Improving the Performance Per Area Factor of RISC-V Based Multi-Core Systems

Tobias Strauch
R&D, EDAptix, Munich, Germany
tobias@edaptix.com

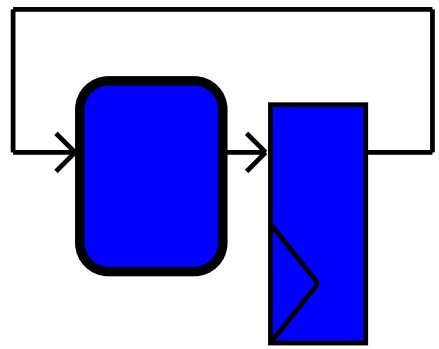
4th RISC-V Workshop at MIT, Boston, USA

July 13, 2016

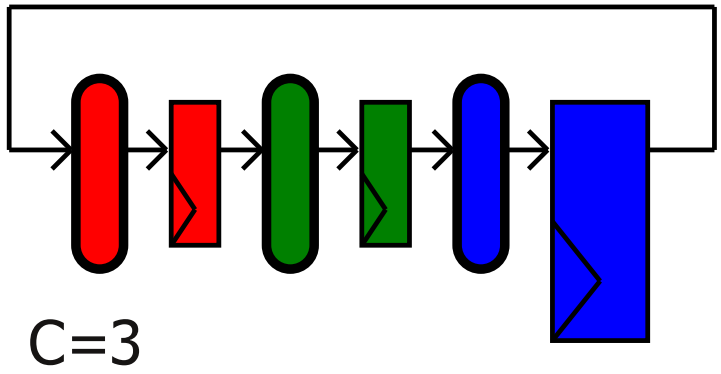
Agenda

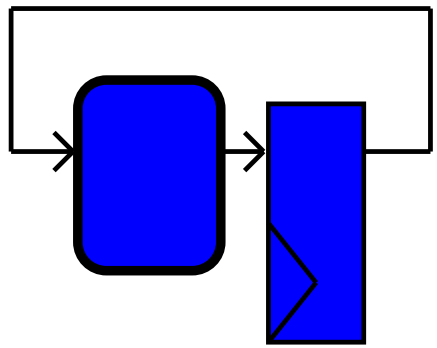
- C-Slow Retiming
- System Hyper Pipelining
  - Basic Idea
  - Performance Balancing
  - Deep Pipelining
  - Extended Pipelining
  - Performance per Area Factor
- microRISC Project
- miniRISC Project
- Miscellaneous

C-Slow Retiming

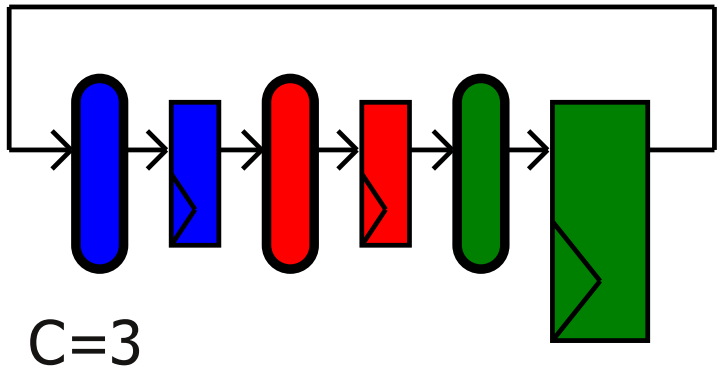


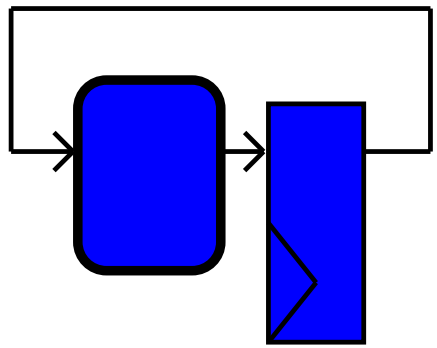
C-Slow Retiming



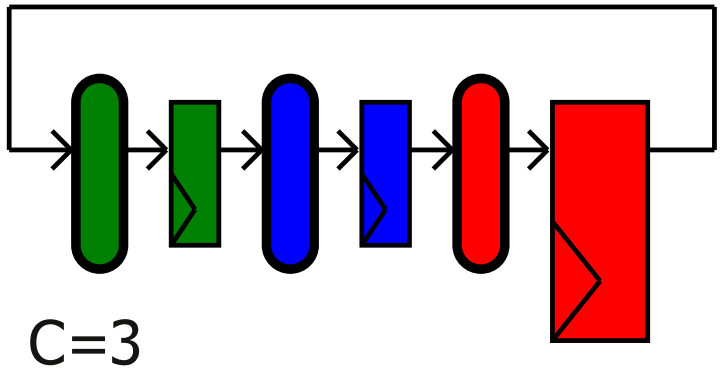


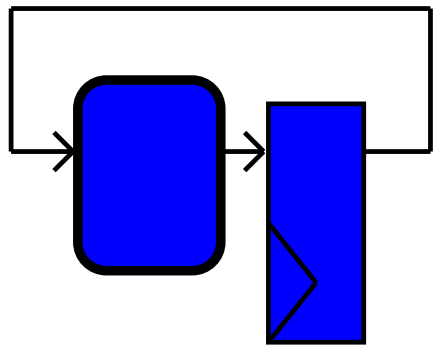
C-Slow Retiming



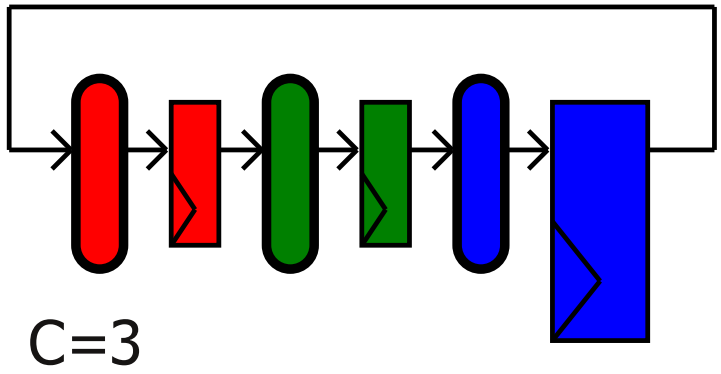


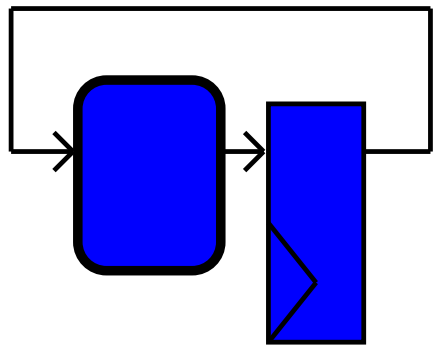
C-Slow Retiming



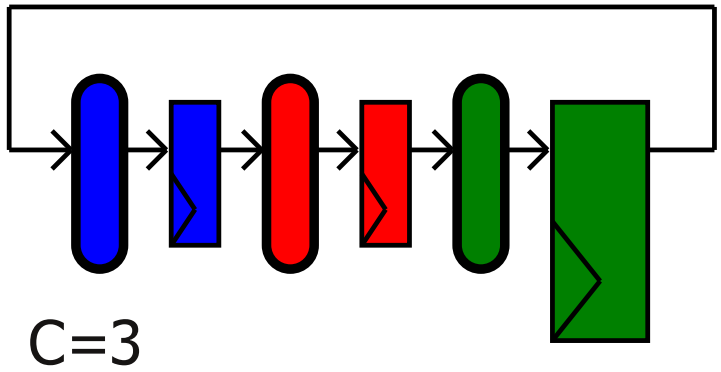


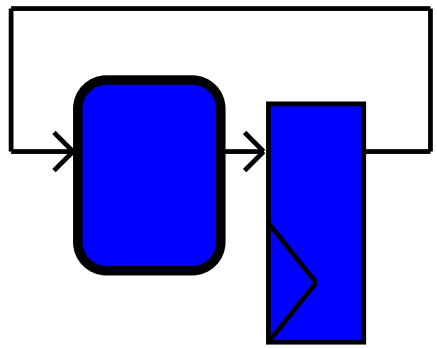
C-Slow Retiming



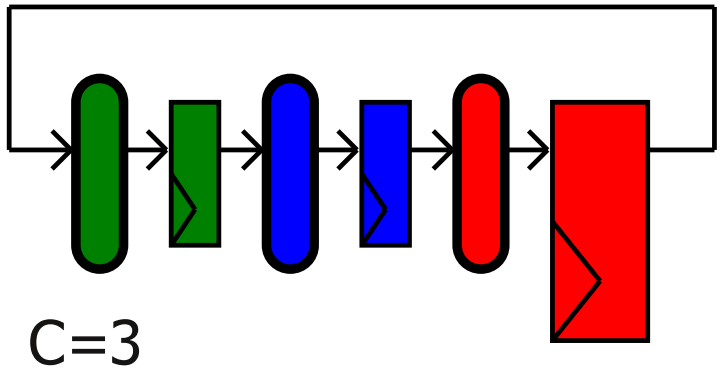


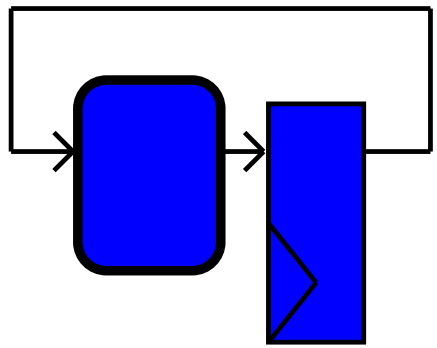
C-Slow Retiming



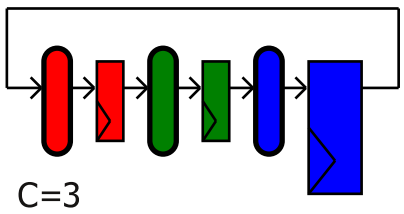


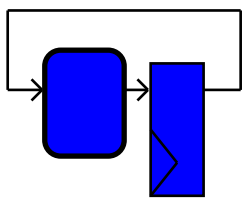
C-Slow Retiming

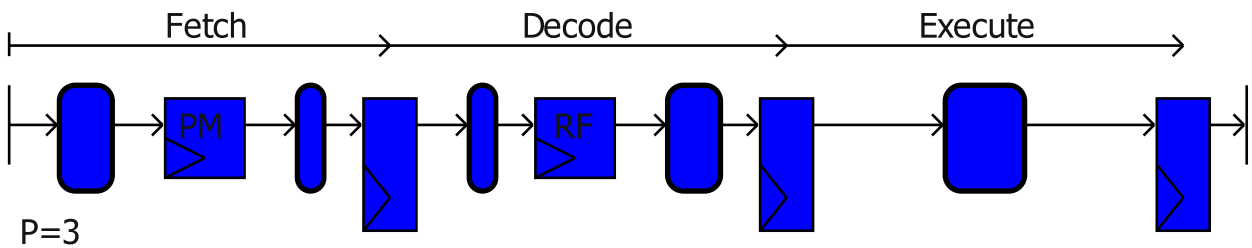




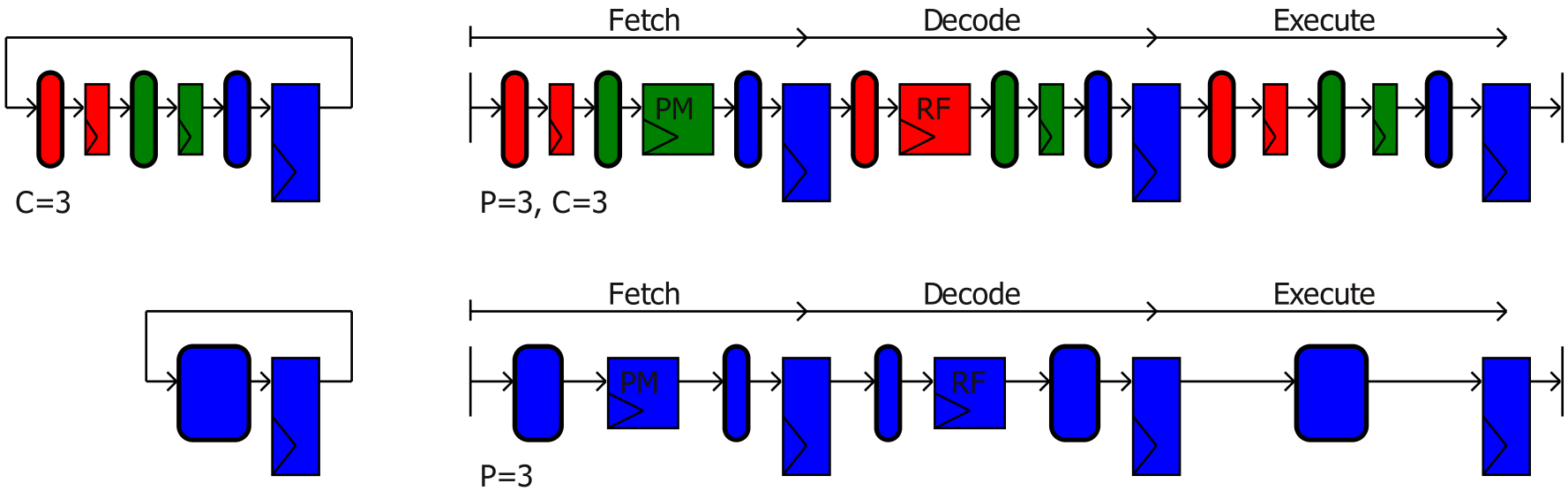
C-Slow Retiming



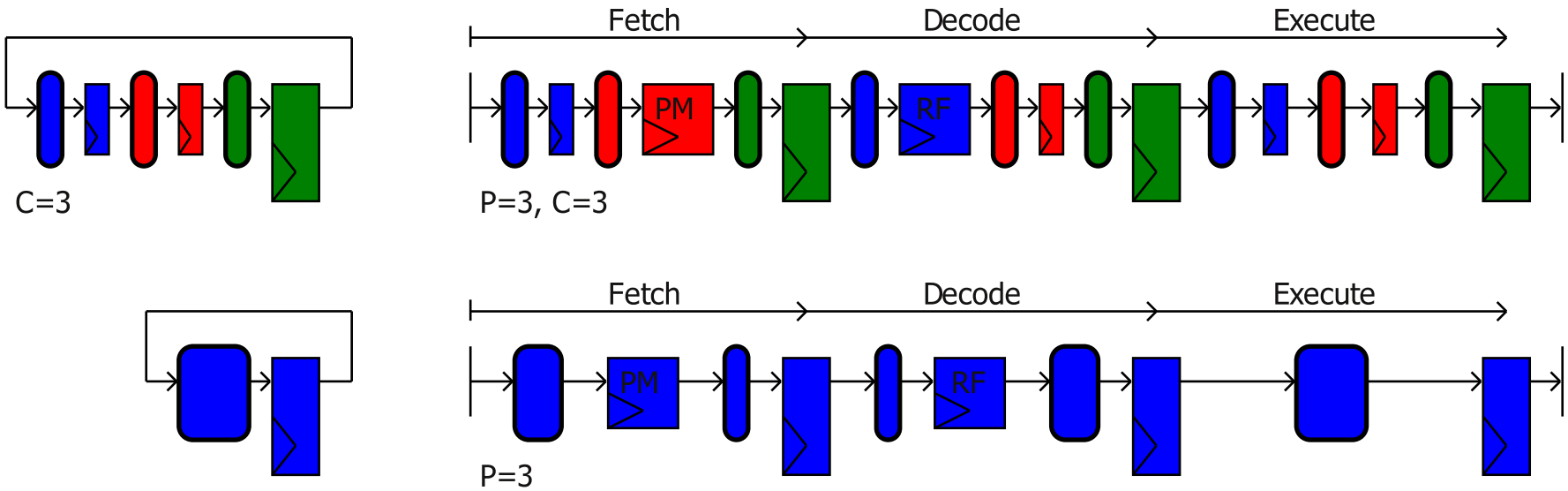




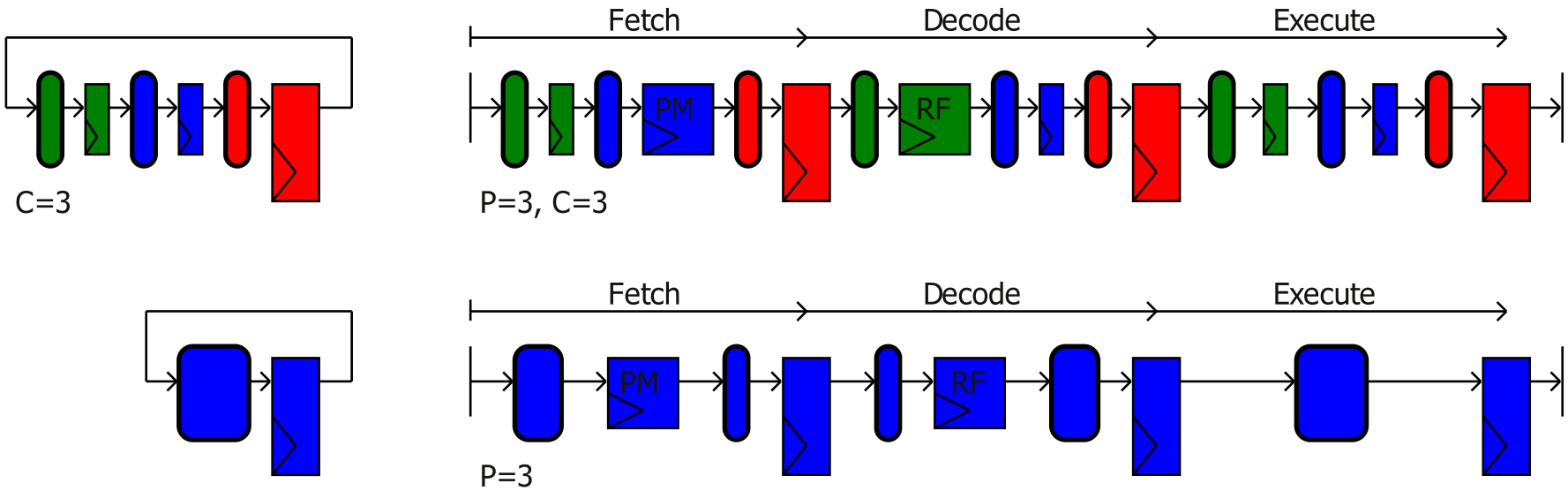
C-Slow Retiming



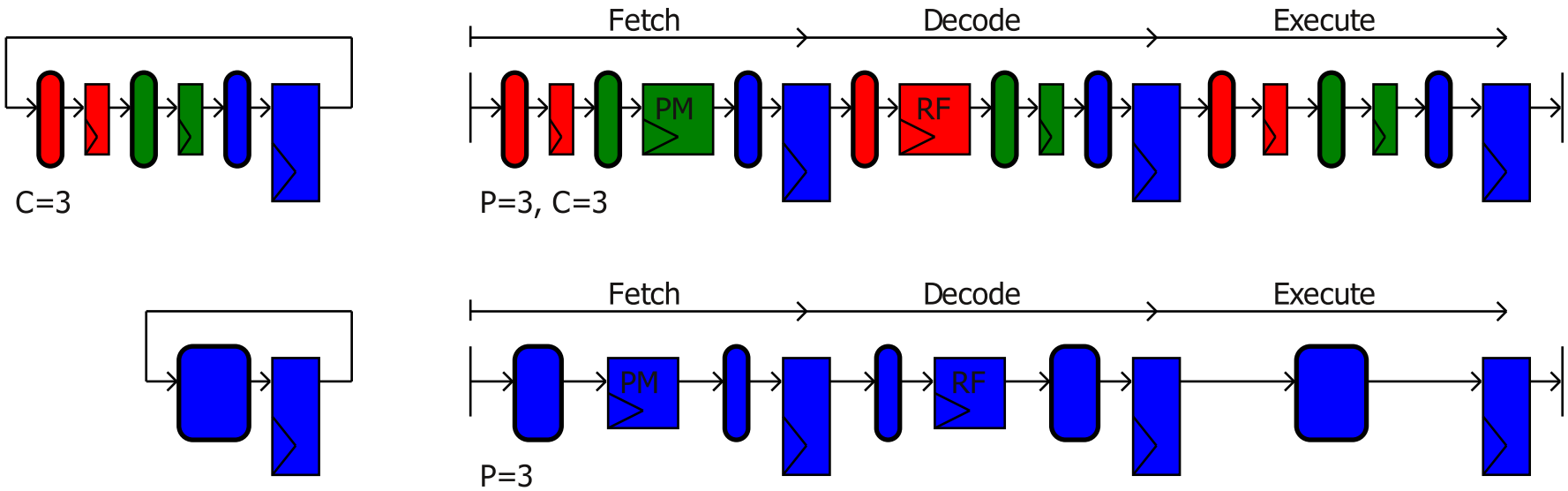
C-Slow Retiming



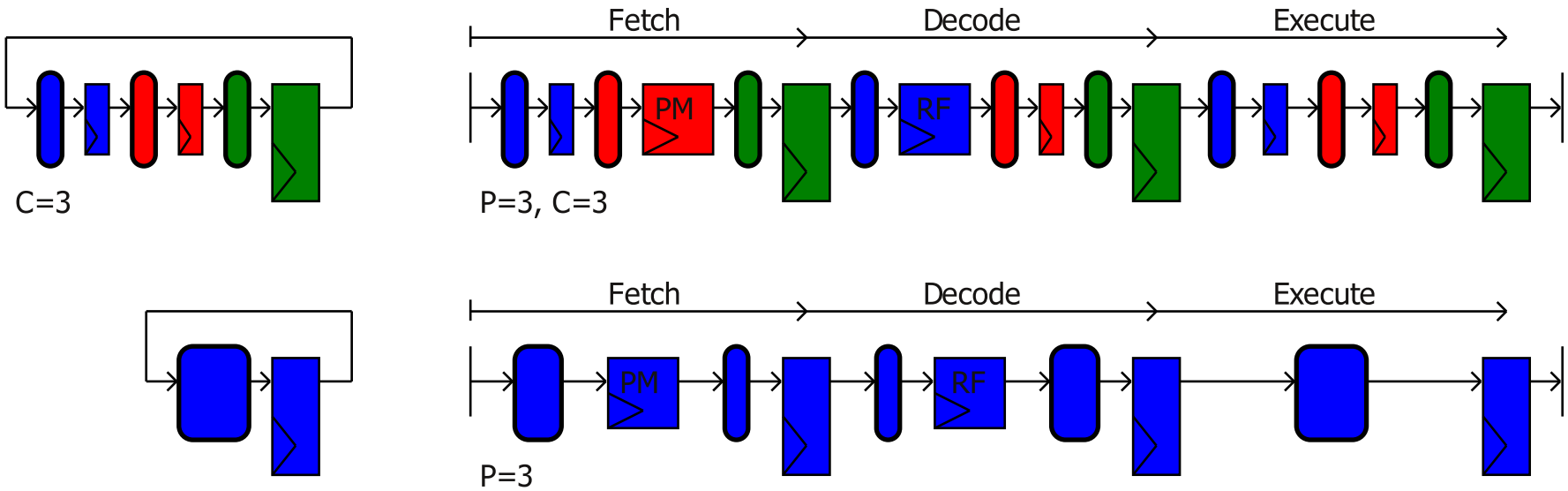
C-Slow Retiming



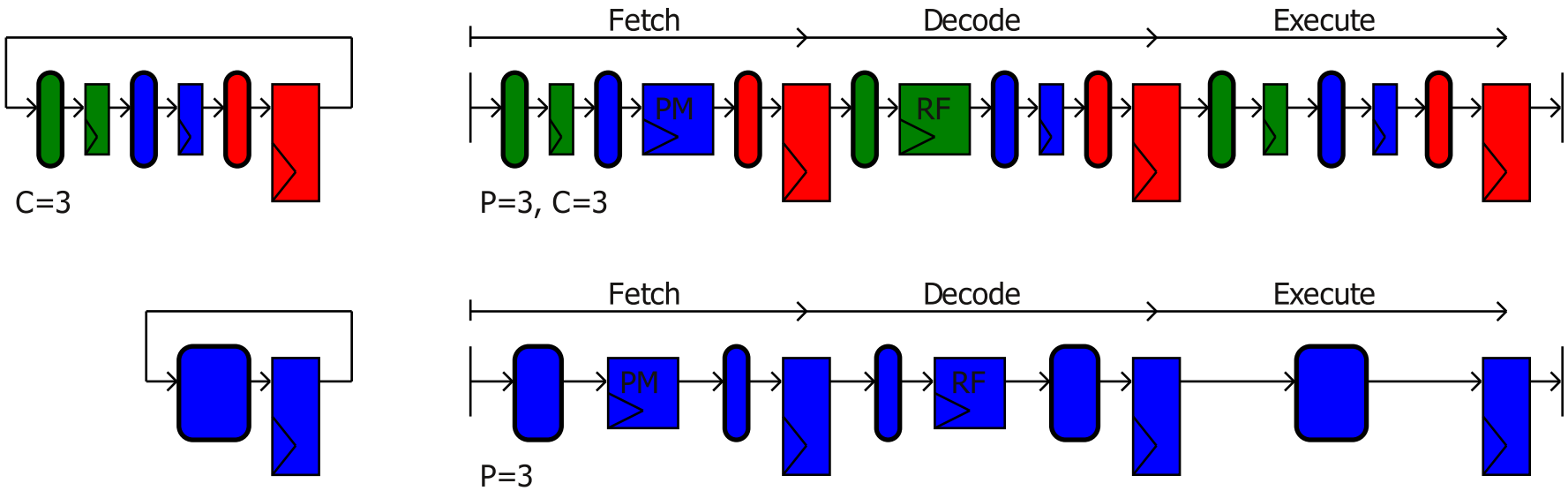
C-Slow Retiming



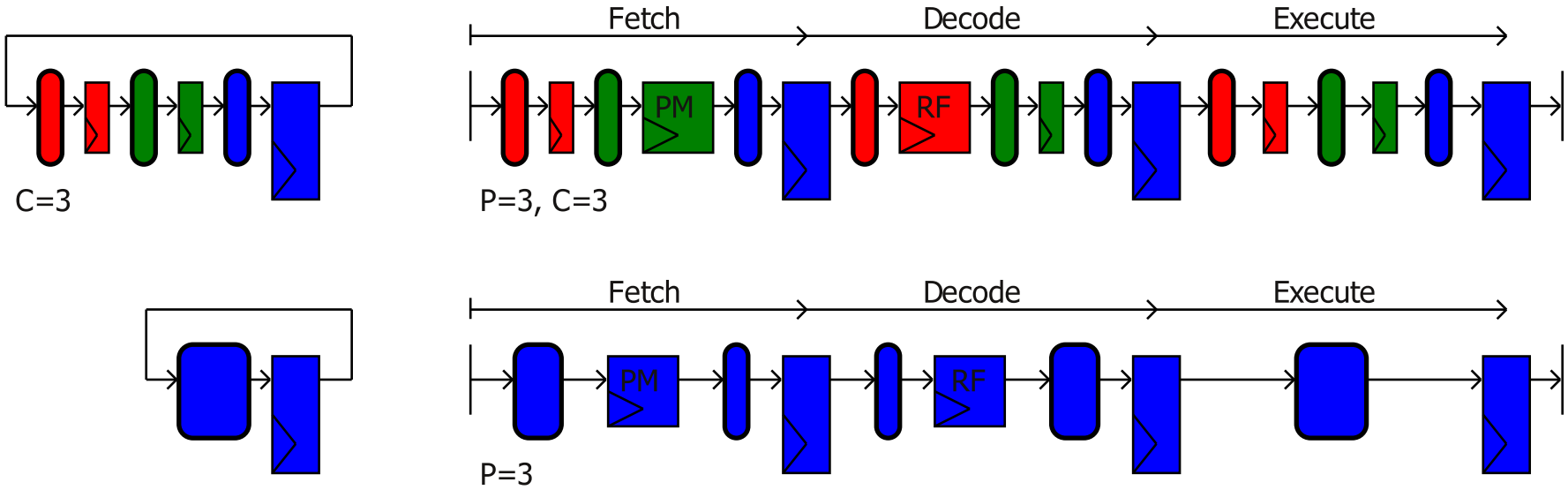
C-Slow Retiming



C-Slow Retiming



C-Slow Retiming

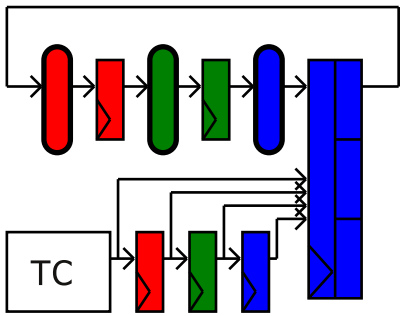


- Known since the 60's, Barrel processors
- Leiserson [1]
- Berkeley summer class in 2001
- CSR on FPGAs (LUT level) by Weaver et al. [2]
- Millions of engineers

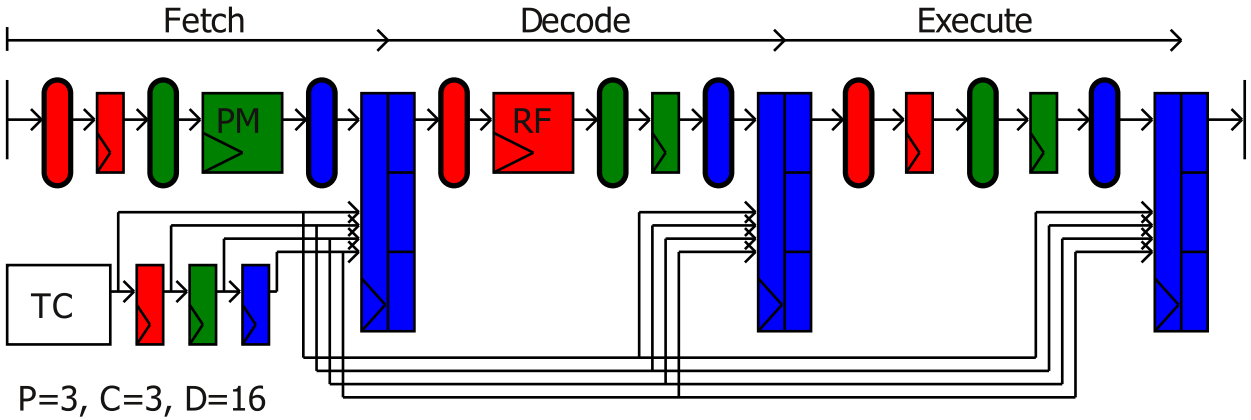
C-Slow Retiming (Personal Work)

- Diploma Thesis in 1998 on CSR of FSM
- LSI Logic (Milpitas, CA), RISC Processor, CSR on gate level
- 2001: Timing estimation on RTL, RTL code modification, ...
- 2010: Automatically apply CSR on RTL
- 2010: AVR Core (VHDL) on opencores.org
- 2010: OpenRISC Core (Verilog) on opencores.org
- Papers on CSR on RTL, CSR in safety critical designs, ...

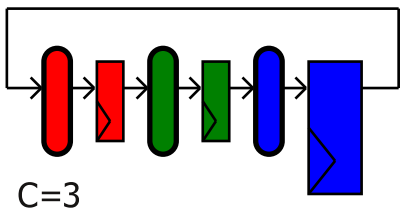
System Hyper Pipelining



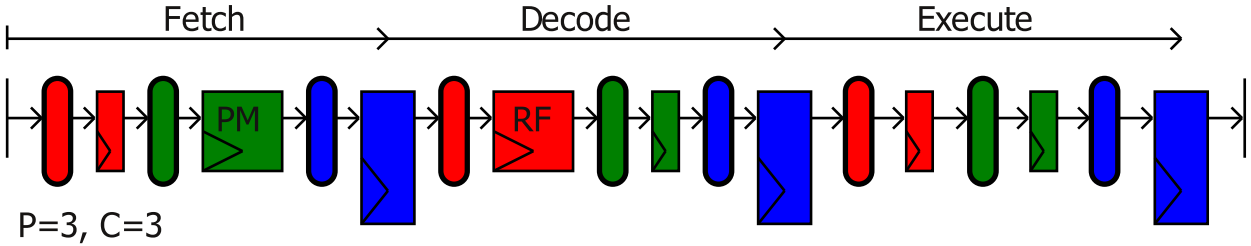
C=3, D=16



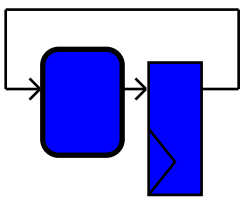
P=3, C=3, D=16

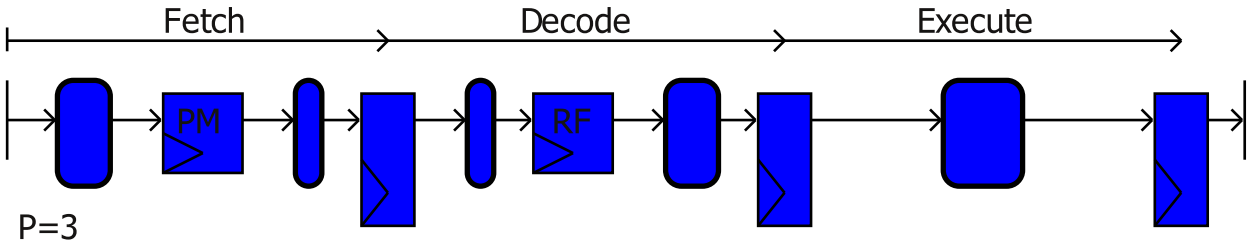


C=3



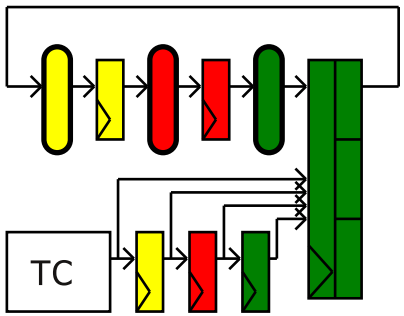
P=3, C=3



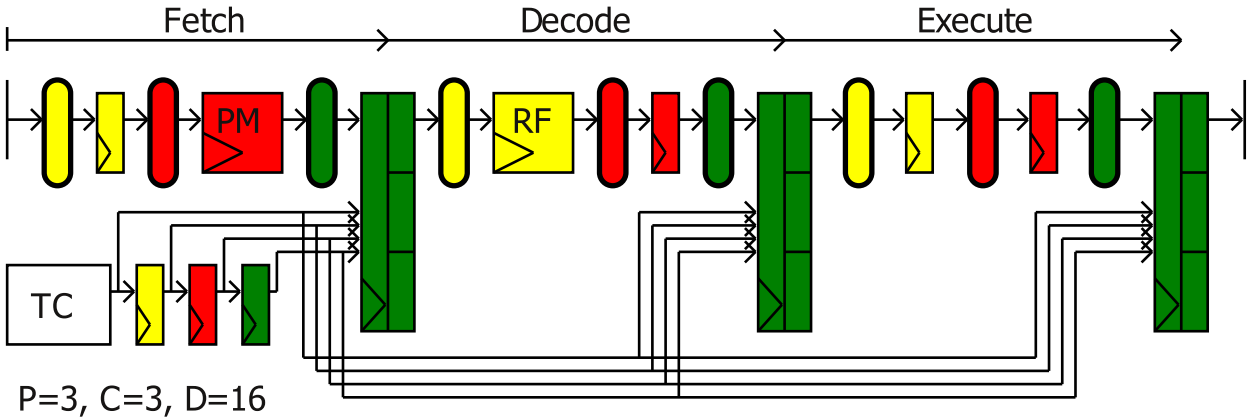


P=3

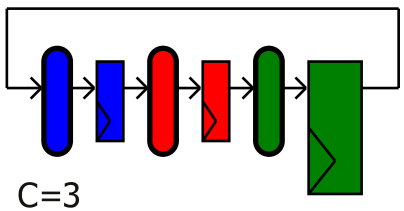
System Hyper Pipelining



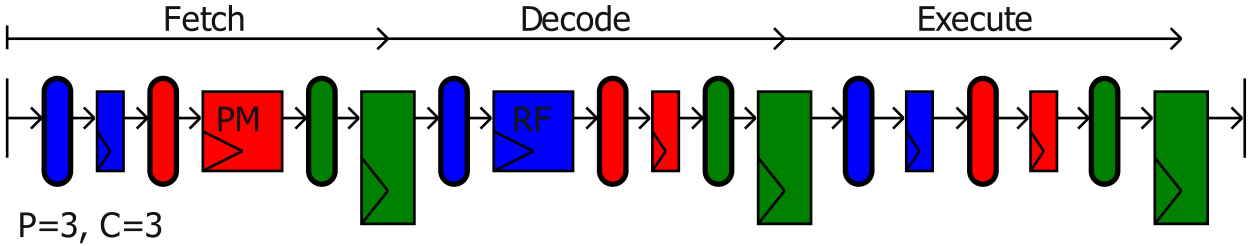
C=3, D=16



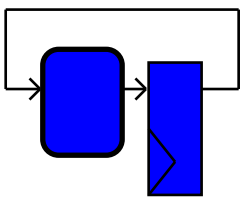
P=3, C=3, D=16

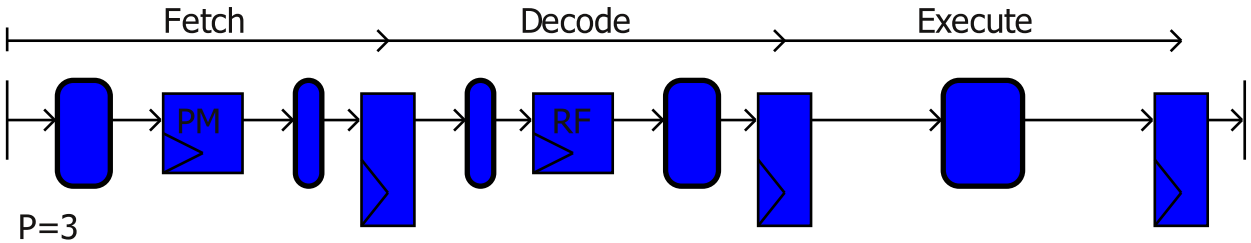


C=3



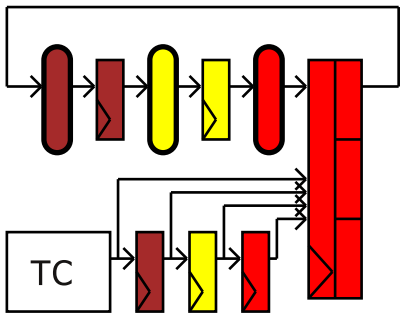
P=3, C=3



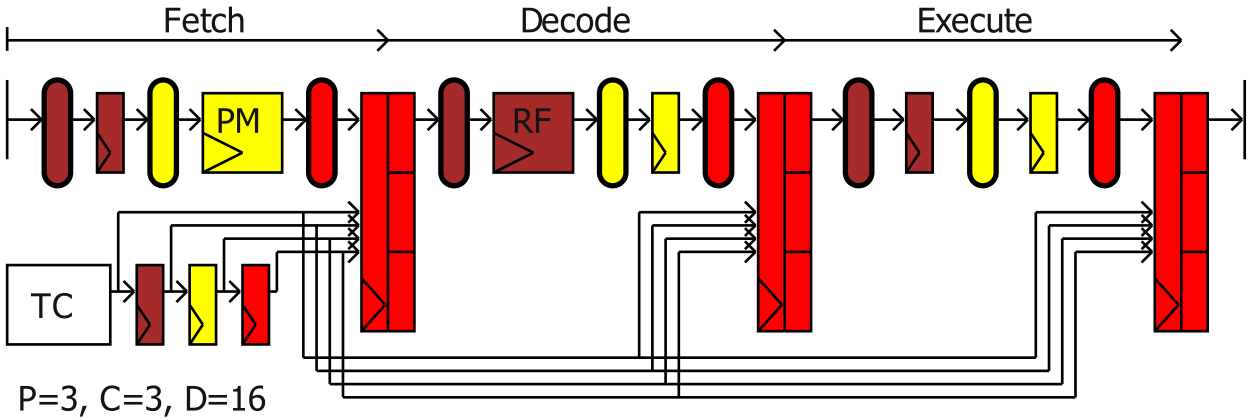


P=3

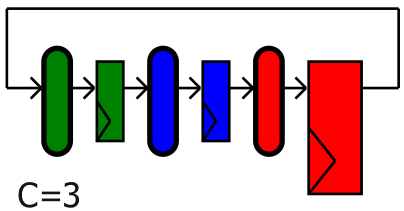
System Hyper Pipelining



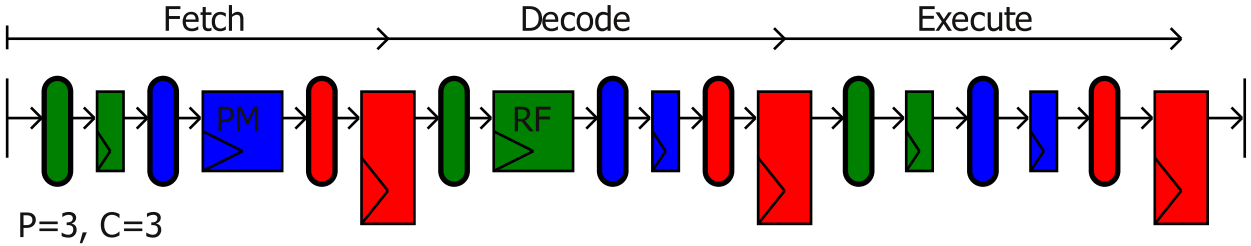
C=3, D=16



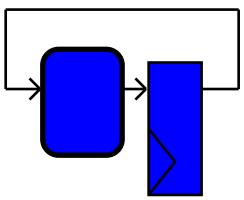
P=3, C=3, D=16

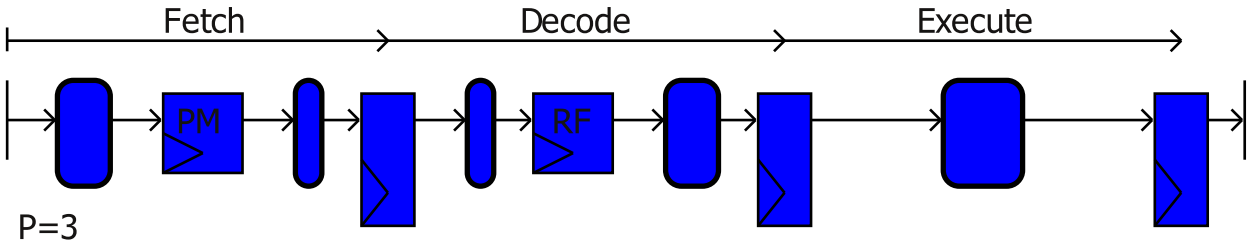


C=3



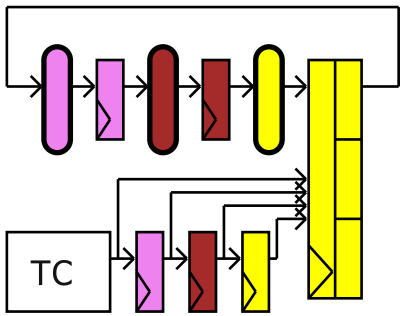
P=3, C=3



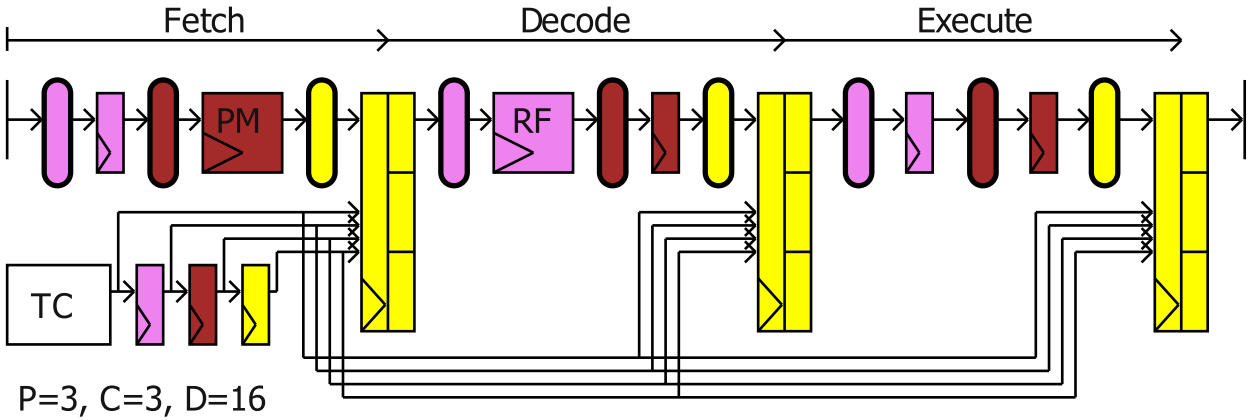


P=3

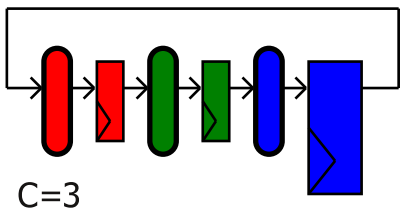
System Hyper Pipelining



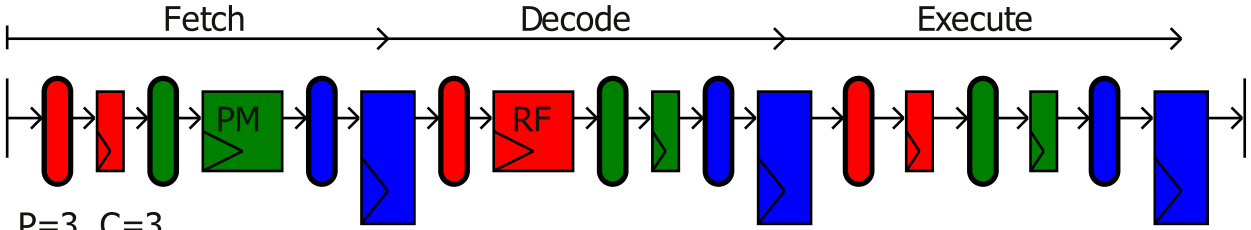
C=3, D=16



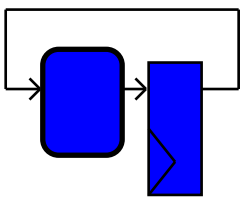
P=3, C=3, D=16

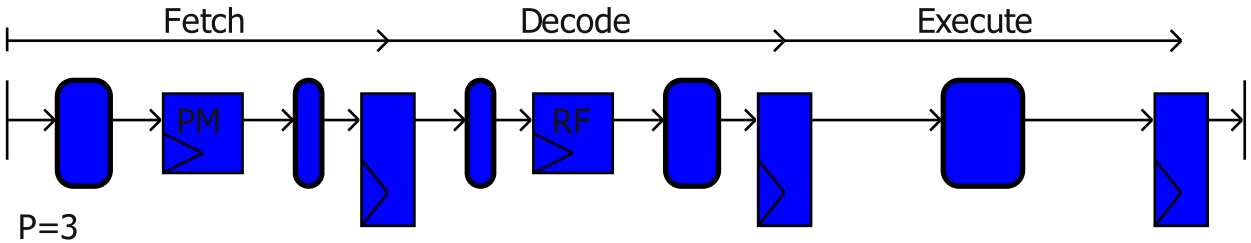


C=3



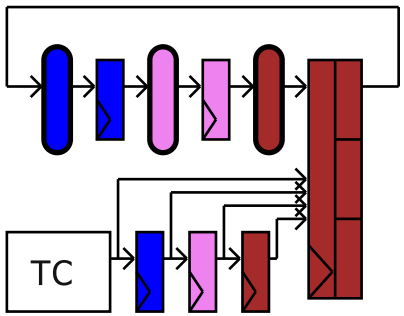
P=3, C=3



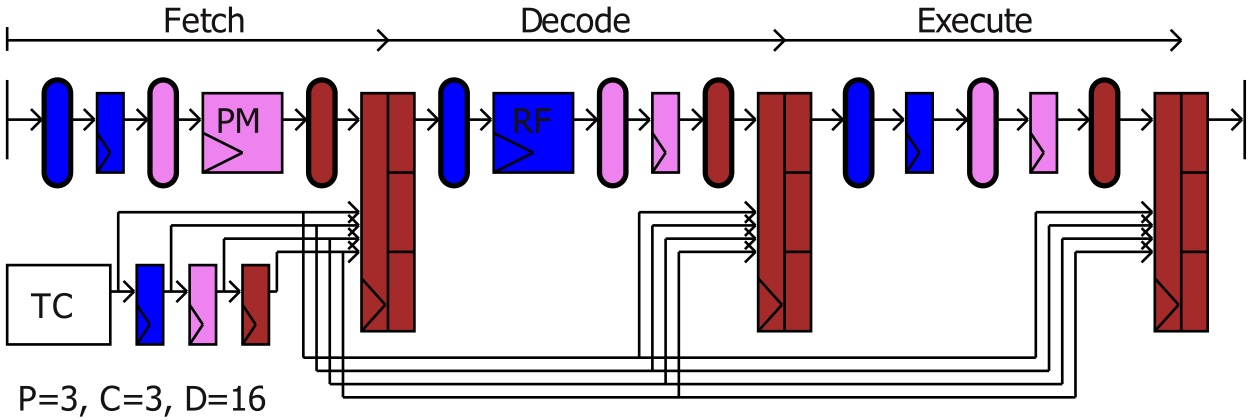


P=3

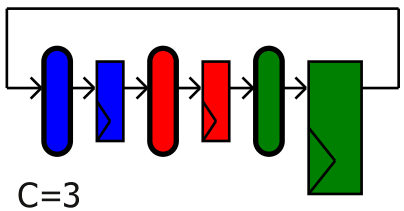
System Hyper Pipelining



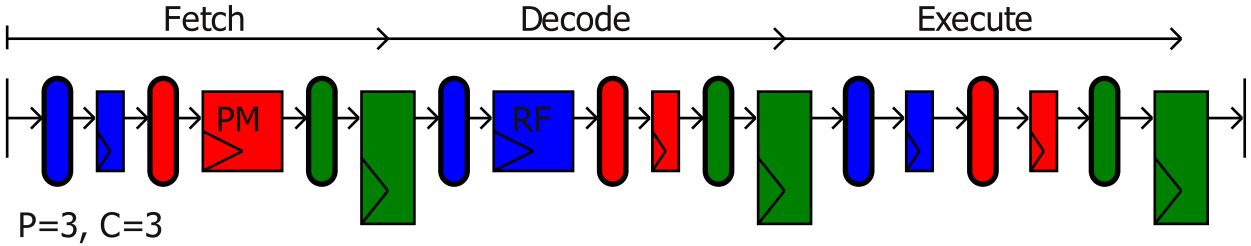
C=3, D=16



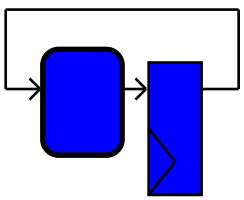
P=3, C=3, D=16

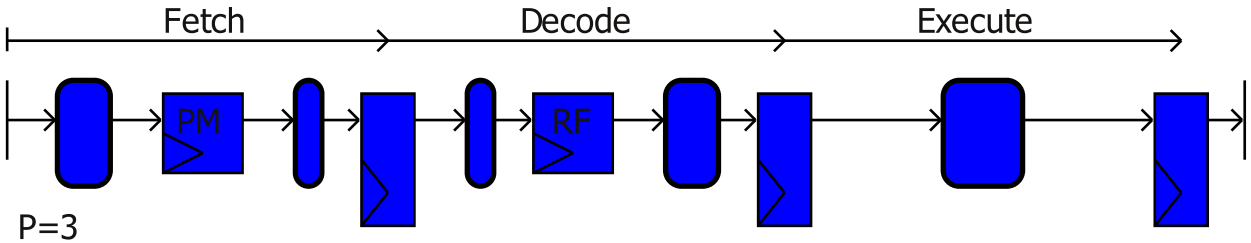


C=3



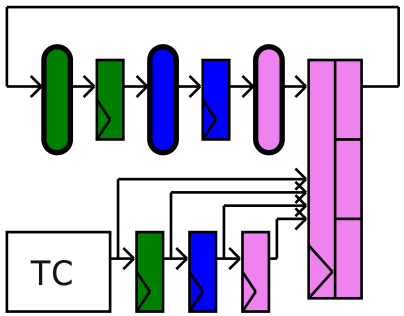
P=3, C=3



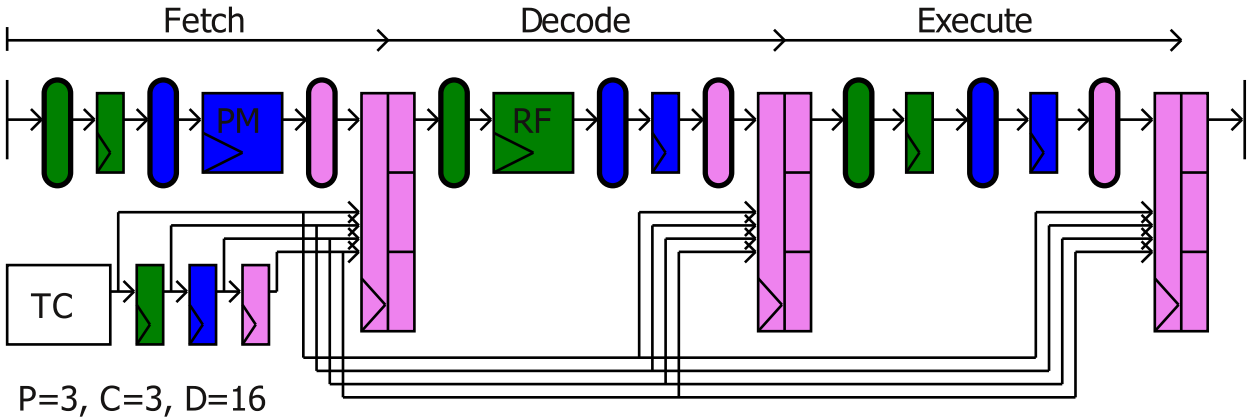


P=3

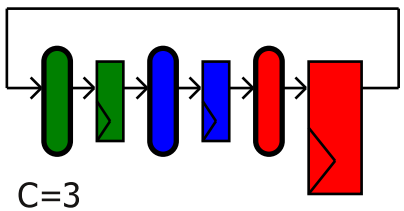
System Hyper Pipelining



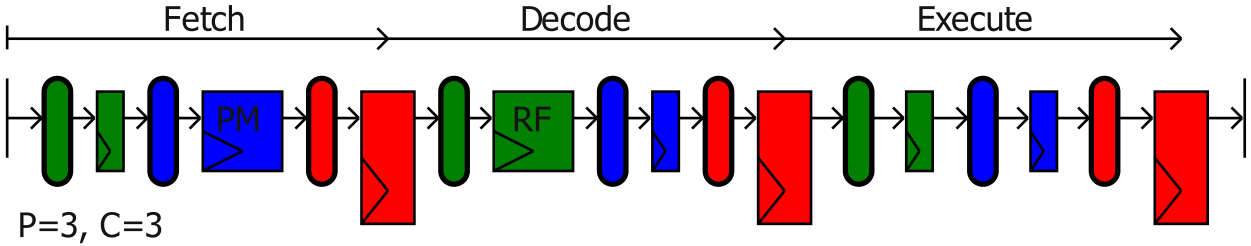
C=3, D=16



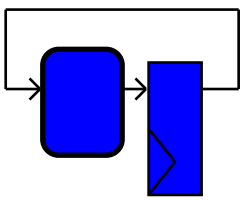
P=3, C=3, D=16

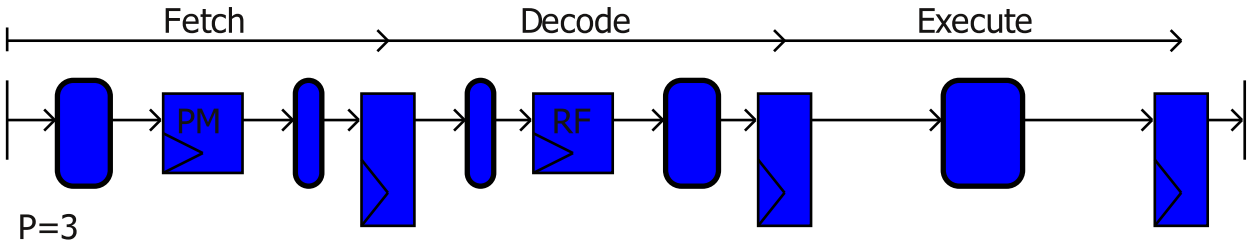


C=3



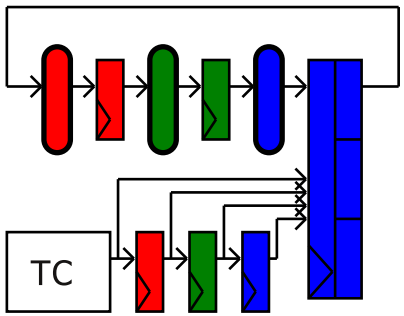
P=3, C=3



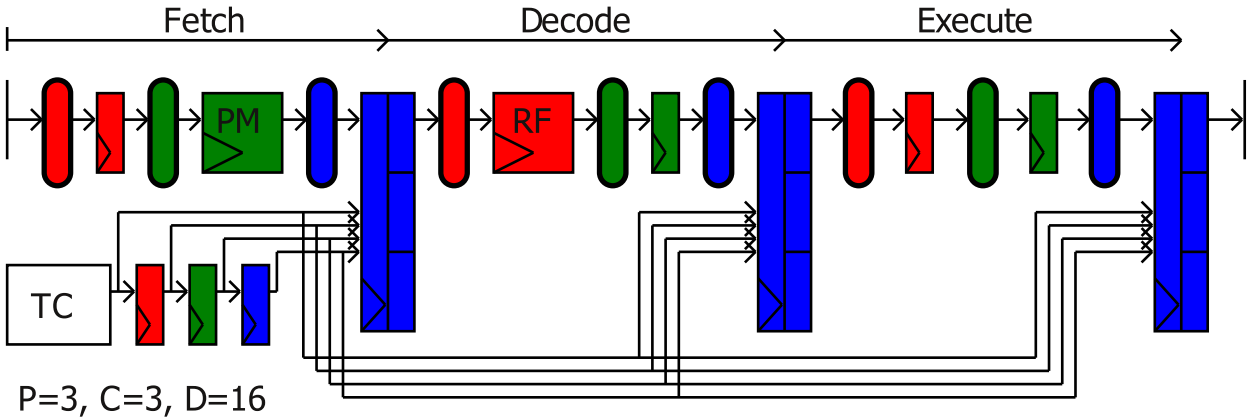


P=3

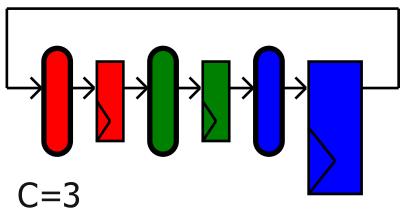
System Hyper Pipelining



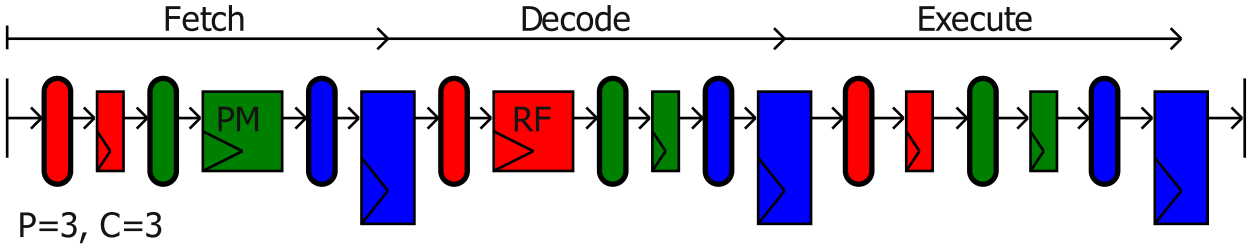
C=3, D=16



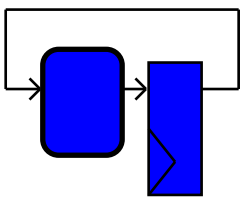
P=3, C=3, D=16

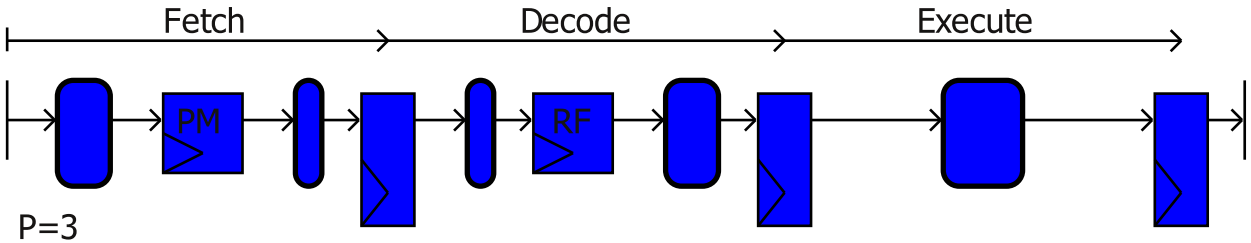


C=3



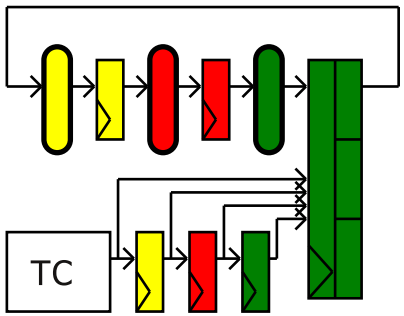
P=3, C=3



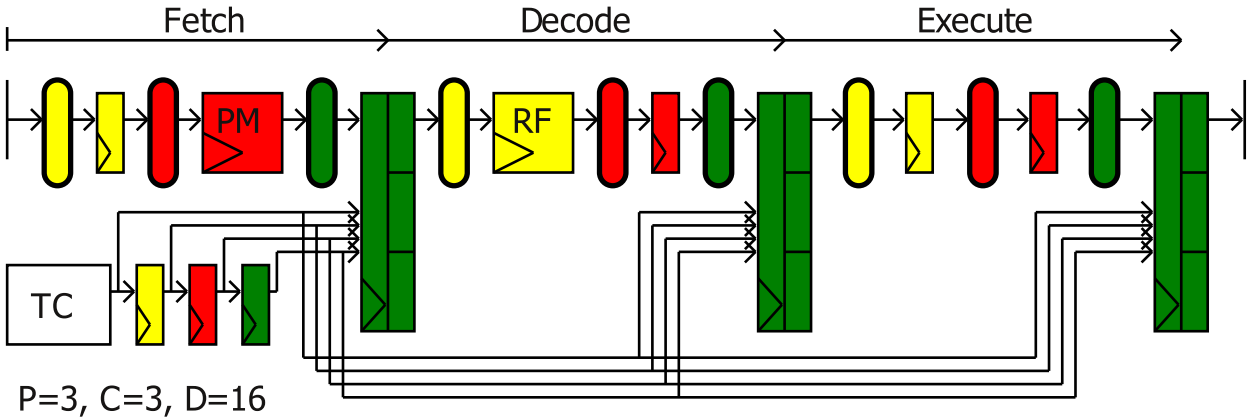


P=3

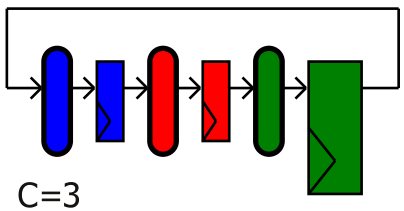
System Hyper Pipelining



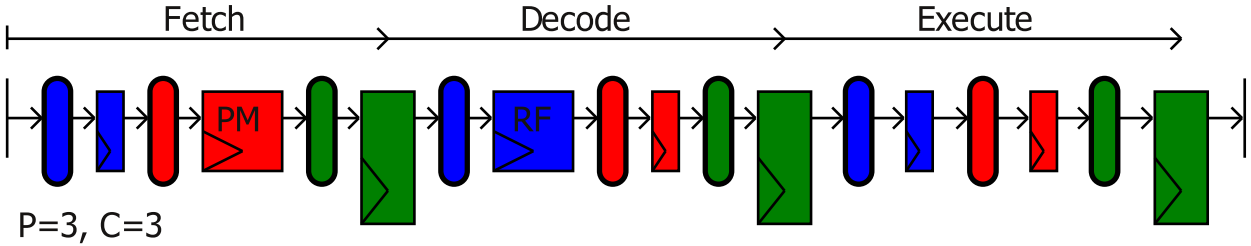
C=3, D=16



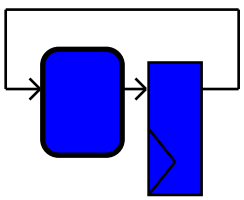
P=3, C=3, D=16

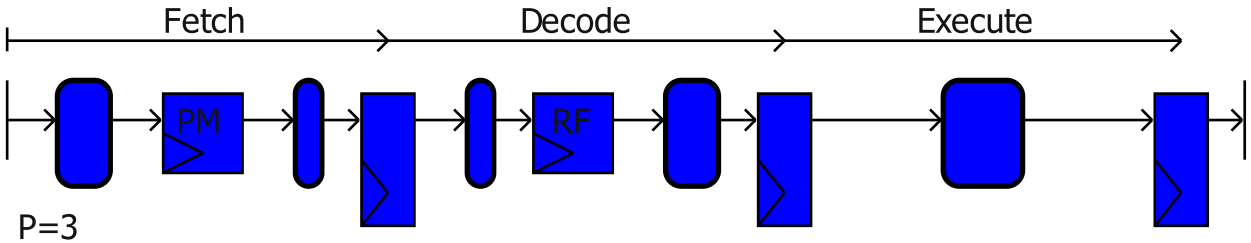


C=3



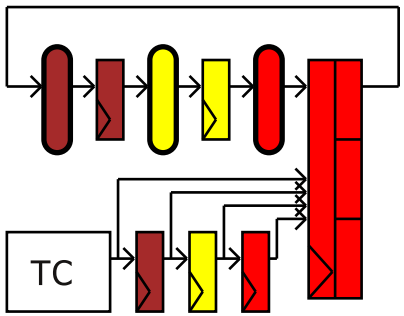
P=3, C=3



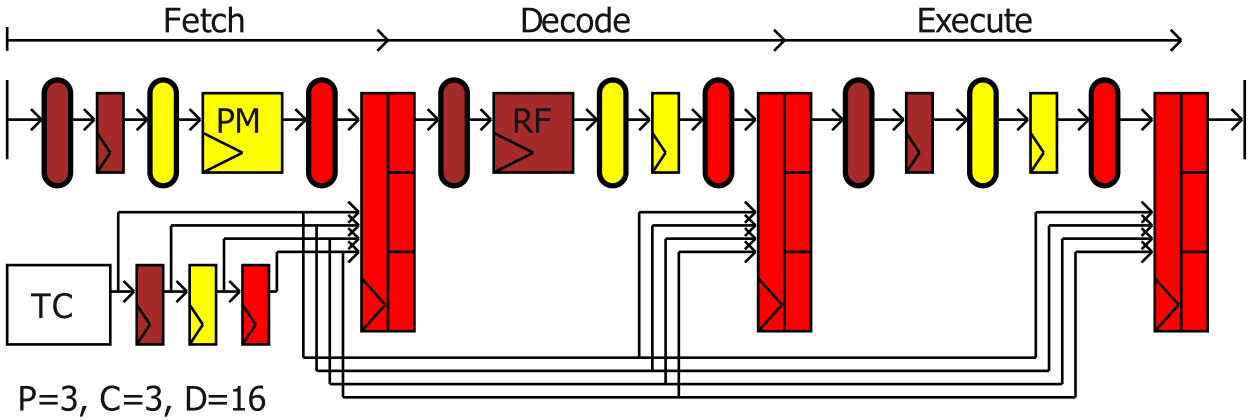


P=3

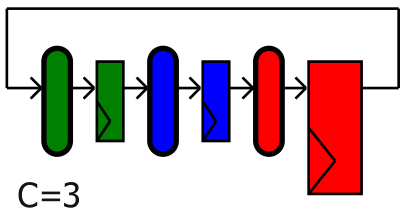
System Hyper Pipelining



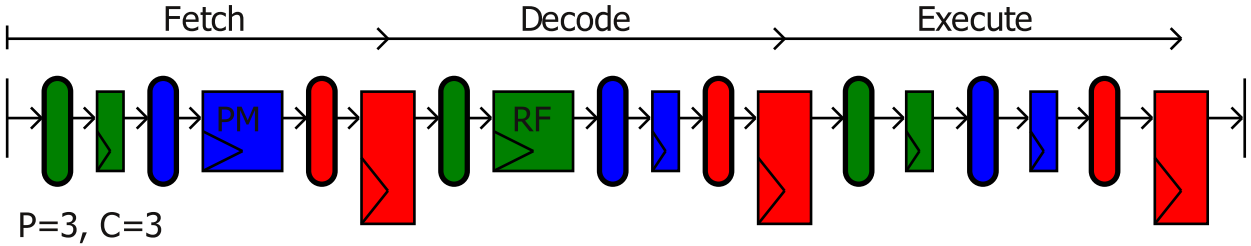
C=3, D=16



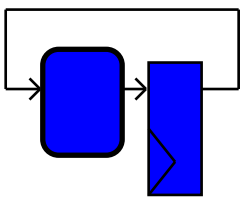
P=3, C=3, D=16

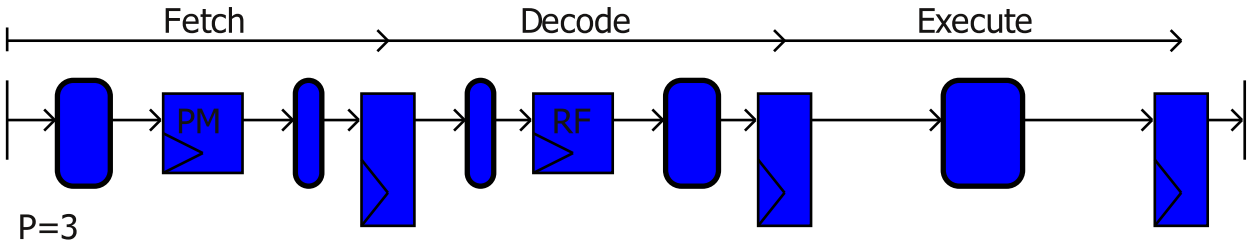


C=3



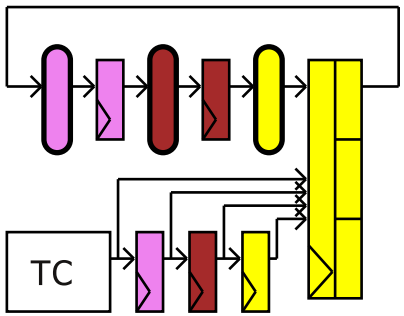
P=3, C=3



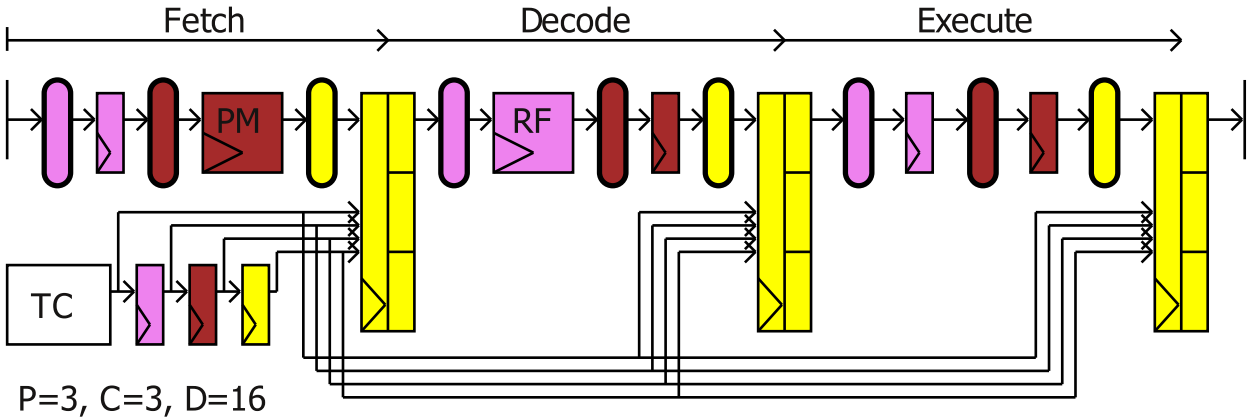


P=3

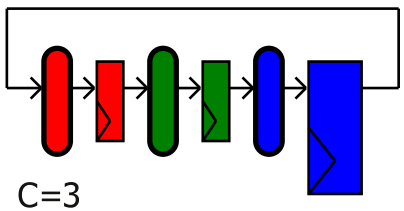
System Hyper Pipelining



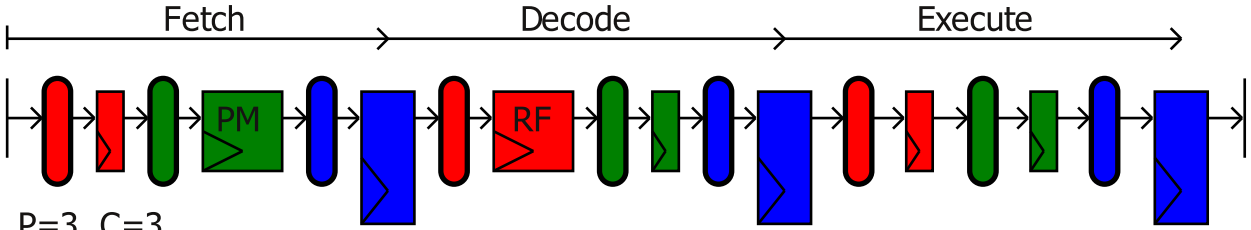
C=3, D=16



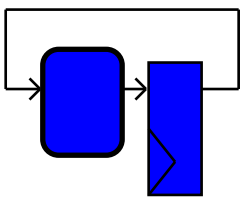
P=3, C=3, D=16

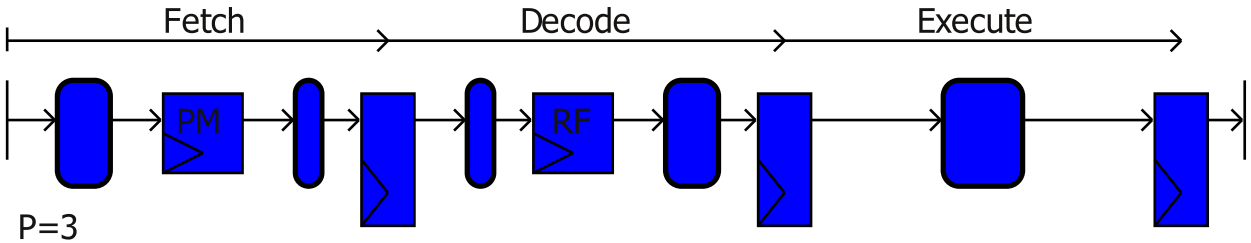


C=3



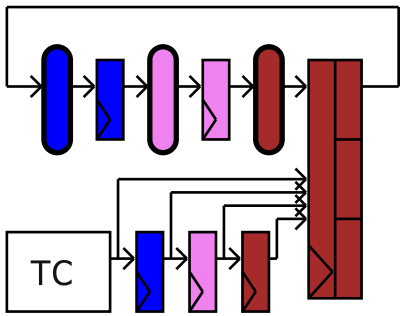
P=3, C=3



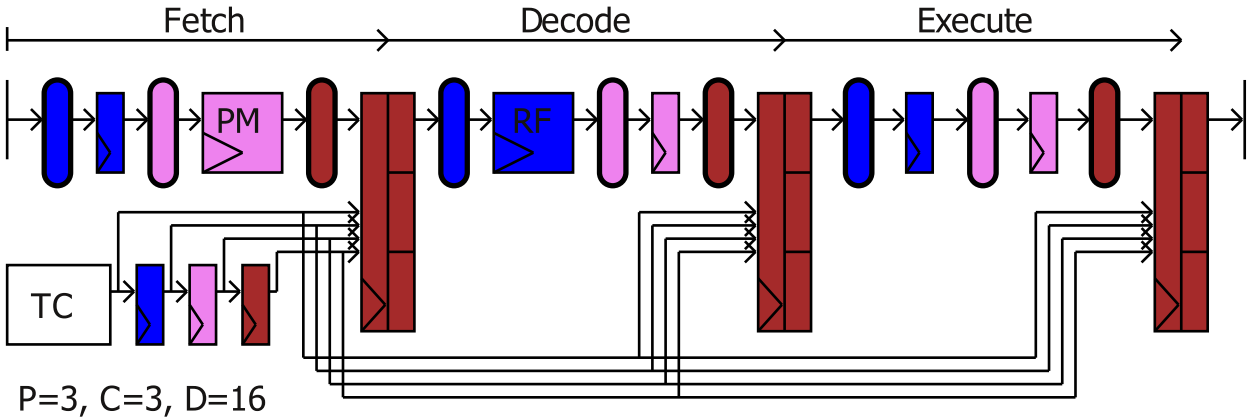


P=3

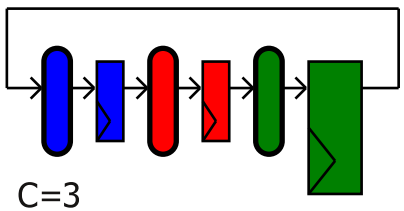
System Hyper Pipelining



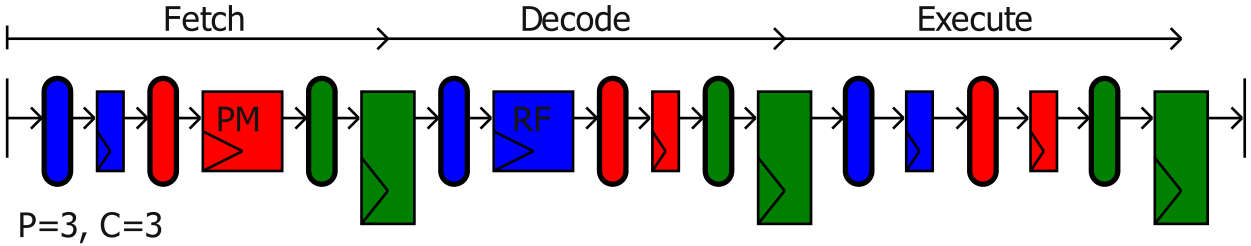
C=3, D=16



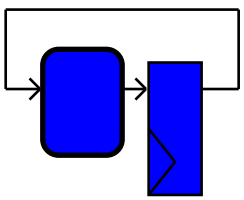
P=3, C=3, D=16

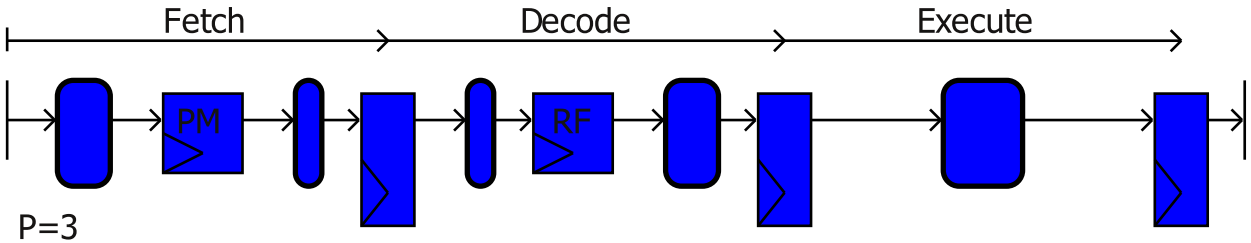


C=3



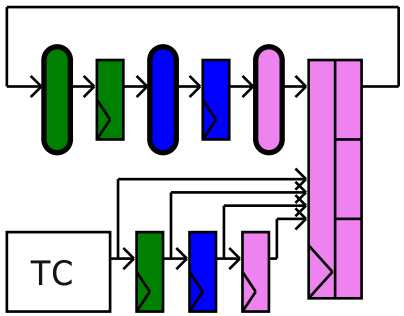
P=3, C=3



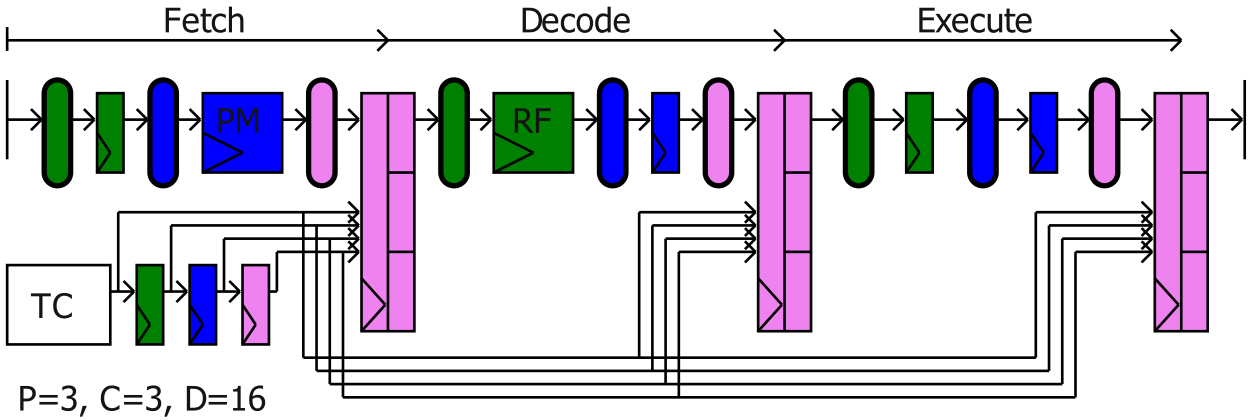


P=3

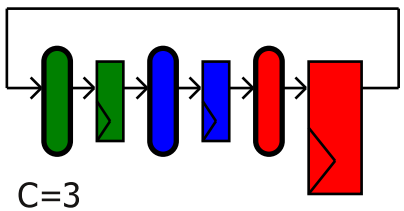
System Hyper Pipelining



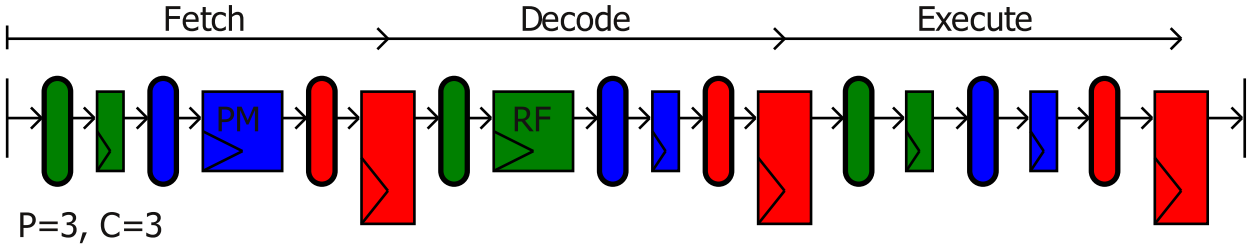
C=3, D=16



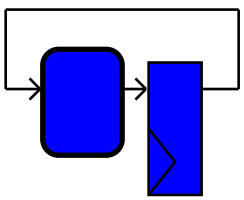
P=3, C=3, D=16

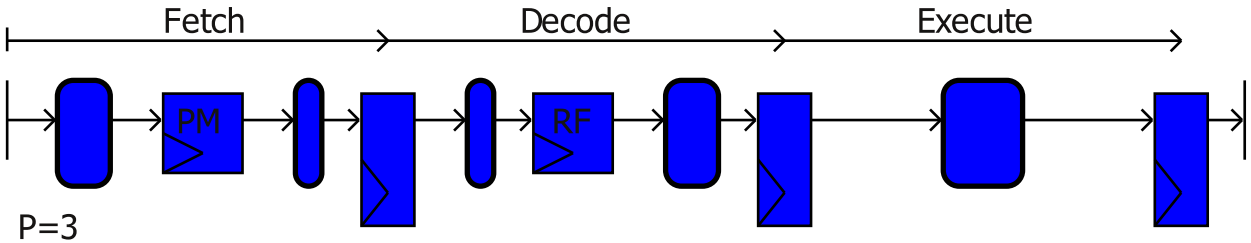


C=3



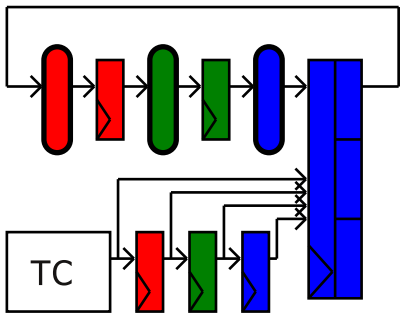
P=3, C=3



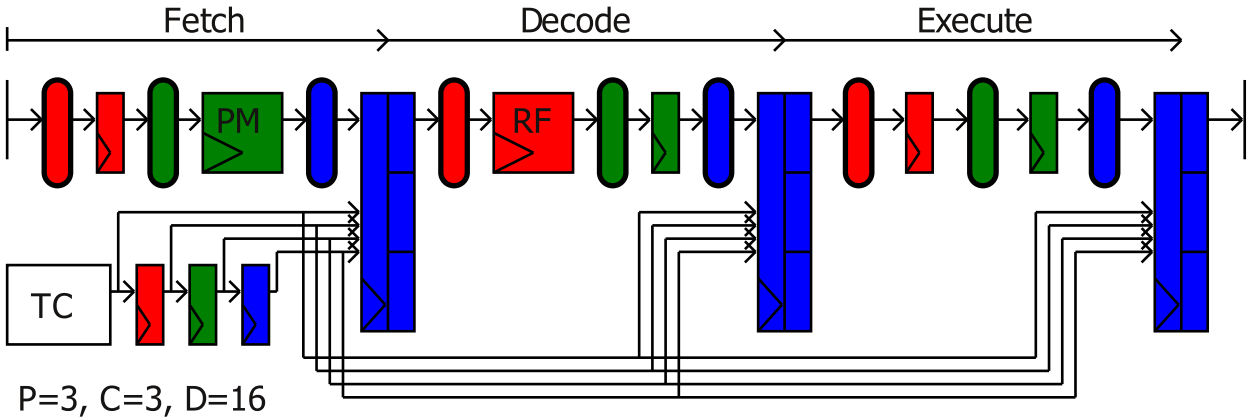


P=3

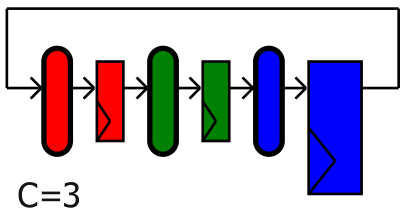
System Hyper Pipelining



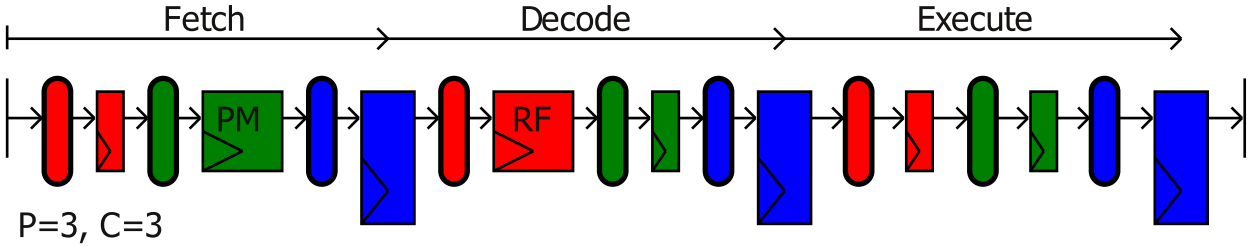
C=3, D=16



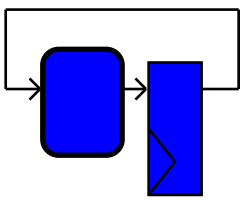
P=3, C=3, D=16

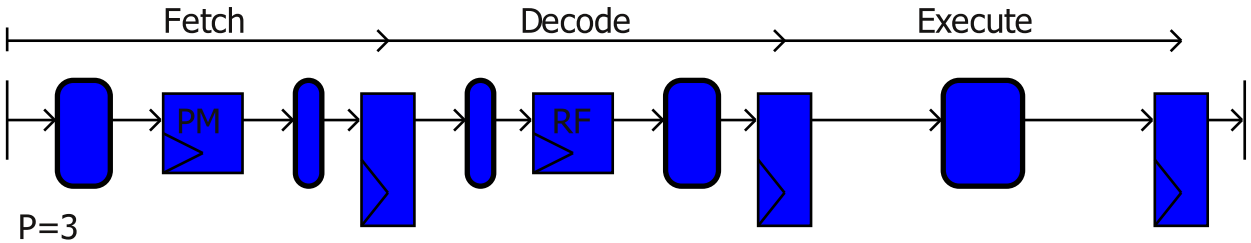


C=3



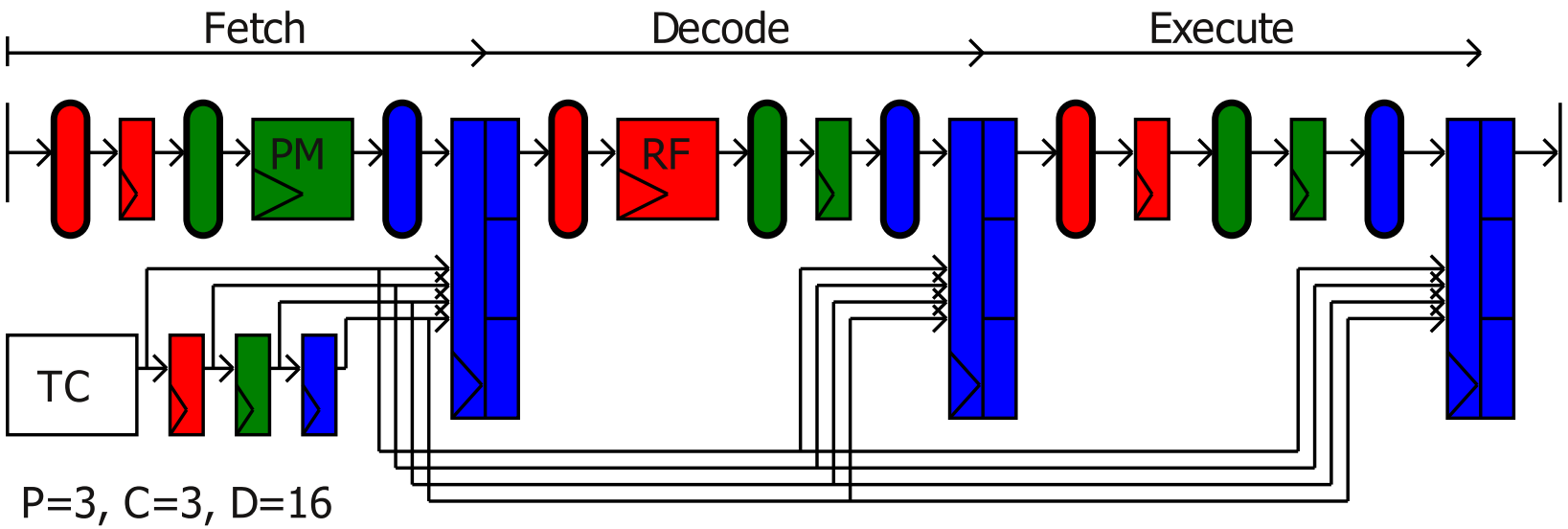
P=3, C=3





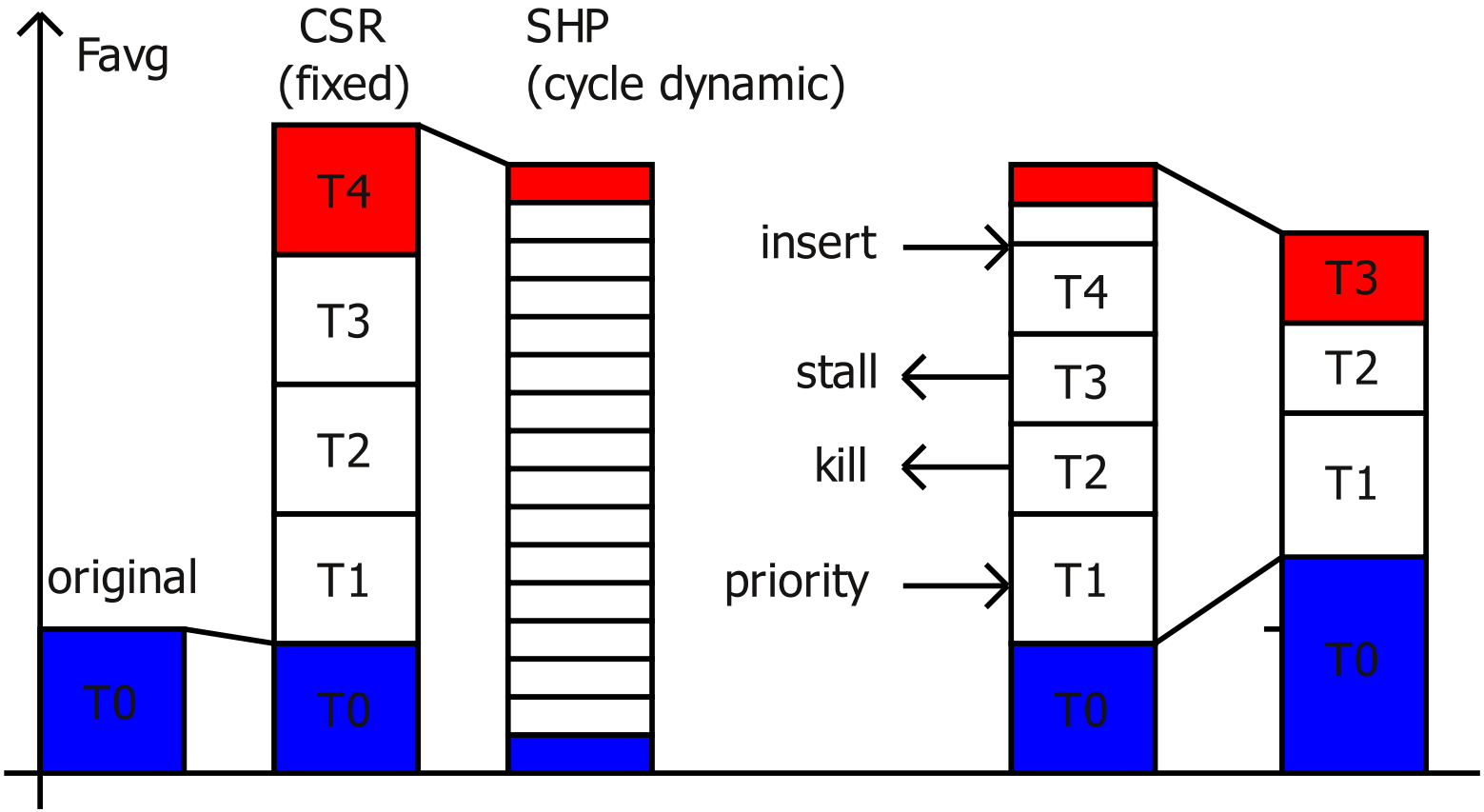
P=3

System Hyper Pipelining



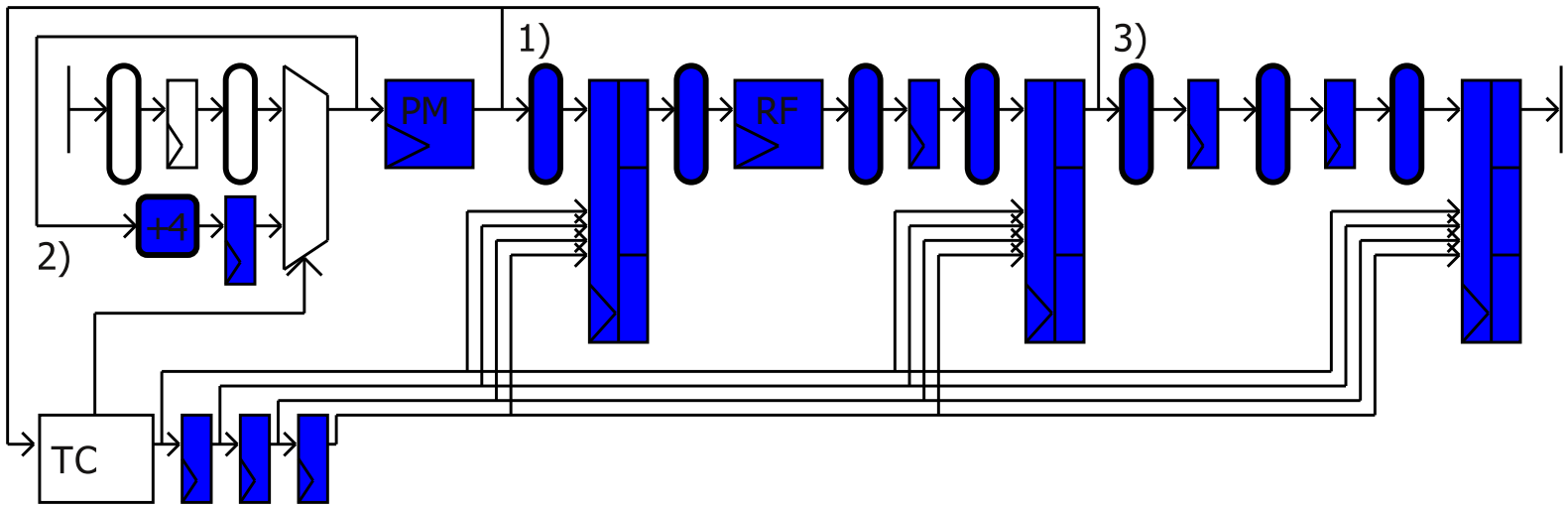
- Based on CSR
- Replaces original registers with memories (D).
- Adds thread stalling and therefore thread bypassing features.
- Cycle accurate performance balancing
- Late read / early write [3]

SHP, Performance Balancing



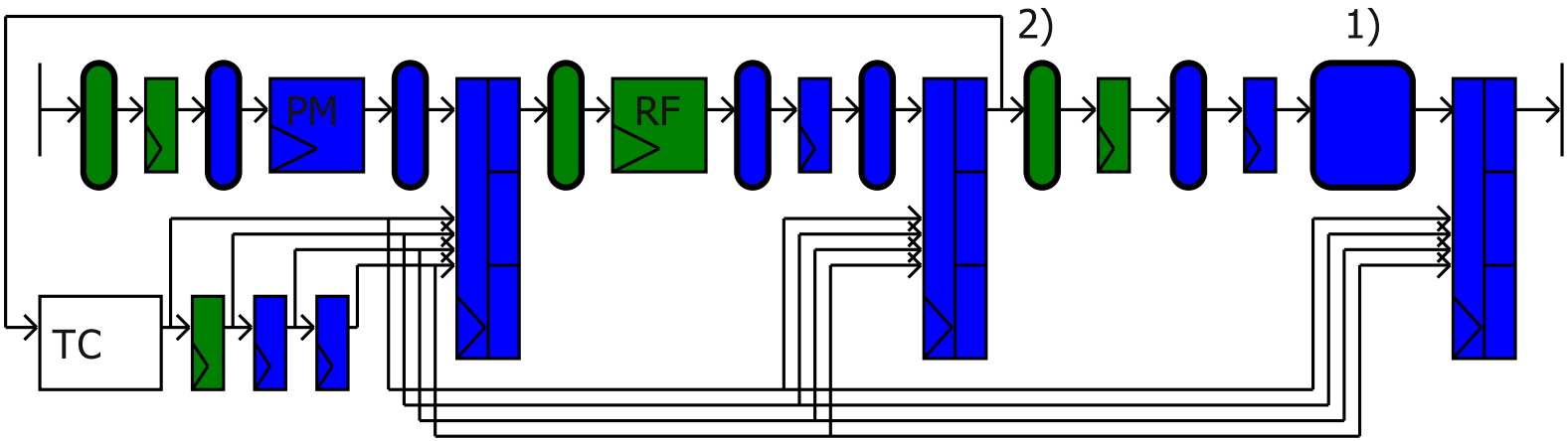
- Predictive runtime of specific threads
- Sync. start/execution, "Dynamic Length Instruction Words"

SHP, Deep Pipelining, T0 in “Beast Mode“



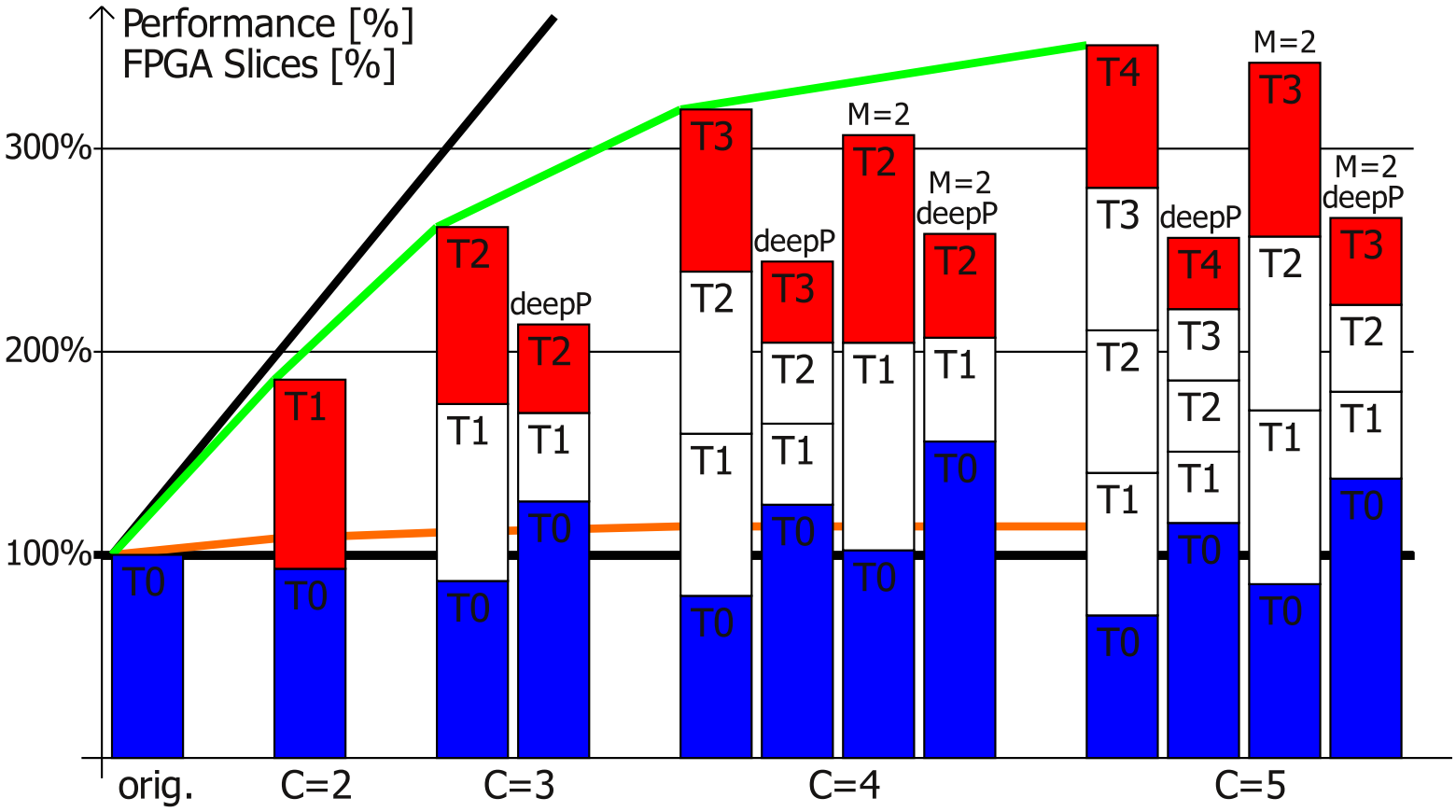
- Detect instruction dependency, LSB RISC-V ISA
- 1) Instruction LSB-sniffing
- 2) Enhanced PM read
- 3) Consider stall/flush signal
- No back-to-back, turned on/off on the fly

SHP, Extended Pipelining, “Frequency-Over-Scaling“



- Less registers per path than necessary
- Multi-cycle path for example in datapath section
- Re-execution of thread with valid multi-cycle path

SHP, Performance per Area Factor

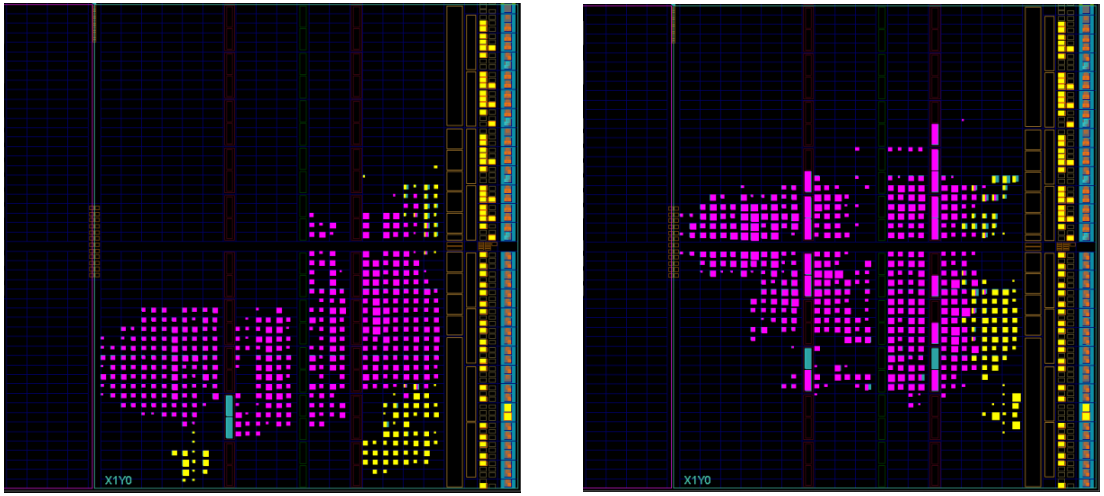


- CHStone

SHP, System Level

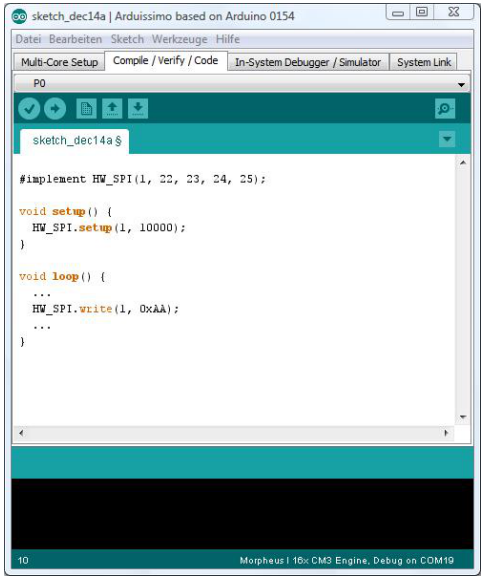
- Performance (frequency) over area curve
- System level performance improvements !!!
- microRISC (Vscale), microcontroller, virtual peripherals, ...
- miniRISC (lowRISC, Rocket), SoC, OS, FPU, accelerators, ...

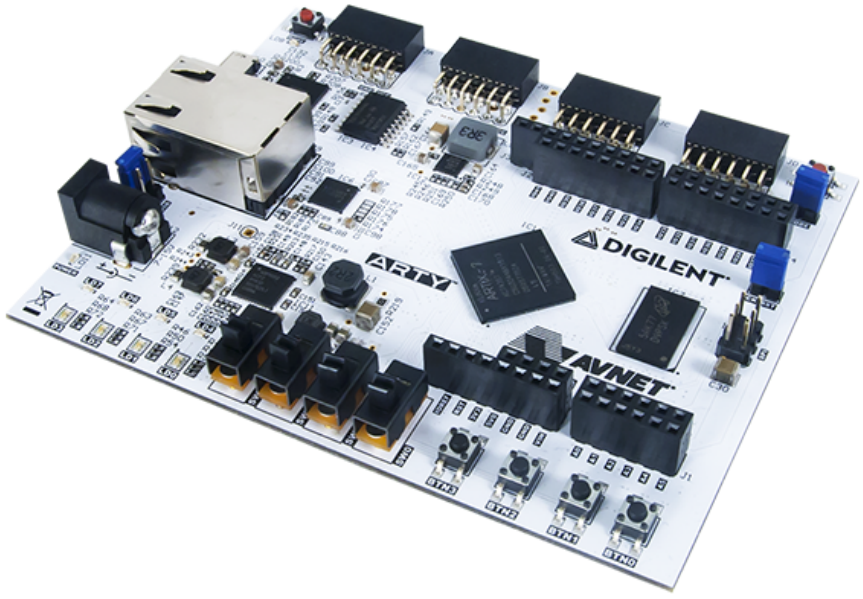
microRISC



- Vscale @80MHz async DDR3, SHP-ed @250MHz synchron
- Based on RV32IM subset (no FENCE and no SI, "RV32B")
- Individual threads handle interrupts (less stack activity).
- Main focus on virtual peripherals
- TC with event handler (complex timer)

microRISC

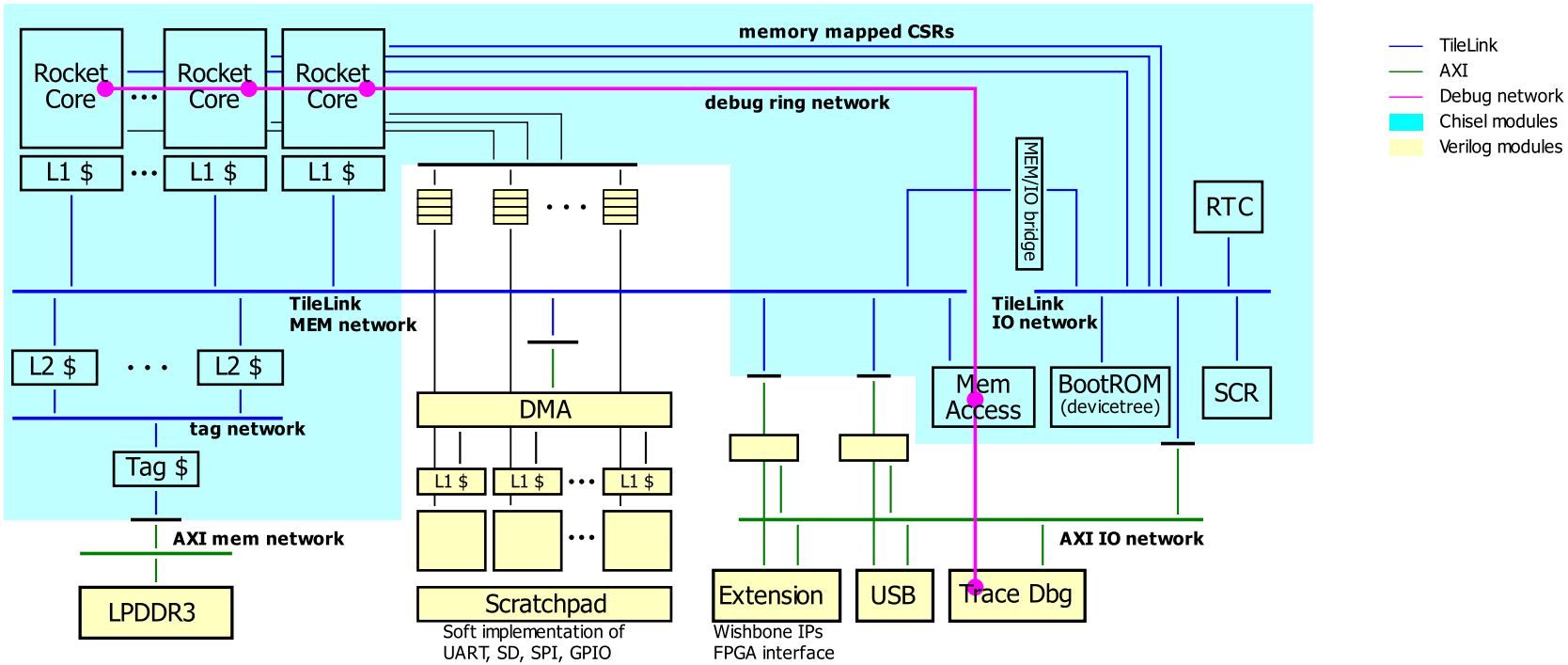




- UARTs
- PWMs
- ...

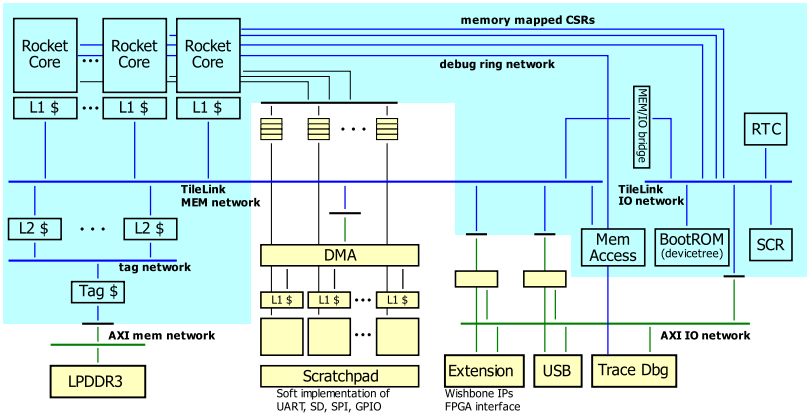
- Standard toolchain compatible (C++, ...)
- Demo: How to run timing critical software peripherals based on a highly dynamic SHP-ed core.

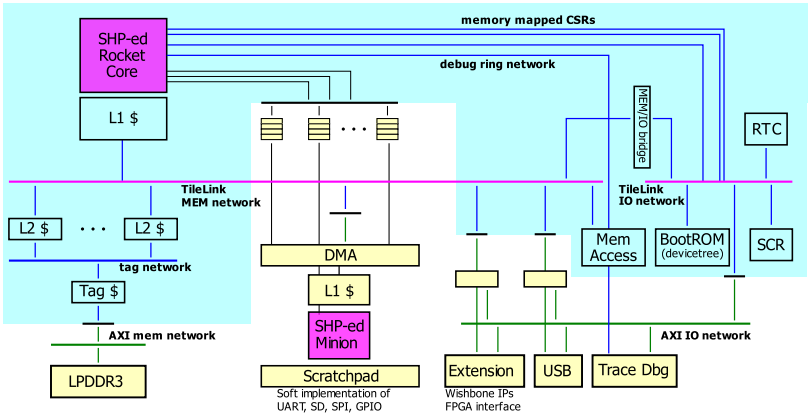
lowRISC / University of Cambridge



- Diagram: Wei Song, lowRISC / University of Cambridge
- Wei: "personal view"

miniRISC (based on lowRISC)





- Minions part of Rocket core, heterogeneous multi-core
- SHP impact on performance per area and SoC architecture
- Arbiter/multiplexer: blocking
- Multilayer: complex
- SHP: time sliced usage at higher speed

Miscellaneous

- Simple programming model
- Fork-join operations, OpenMP
- Estimated ASIC numbers in paper [3]
- Altera Hyper Pipelining technology looks promising
- More on-FPGA memory (memory wall)
- Power consumption (work ongoing)
- Source code of projects in PDVL (VHDL, Verilog)

References

[1] C. Leiserson and J. Saxe, “Retiming Synchronous Circuitry“, Algorithmica, vol. 6, no. 1, pp. 5-35, 1991.

[2] N. Weaver, Y. Markovskiy, Y. Patel and J. Wawrzynek, “Post-Placement C-slow Retiming for the Xilinx Virtex FPGA,“ FPGA 2003, February 23-25, 2003, Monterey, CA, USA.

[3] T. Strauch, “The Effects of System Hyper Pipelining on Three Computational Benchmarks Using FPGAs“, 11th International Symposium in Applied Reconfigurable Computing, ARC 2015, 13-17 April 2015, Bochum, Germany, pp. 280-290.

You made it !!!

- Thank you
- @arduissimo
- www.cloudx.cc
- tobias@cloudx.cc
- Call for cooperation: SHP-ed CPU in an ASIC technology