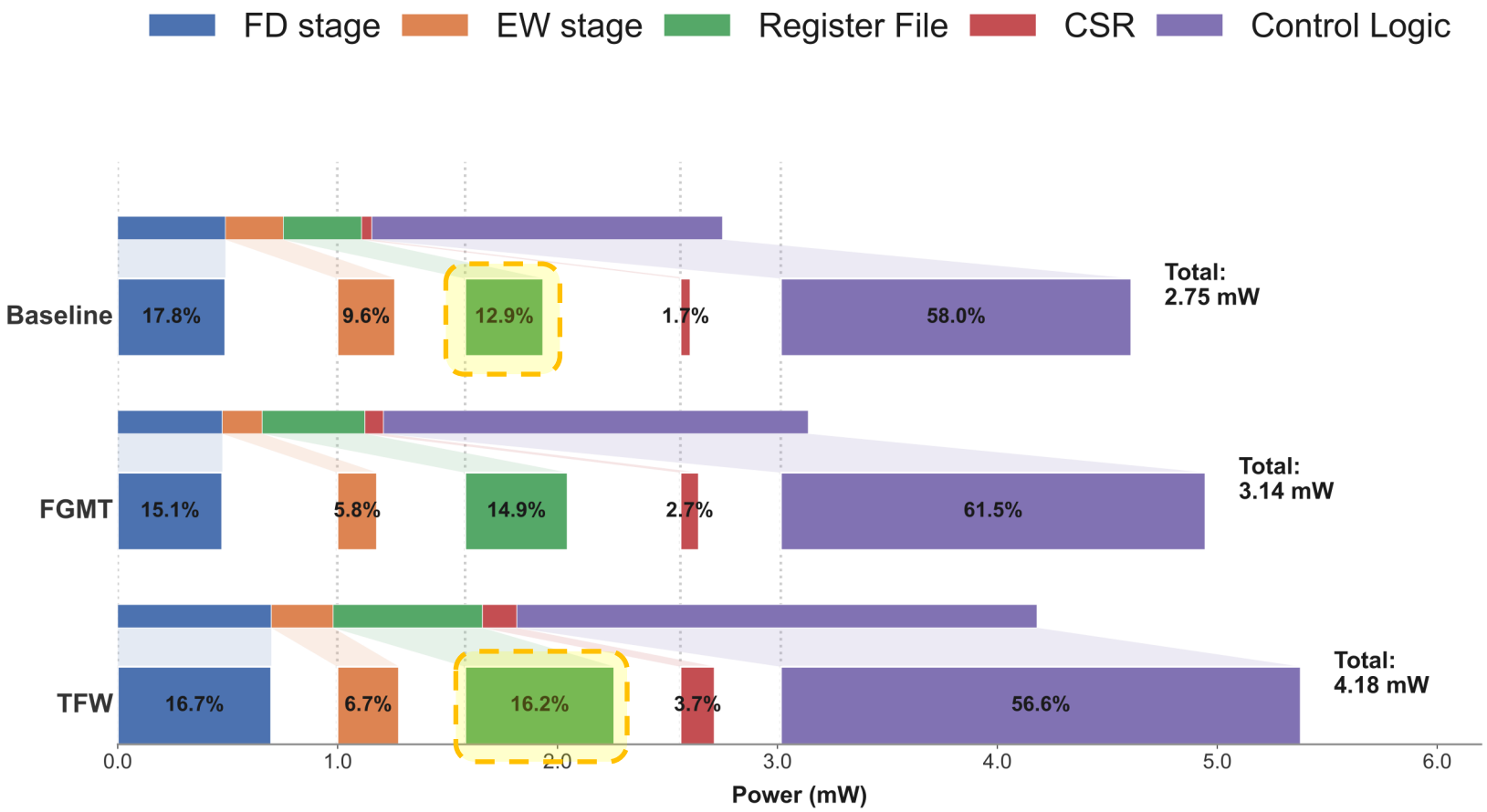
- **FGMT Overhead**
  - *Duplicated **Register File***
  - *Duplicated **CSRs***
- **Minor Area Savings**
  - ***No** Data Dependencies → Removed **data forwarding** lines*
  - ***No** Control Hazards → Simplified **next address** fetch calculation*
- **TFW Overhead Negligible**
- **Overhead** of:
  - Core level: +44%
  - **SoC level: +9%**

Power



- **Vector-driven power signoff**
- **Frequency:** 300MHz
- **Tech:** C40
- **Corner**: TT, 1.10V, 25°C

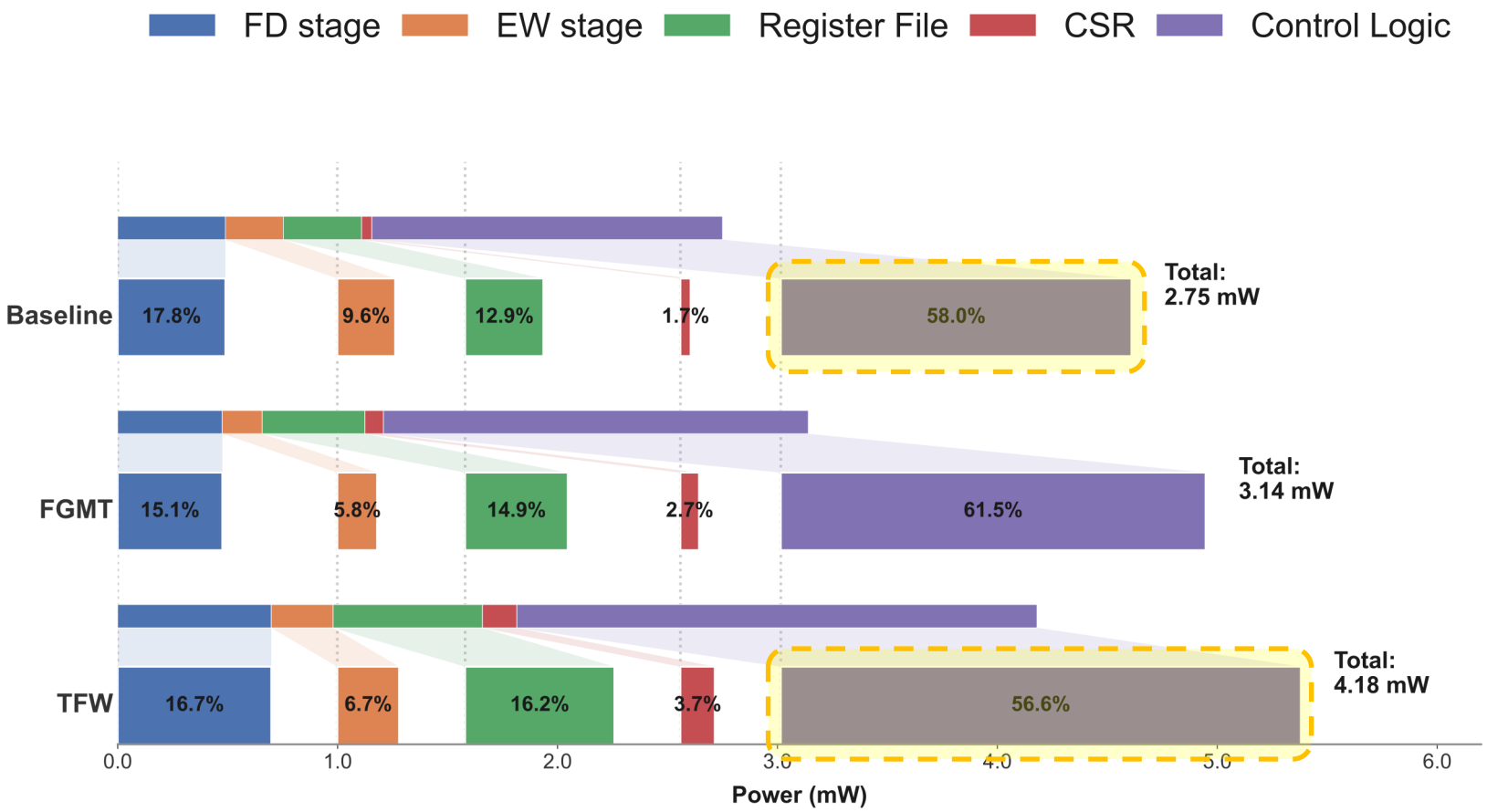
Power



- **FGMT Power Overhead**
  - ***Duplicated*** *RF and CSR*

- **Vector-driven power signoff**
- **Frequency:** 300MHz
- **Tech:** C40
- **Corner**: TT, 1.10V, 25°C

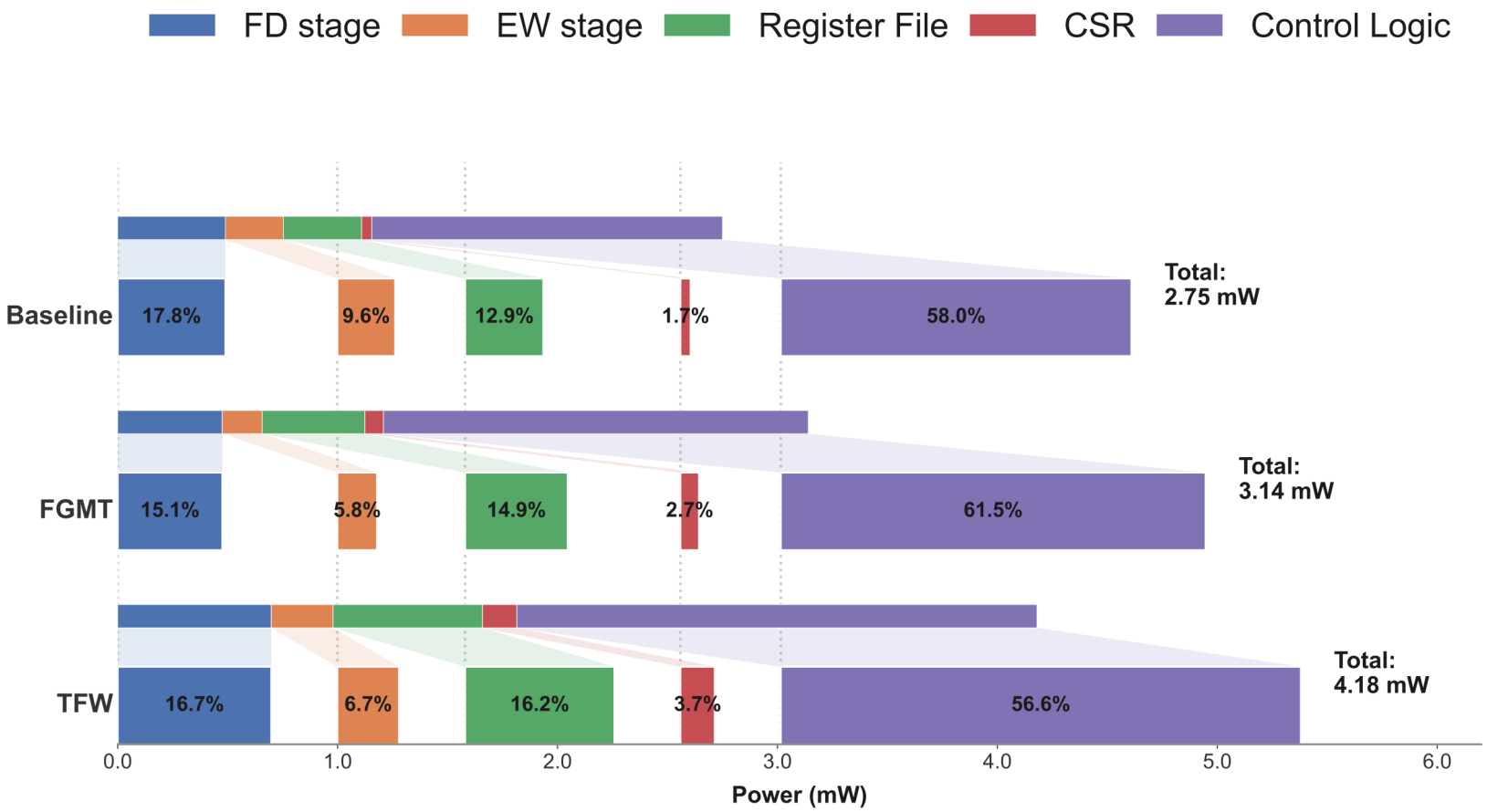
Power



- **FGMT Power Overhead**
  - ***Duplicated*** *RF and CSR*
- **TFW Power**
  - *Higher* ***concurrent utilization*** *of functional units*
  - ***Concurrent*** *write-backs*

- **Vector-driven power signoff**
- **Frequency:** 300MHz
- **Tech:** C40
- **Corner**: TT, 1.10V, 25°C

Power



- **FGMT Power Overhead**
  - ***Duplicated*** *RF and CSR*
- **TFW Power**
  - *Higher* ***concurrent utilization*** *of functional units*
  - ***Concurrent*** *write-backs*
- Overall **overhead** of:
  - **FGMT → +4%**
  - **TFW → +8%**

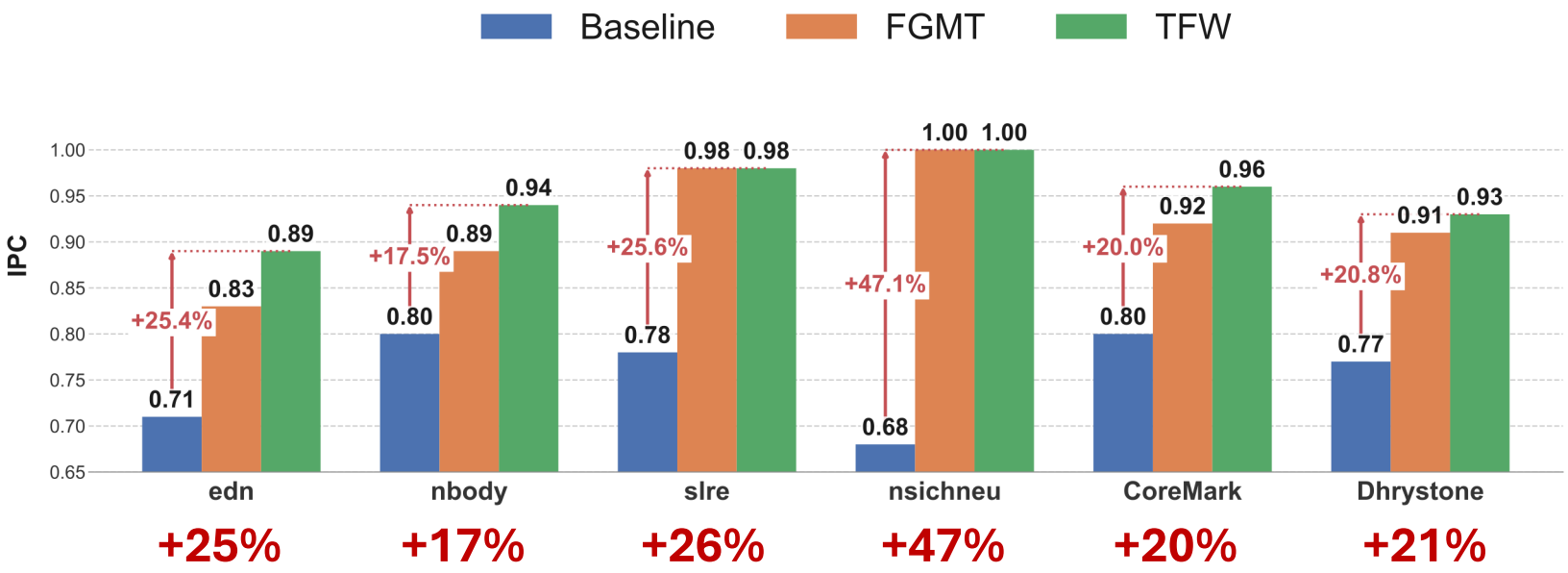
- **Vector-driven power signoff**
- **Frequency:** 300MHz
- **Tech:** C40
- **Corner**: TT, 1.10V, 25°C

Performance

- **Setup**: Multithreaded configurations were tested by running the identical benchmark simultaneously on both hardware threads.

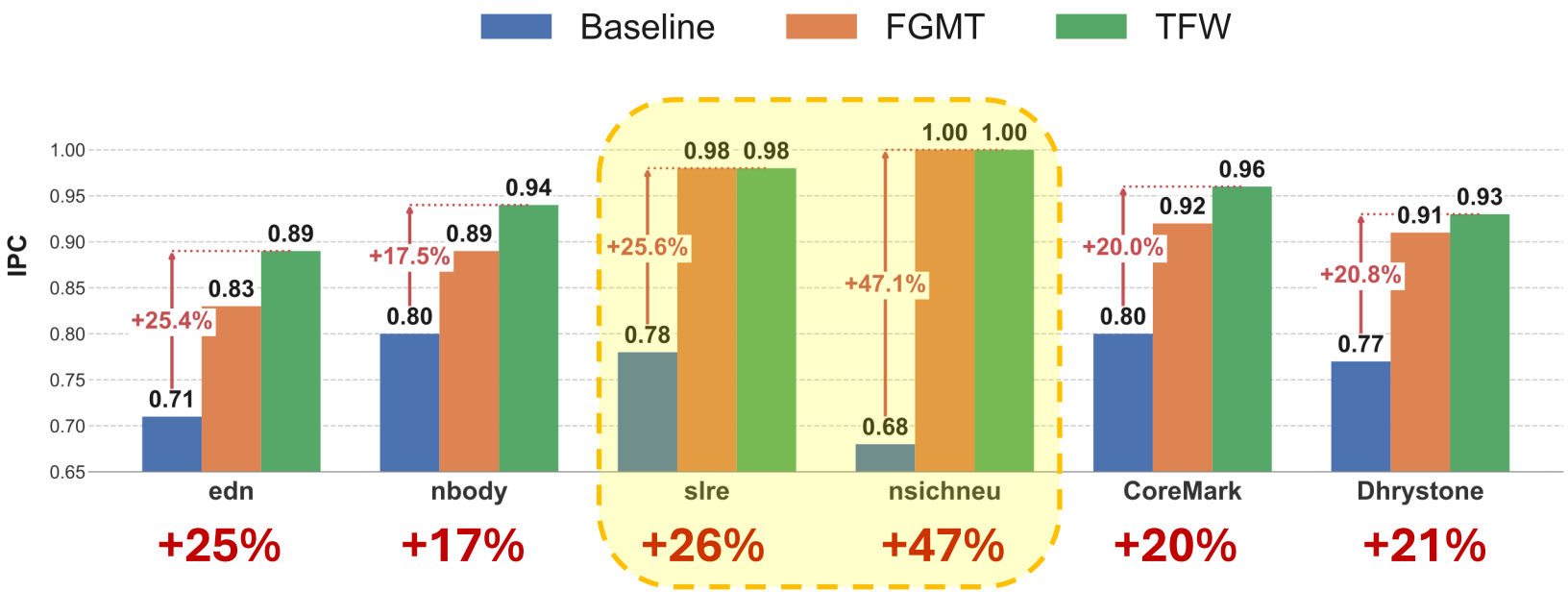
Performance

- **Setup**: Multithreaded configurations were tested by running the identical benchmark simultaneously on both hardware threads.



Performance

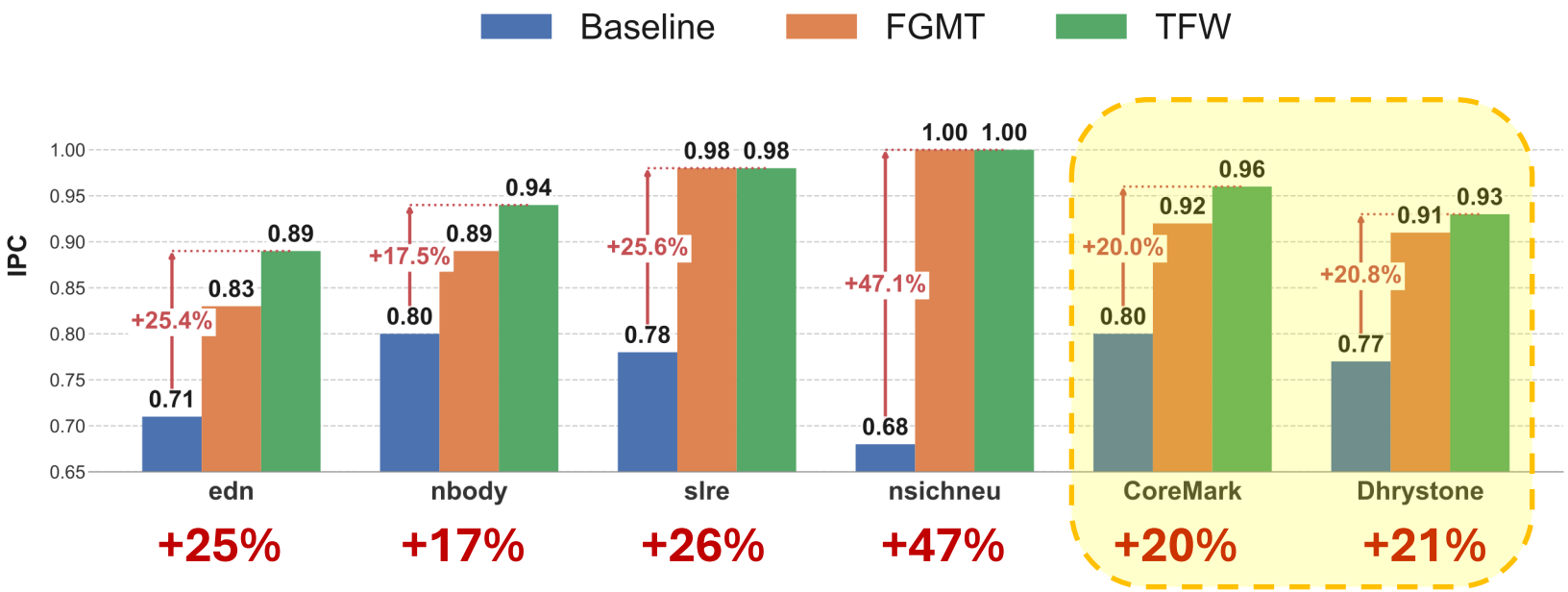
- **Setup**: Multithreaded configurations were tested by running the identical benchmark simultaneously on both hardware threads.



- Most improvement in **control heavy** workloads

Performance

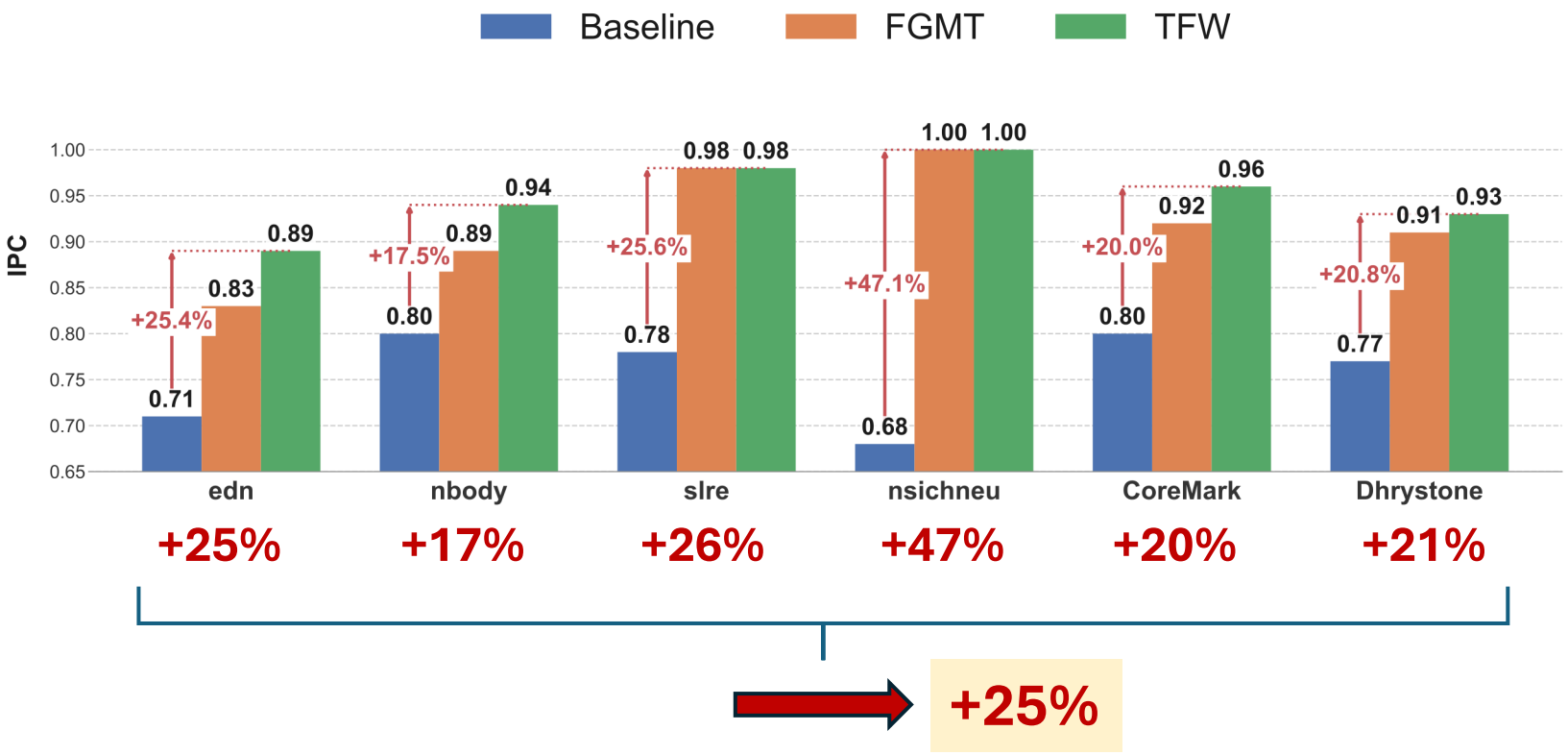
➢ **Setup**: Multithreaded configurations were tested by running the identical benchmark simultaneously on both hardware threads.



- Most improvement in **control heavy** workloads
- Robust improvement in more **comprehensive workloads**

Performance

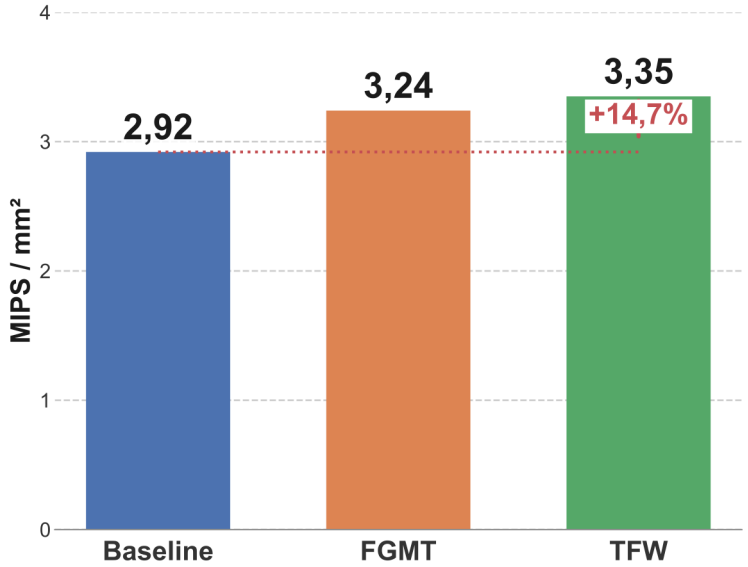
➢ **Setup**: Multithreaded configurations were tested by running the identical benchmark simultaneously on both hardware threads.



- Most improvement in **control heavy** workloads
- Robust improvement in more **comprehensive workloads**
- Mean uplift of **+25%**

Efficiency Metrics

Area Efficiency

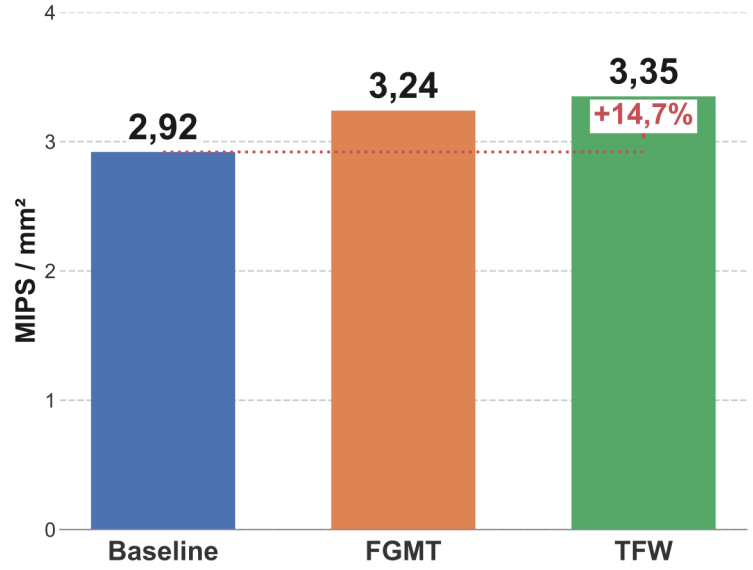


+15%

- **Trade-off:** **+25%** IPC vs. **+9%** SoC Area
- **Peak Model:** TFW → **3.35 MIPS/mm² (+14.7%)**
- **TFW Note:** Achieves **highest throughput without area overhead** compared to FGMT

Efficiency Metrics

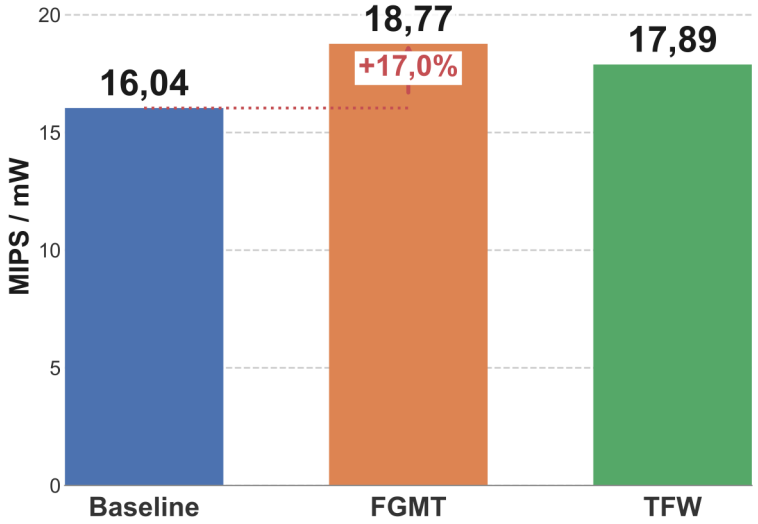
Area Efficiency



+15%

- **Trade-off:** **+25%** IPC vs. **+9%** SoC Area
- **Peak Model:** TFW → **3.35 MIPS/mm² (+14.7%)**
- **TFW Note:** Achieves **highest throughput without area overhead** compared to FGMT

Energy Efficiency



+17%

- **Trade-off:** **+21%** IPC vs. **+4%** SoC Power
- **Peak Model:** FGMT → **18.77 MIPS/mW (+17.0%)**
- **TFW Note:** Drops to +11.6% due to concurrent execution power

Temporal Predictability

Structural Masking of Control Hazards

Temporal Predictability

Structural Masking of Control Hazards

Elimination of Variable Flush Penalties

Removes **data-dependent** pipeline **flushes**:

- ST: CtrlFlow (instr) → 1-2 cycles
- **FGMT: CtrlFlow (instr) → 2 cycles**

Temporal Predictability

Structural Masking of Control Hazards

Elimination of Variable Flush Penalties

Removes **data-dependent** pipeline **flushes**:

- ST: CtrlFlow (instr) → 1-2 cycles
- **FGMT: CtrlFlow (instr) → 2 cycles**

Improved Temporal Predictability

Temporal Predictability

Structural Masking of Control Hazards

Elimination of Variable Flush Penalties

Removes **data-dependent** pipeline **flushes**:

- ST: CtrlFlow (instr) → 1-2 cycles
- **FGMT: CtrlFlow (instr) → 2 cycles**

Improved Temporal Predictability

✅ Setup

- **Benchmark:** slre
- Varying input data
- **NormalizedET** $(\frac{ET}{ACET})$

Temporal Predictability

Structural Masking of Control Hazards

Elimination of Variable Flush Penalties

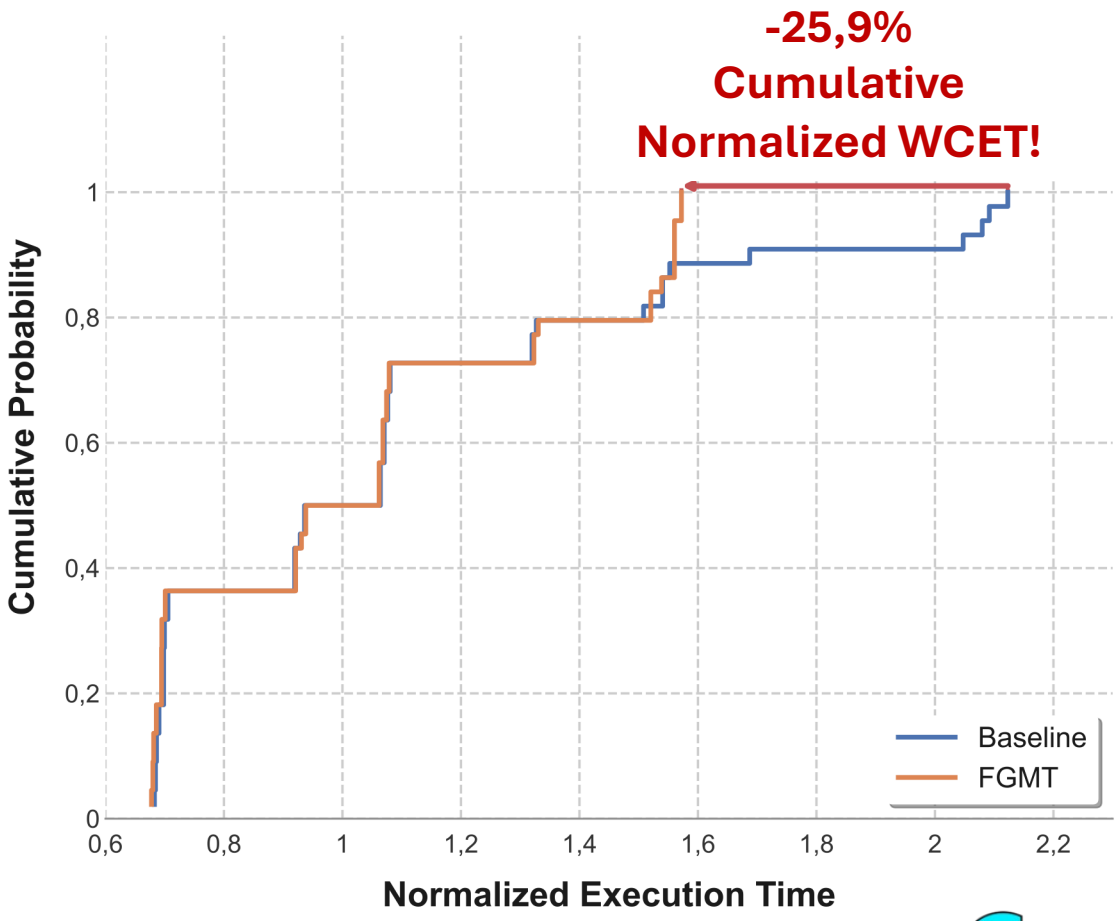
Removes **data-dependent** pipeline **flushes**:

- ST: CtrlFlow (instr) → 1-2 cycles
- **FGMT: CtrlFlow (instr) → 2 cycles**

Improved Temporal Predictability

✅ Setup

- **Benchmark:** slre
- Varying input data
- **NormalizedET** $(\frac{ET}{ACET})$



Temporal Predictability

Structural Masking of Control Hazards

Elimination of Variable Flush Penalties

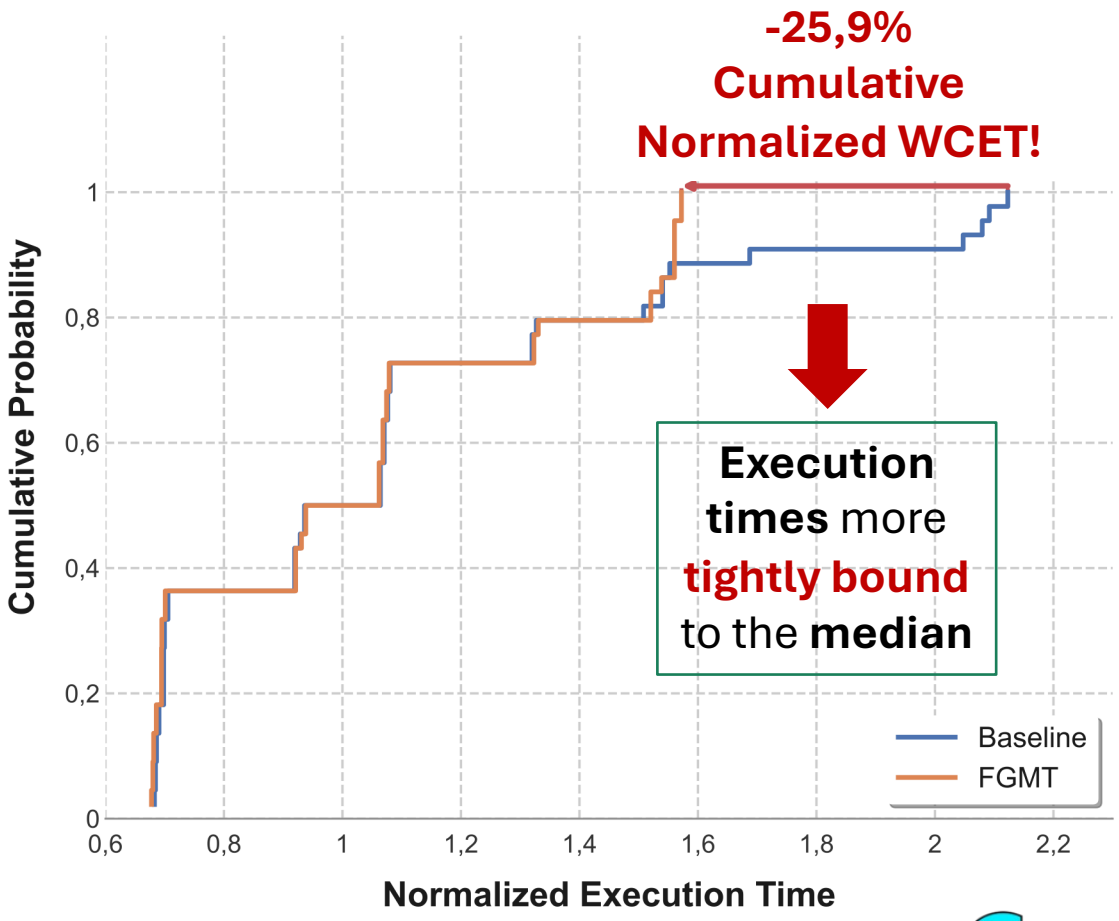
Removes **data-dependent** pipeline **flushes**:

- ST: CtrlFlow (instr) → 1-2 cycles
- **FGMT: CtrlFlow (instr) → 2 cycles**

Improved Temporal Predictability

✅ Setup

- **Benchmark:** slre
- Varying input data
- **NormalizedET** $(\frac{ET}{ACET})$



Conclusion

We introduce an enhanced **32-bit RISC-V core (RV32ECM)** supporting **Fine-Grained Multi-Threading (FGMT)** and **Thread Forwarding (TFW)** for real-time embedded systems.

Conclusion

We introduce an enhanced **32-bit RISC-V core (RV32ECM)** supporting **Fine-Grained Multi-Threading (FGMT)** and **Thread Forwarding (TFW)** for real-time embedded systems.

- **Standard FGMT:** Intrinsically **masks control hazards**, delivering a **21% IPC improvement**

Conclusion

We introduce an enhanced **32-bit RISC-V core (RV32ECM)** supporting **Fine-Grained Multi-Threading (FGMT)** and **Thread Forwarding (TFW)** for real-time embedded systems.

- **Standard FGMT:** Intrinsically **masks control hazards**, delivering a **21% IPC improvement**
- **New TFW mechanism:** Bypasses multi-cycle stalls by overlapping **concurrent thread execution**, boosting cumulative **IPC by 25%** over the single-threaded baseline.

Conclusion

We introduce an enhanced **32-bit RISC-V core (RV32ECM)** supporting **Fine-Grained Multi-Threading (FGMT)** and **Thread Forwarding (TFW)** for real-time embedded systems.

- **Standard FGMT:** Intrinsically **masks control hazards**, delivering a **21% IPC improvement**
- **New TFW mechanism:** Bypasses multi-cycle stalls by overlapping **concurrent thread execution**, boosting cumulative **IPC by 25%** over the single-threaded baseline.
- **Zero added area footprint:** TFW maintains the exact same **9% area overhead** as standard FGMT through structural optimizations.

Conclusion

We introduce an enhanced **32-bit RISC-V core (RV32ECM)** supporting **Fine-Grained Multi-Threading (FGMT)** and **Thread Forwarding (TFW)** for real-time embedded systems.

- **Standard FGMT:** Intrinsically **masks control hazards**, delivering a **21% IPC improvement**
- **New TFW mechanism:** Bypasses multi-cycle stalls by overlapping **concurrent thread execution**, boosting cumulative **IPC by 25%** over the single-threaded baseline.
- **Zero added area footprint:** TFW maintains the exact same **9% area overhead** as standard FGMT through structural optimizations.
- **Improved predictability**

Conclusion

We introduce an enhanced **32-bit RISC-V core (RV32ECM)** supporting **Fine-Grained Multi-Threading (FGMT)** and **Thread Forwarding (TFW)** for real-time embedded systems.

- **Standard FGMT:** Intrinsically **masks control hazards**, delivering a **21% IPC improvement**
- **New TFW mechanism:** Bypasses multi-cycle stalls by overlapping **concurrent thread execution**, boosting cumulative **IPC by 25%** over the single-threaded baseline.
- **Zero added area footprint:** TFW maintains the exact same **9% area overhead** as standard FGMT through structural optimizations.
- **Improved predictability**
- **Enhanced System Efficiency:** The architecture achieves **+14.7% area efficiency** (MIPS/mm^2) and up to **+17% energy efficiency** (MIPS/mW).


Thank you!
Questions?