SystemC Refinement of Abstract Adaptive Processes for Implementation into Dynamically Reconfigurable Hardware

F. Herrera
E. Villar

P.A. Hartmann

Outline

- Motivation
- Contribution
- Results
- Conclusions

Motivation (Requirements for Specification and Implementation)

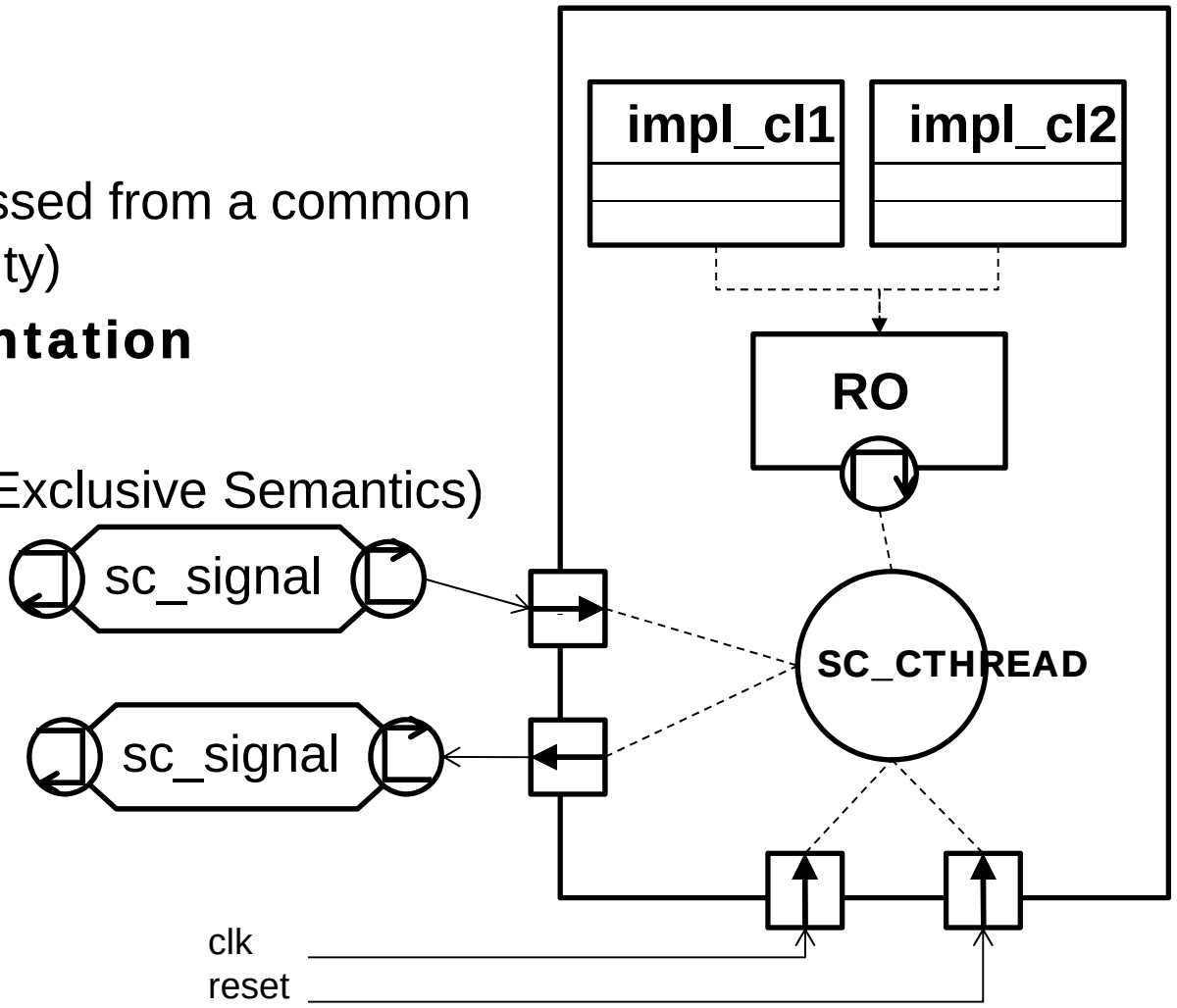
- **Challenges**
  - Complexity, Changing Environments, Reusability, Performance
- **Modelling**
  - Abstraction (High-Level Modelling)
  - **Adaptivity: A main aspect**
- **Design**
  - ESL, Co-Design (HW/SW solutions)
  - DRHW: Hw Performance for Mutually Exclusive Algorithms fulfilling a same functionality

Motivation (SoA)

- **Adaptivity in SW Programming**
  - CLOS, Dylan, …
- **SystemC High-Level Modelling (ESL)**
  - SystemC-H, SysteMoC, HetSC
  - DRHW
- **ANDRES**
  - Adaptive HetSC (A-HetSC)
  - OSSS+R

Motivation (Previous Work: OSSS+R)

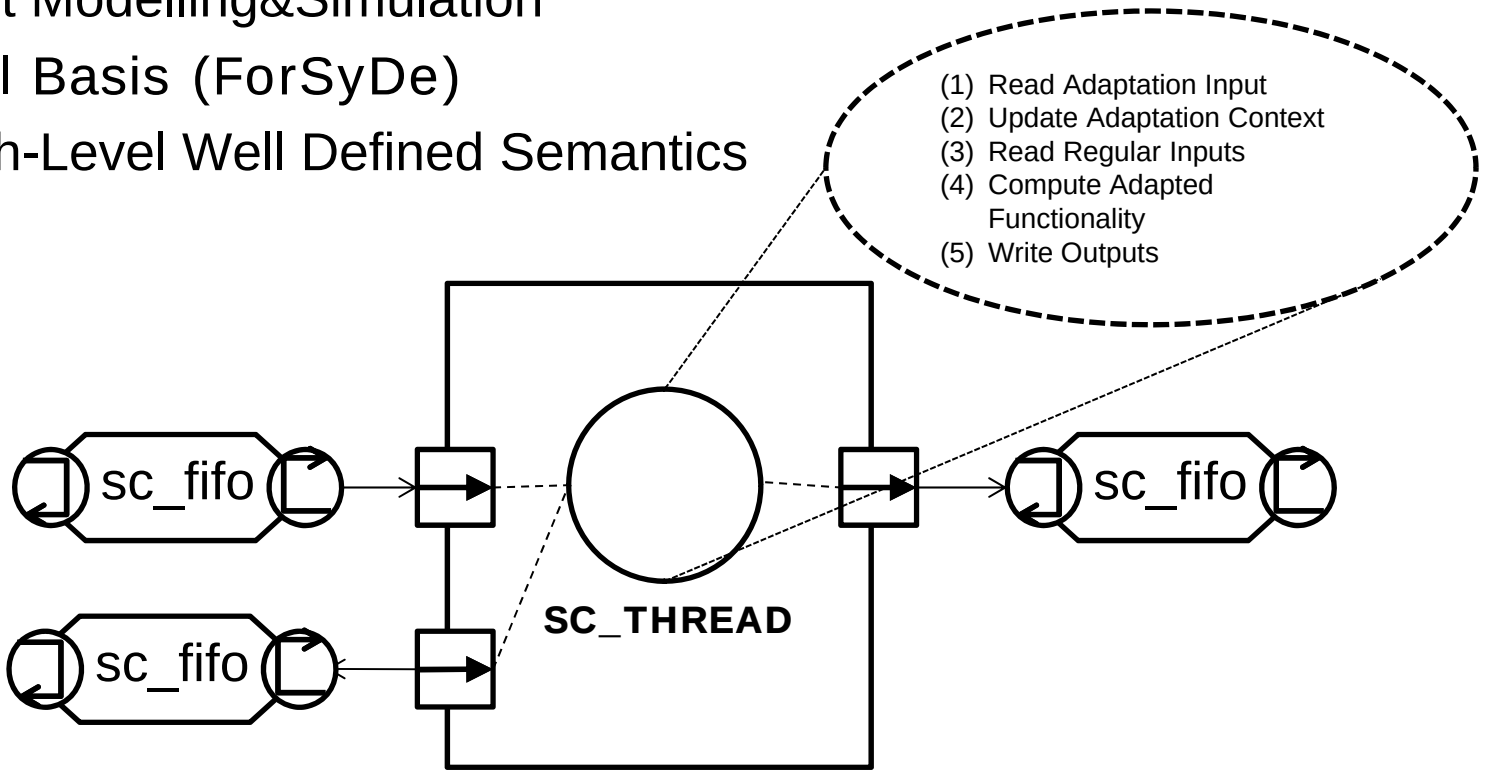
- **Modelling**
  - OO abstraction
  - functionality accessed from a common Interface (Adaptivity)
- **Link to Implementation**
  - Fossy
  - DRHW (Mutually Exclusive Semantics)
  - Clocked Model
  - Abstraction and Simulation Speed Improvable!



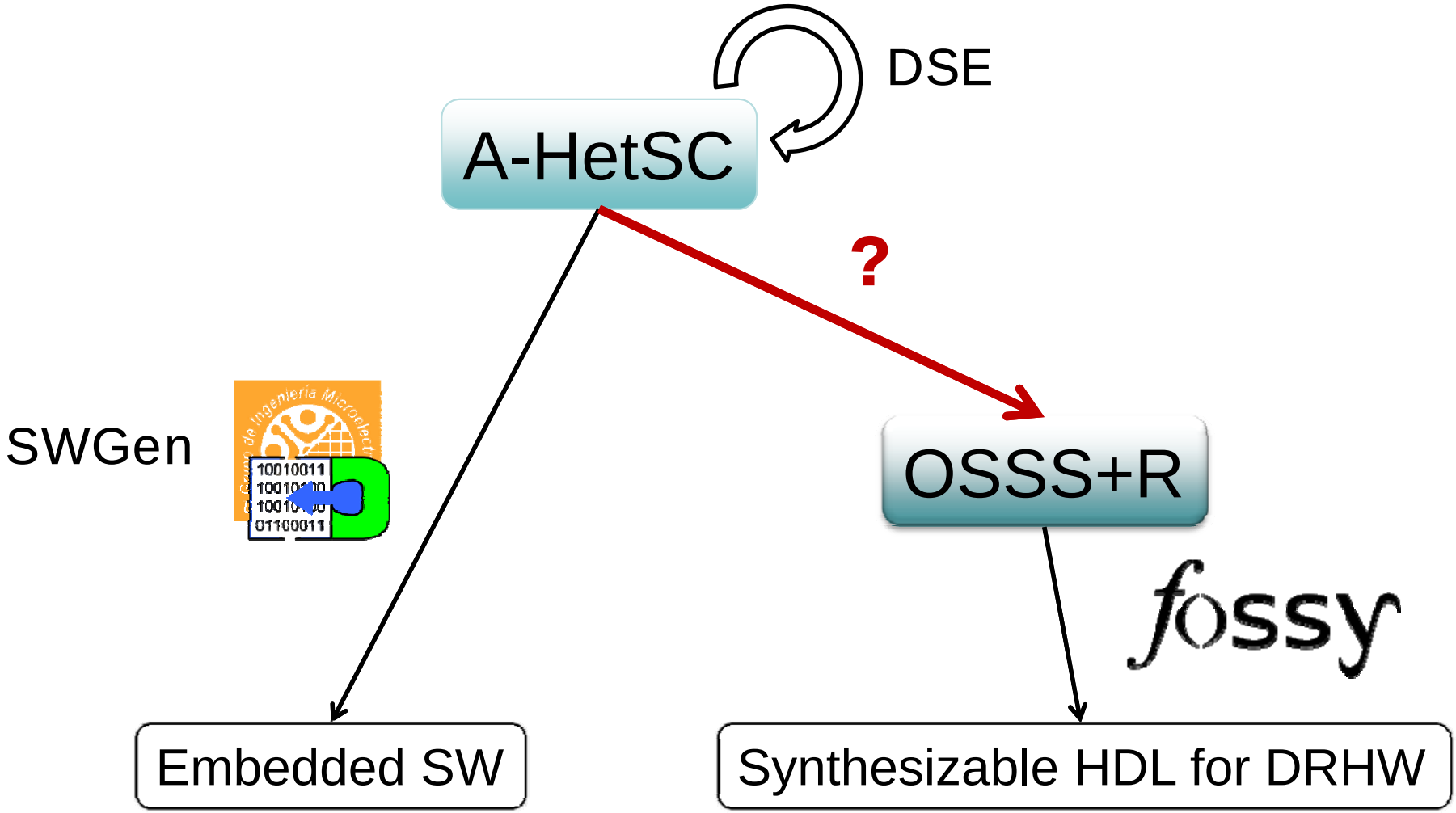
Motivation (Previous Work: A-HetSC)

- Untimed Specification of Adaptivity (HetSC Adaptive Processes, HAPs)
  - Fast Modelling&Simulation
- Formal Basis (ForSyDe)
  - High-Level Well Defined Semantics

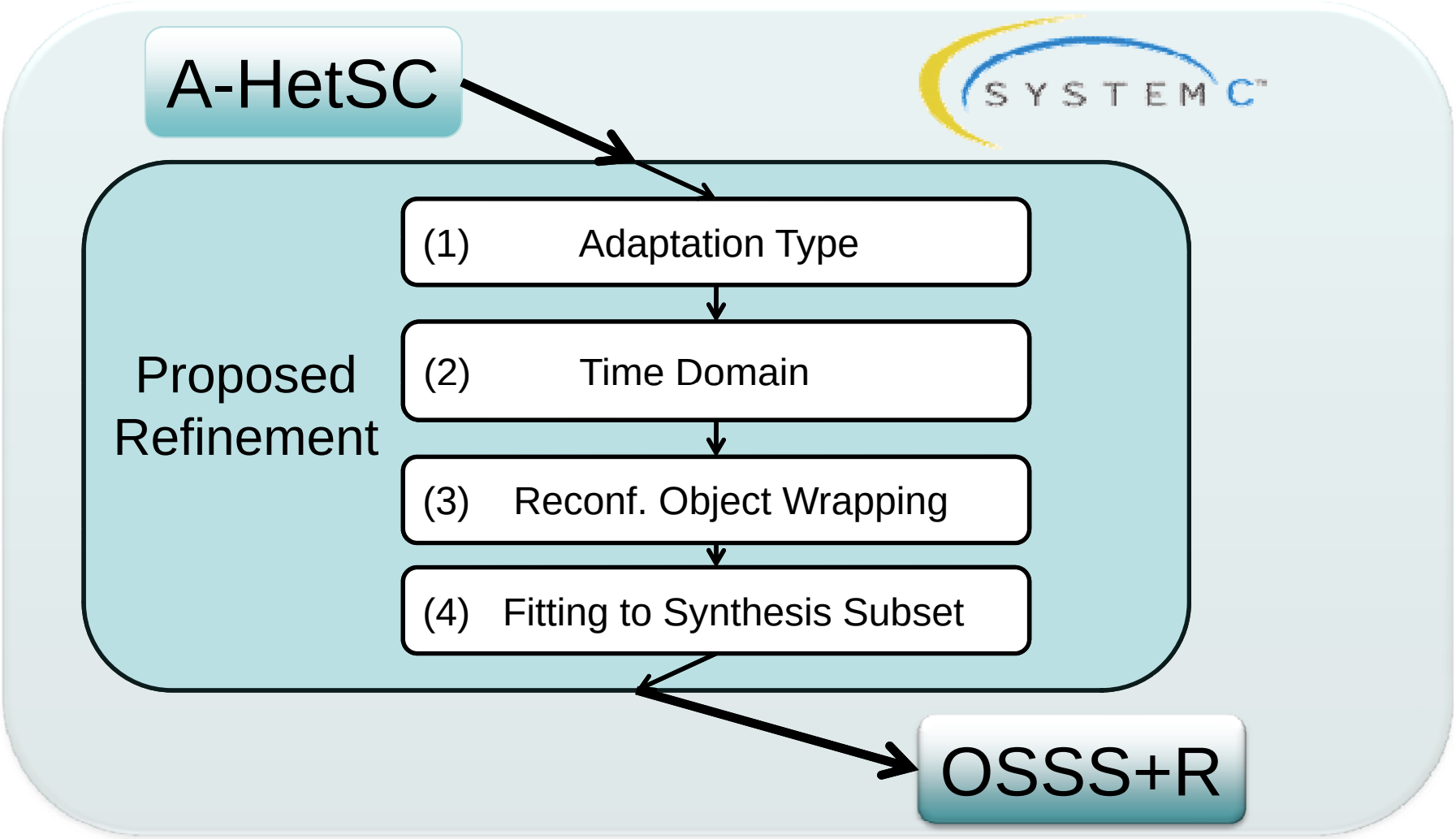
– Link to HW implementation



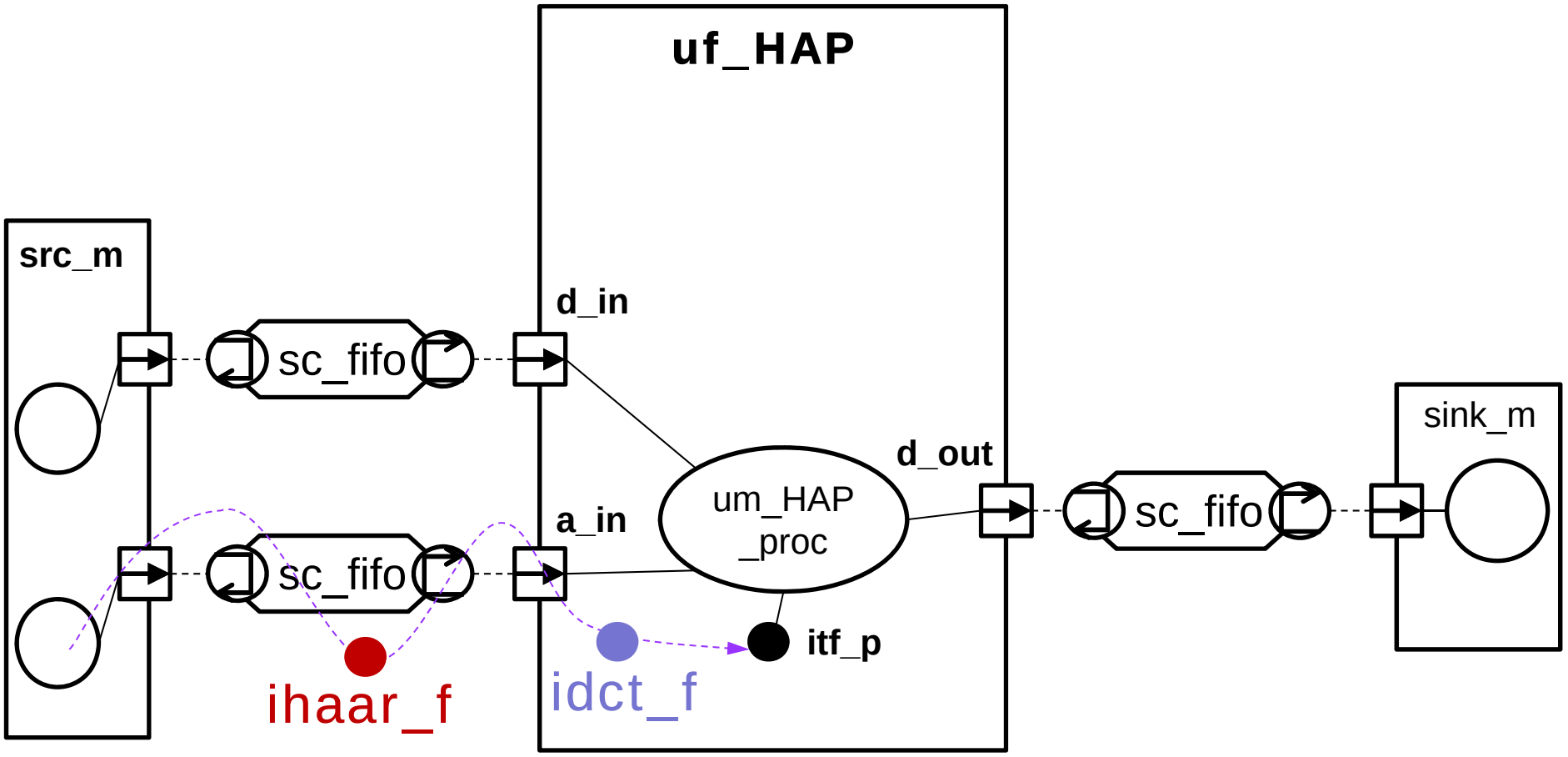
Motivation (From A-HetSC to OSSS+R)



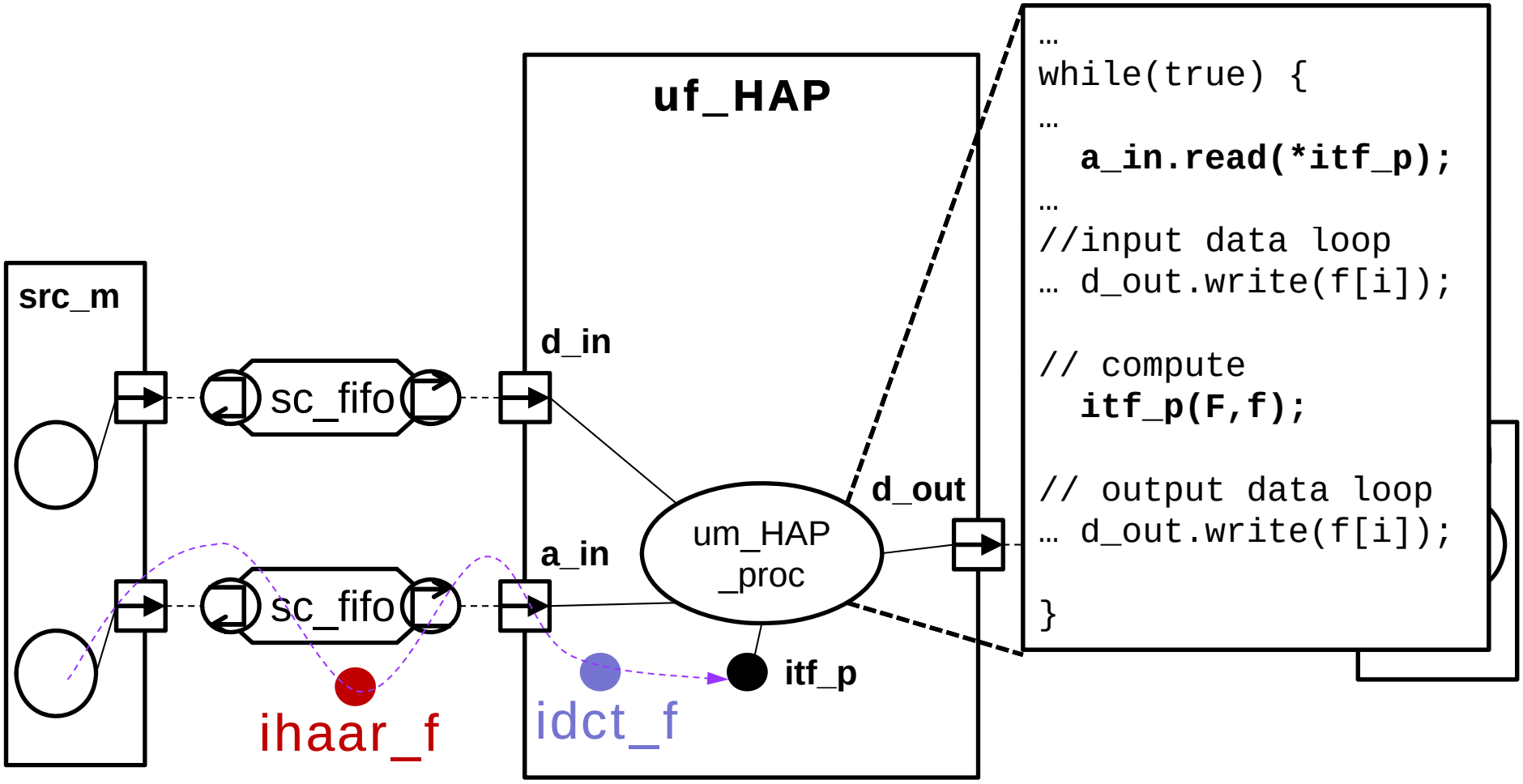
Contribution: Refinement Flow from A-HetSC to OSSS+R



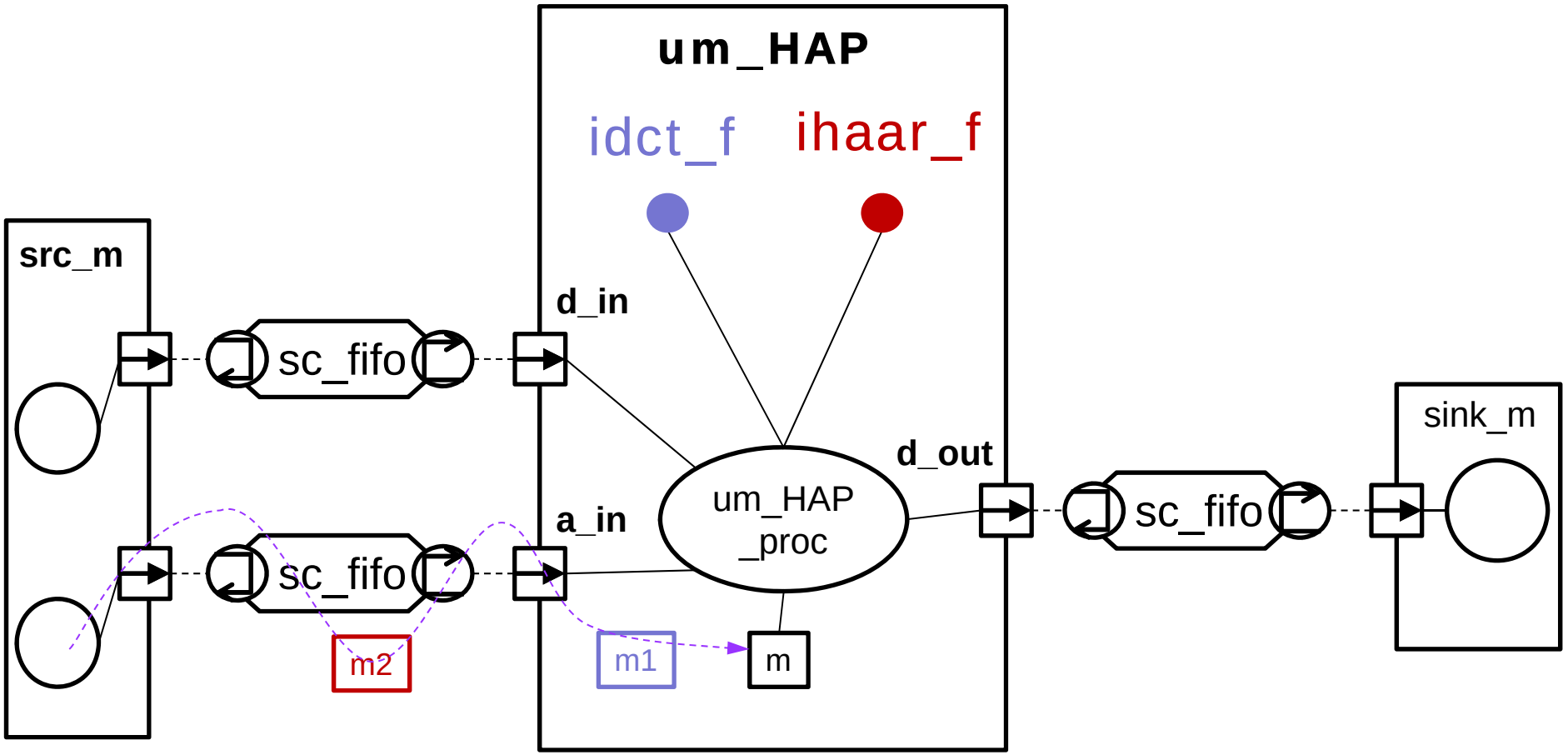
Starting Model: Function Adaptation



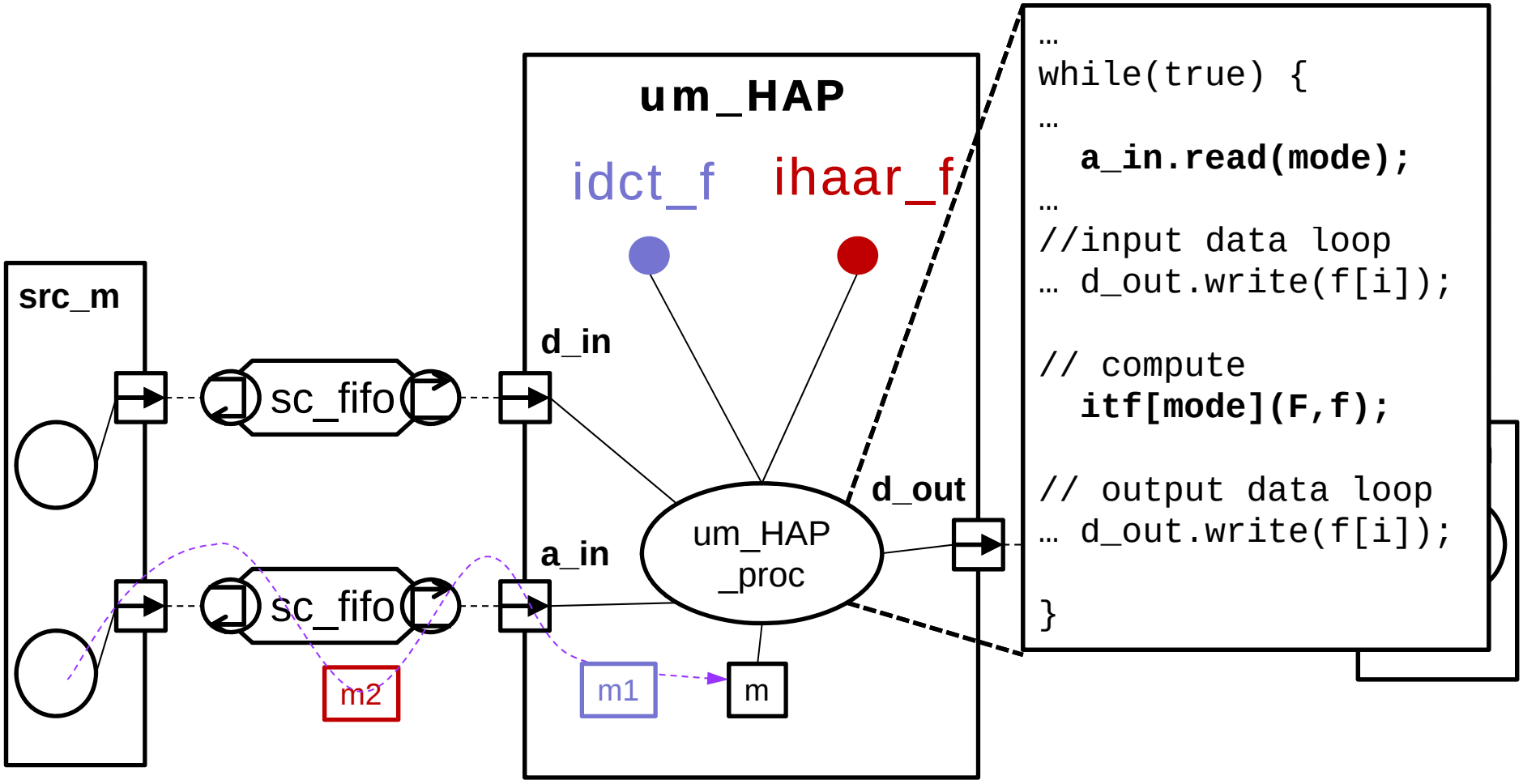
Starting Model: Function Adaptation



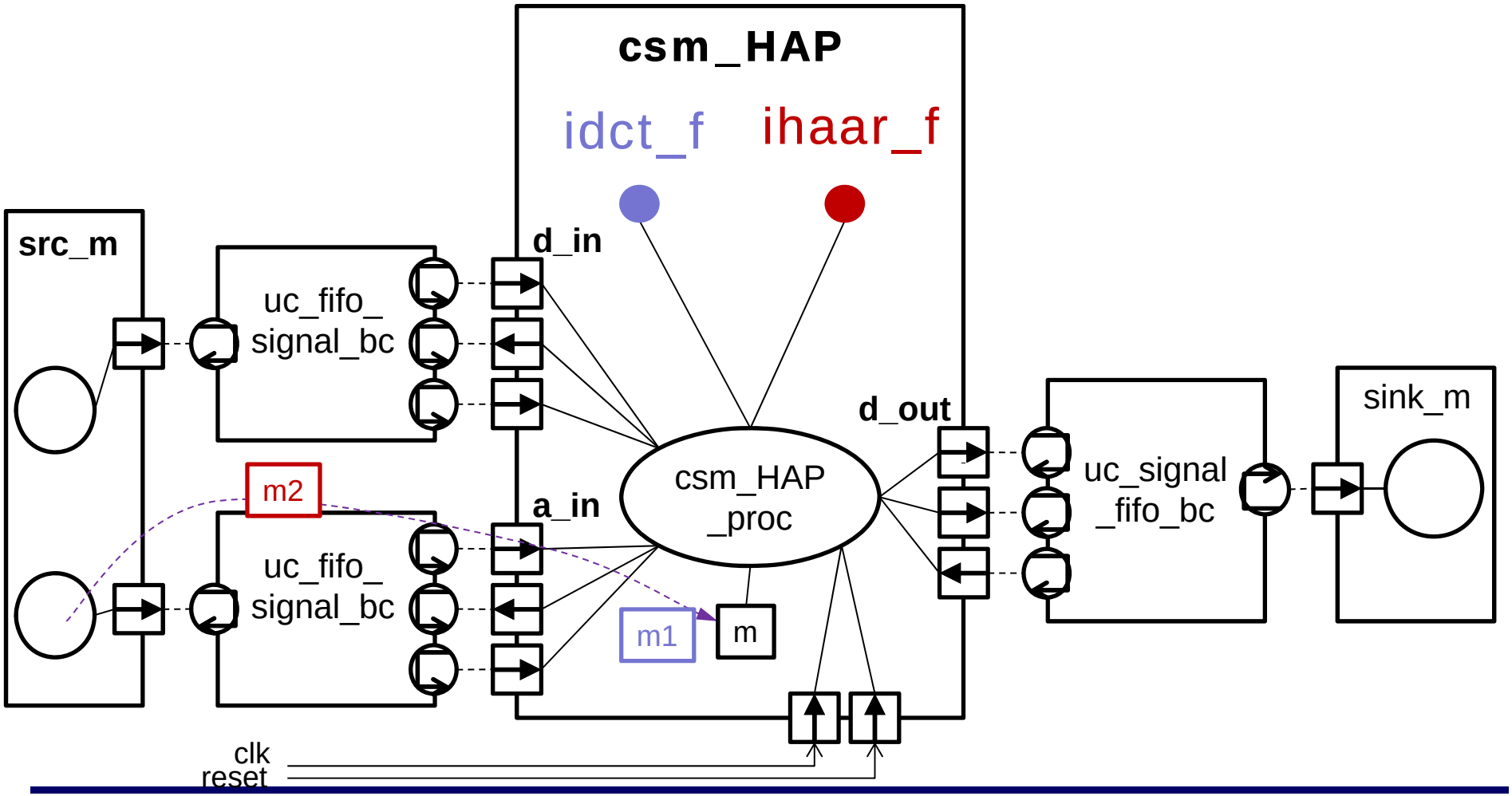
1st Ref.: Adaptation Type



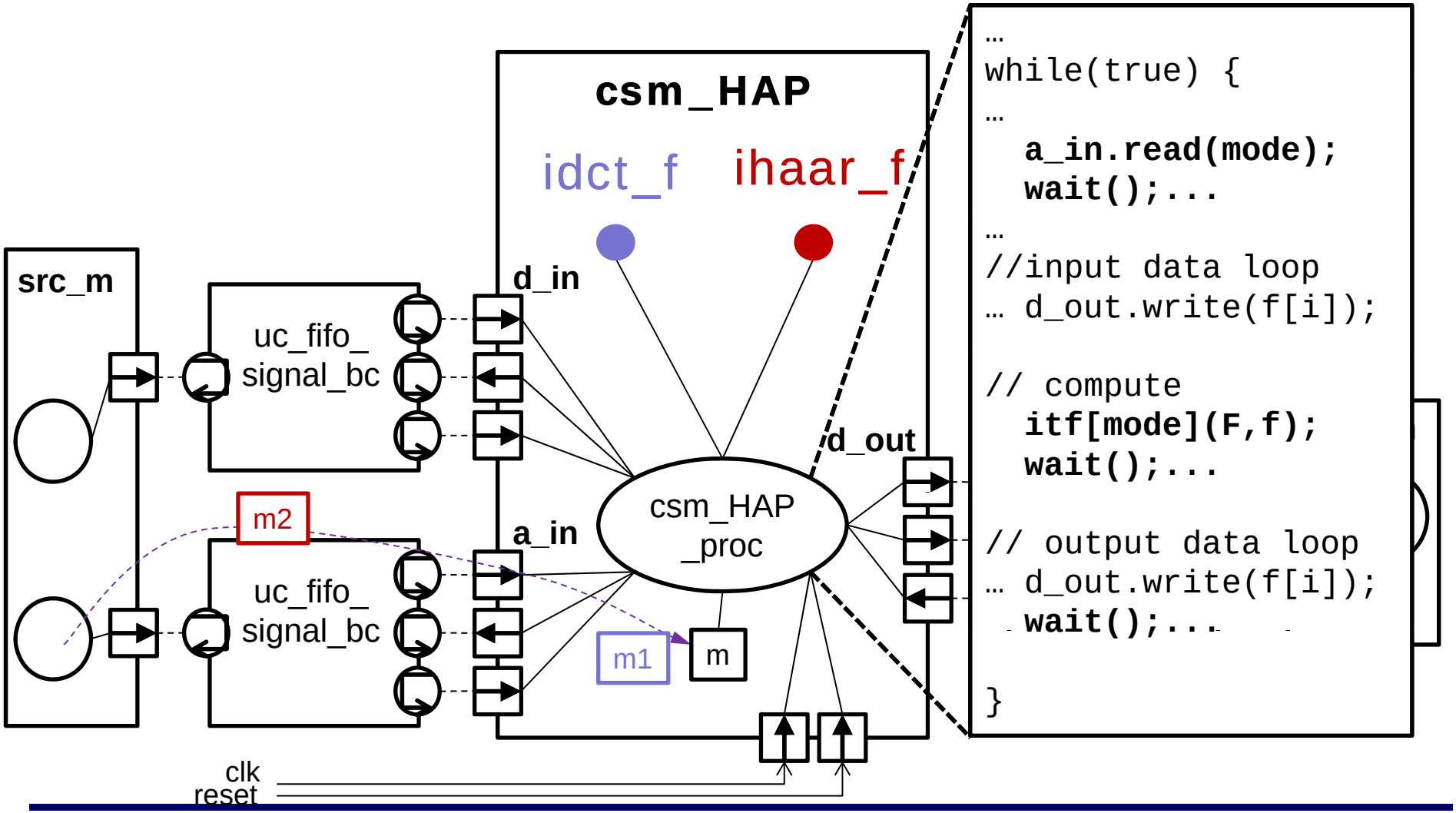
1st Ref.: Adaptation Type



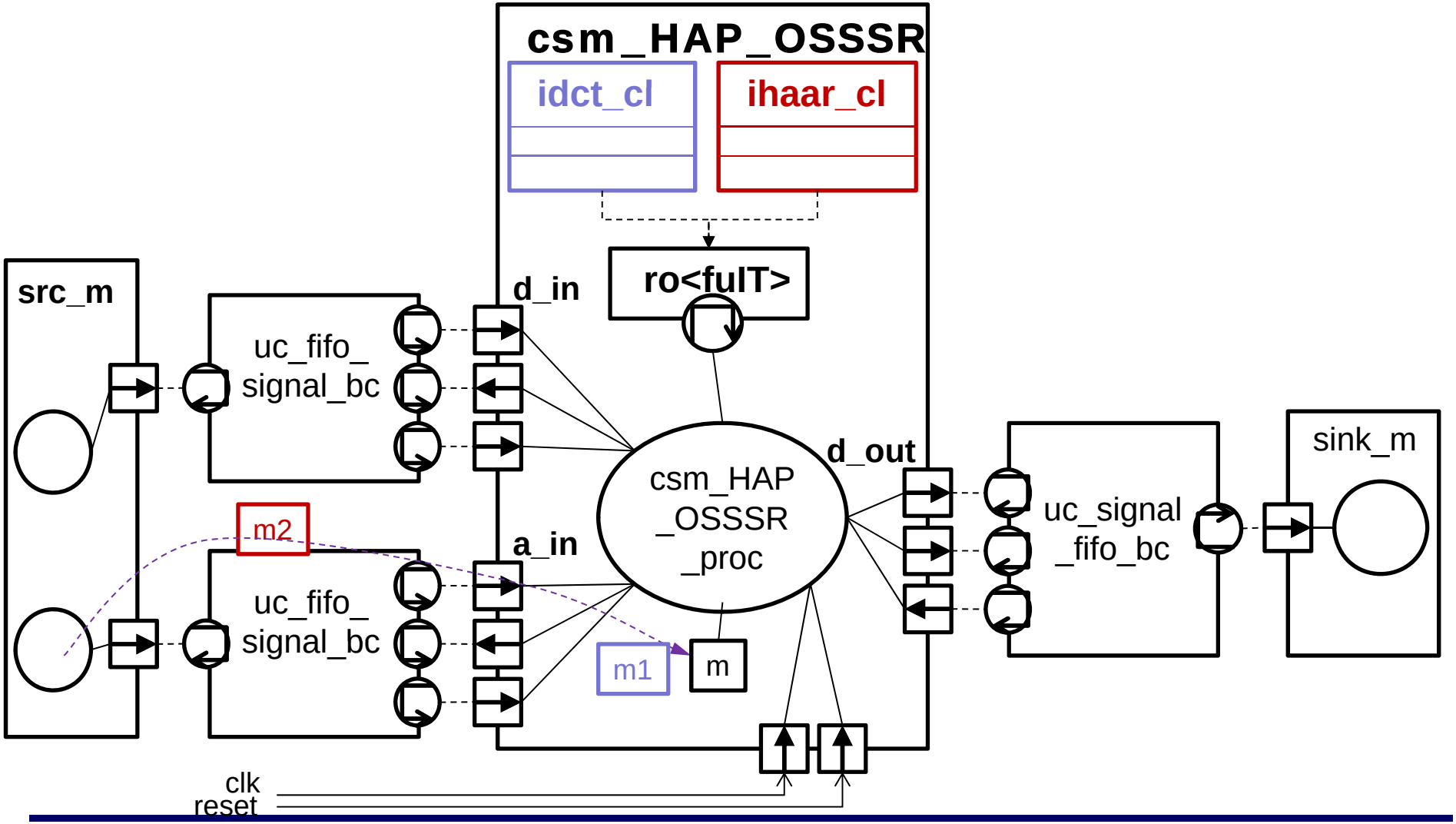
2nd Ref.: Time Domain



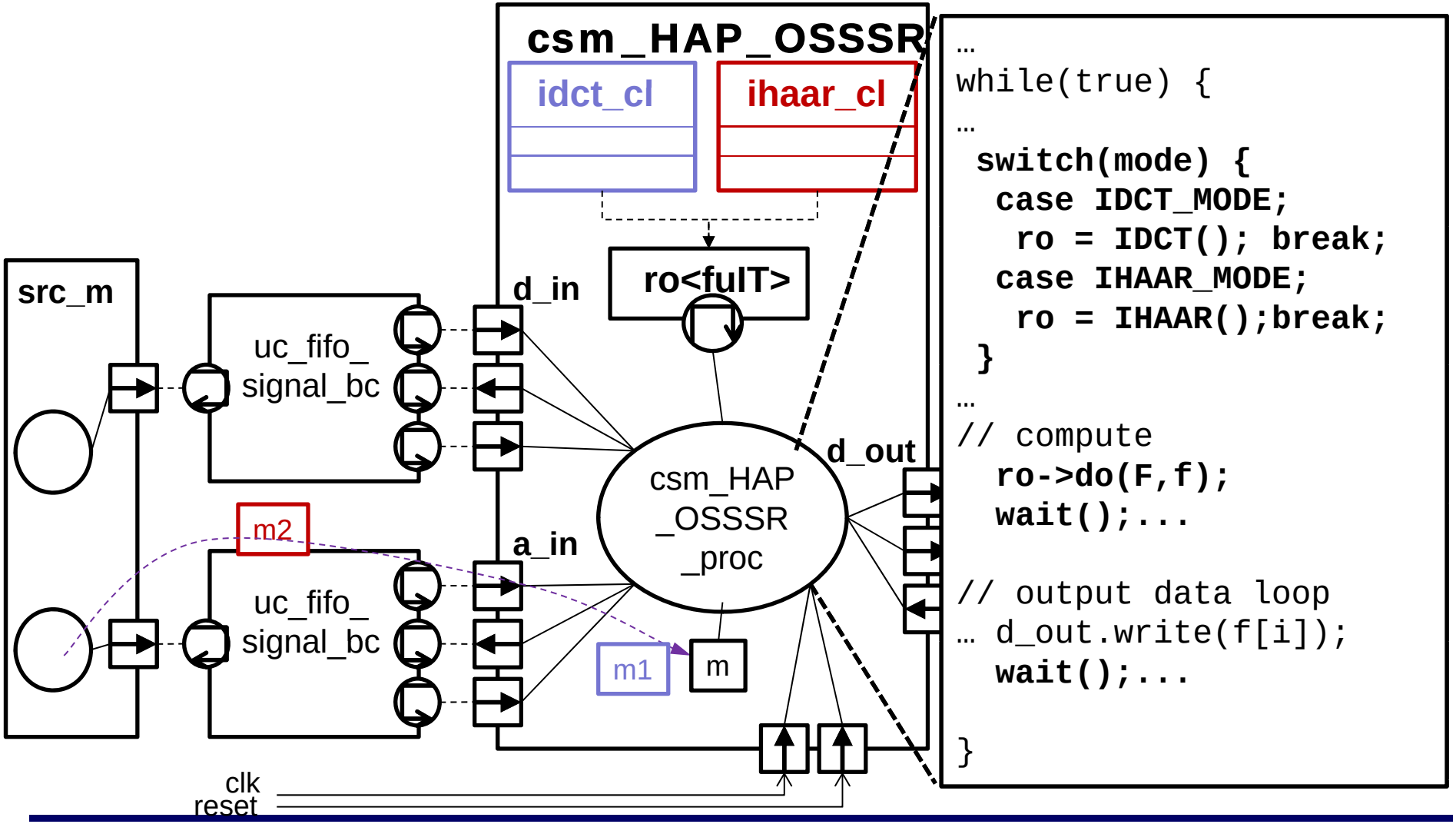
2nd Ref.: Time Domain



3rd Ref.: Reconf. Object Wrapping



3rd Ref.: Reconf. Object Wrapping



4th Refinement: Fitting to Synthesis Subset

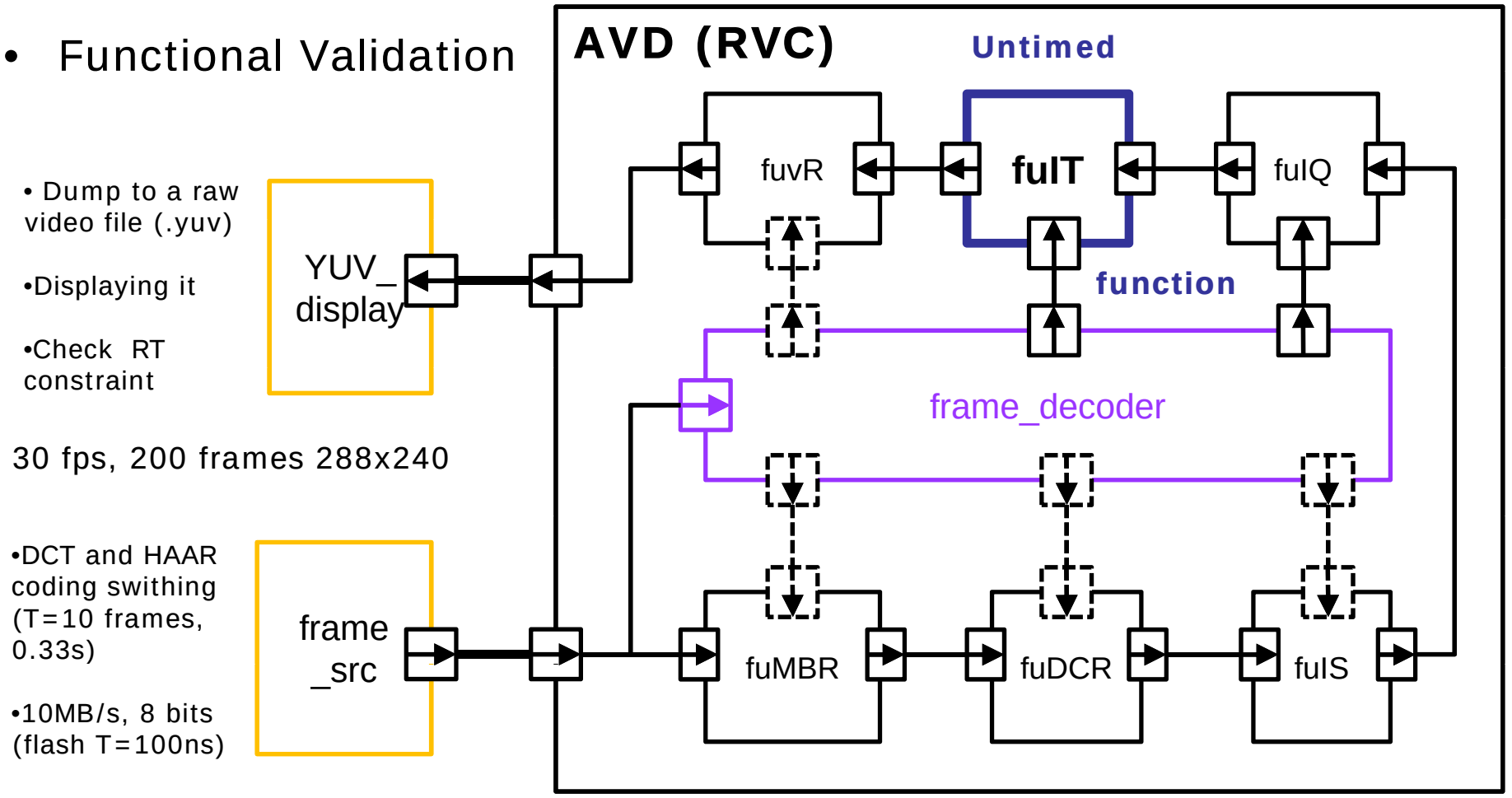
- OSSS+R specific
  - Pass by reference (instead by pointer)
  - Chain of “=“ splitted into single “=“ statements
  - Single compilation unit

- SystemC Synthesys Subset
  - SC_CTHREAD
  - Bounded loops

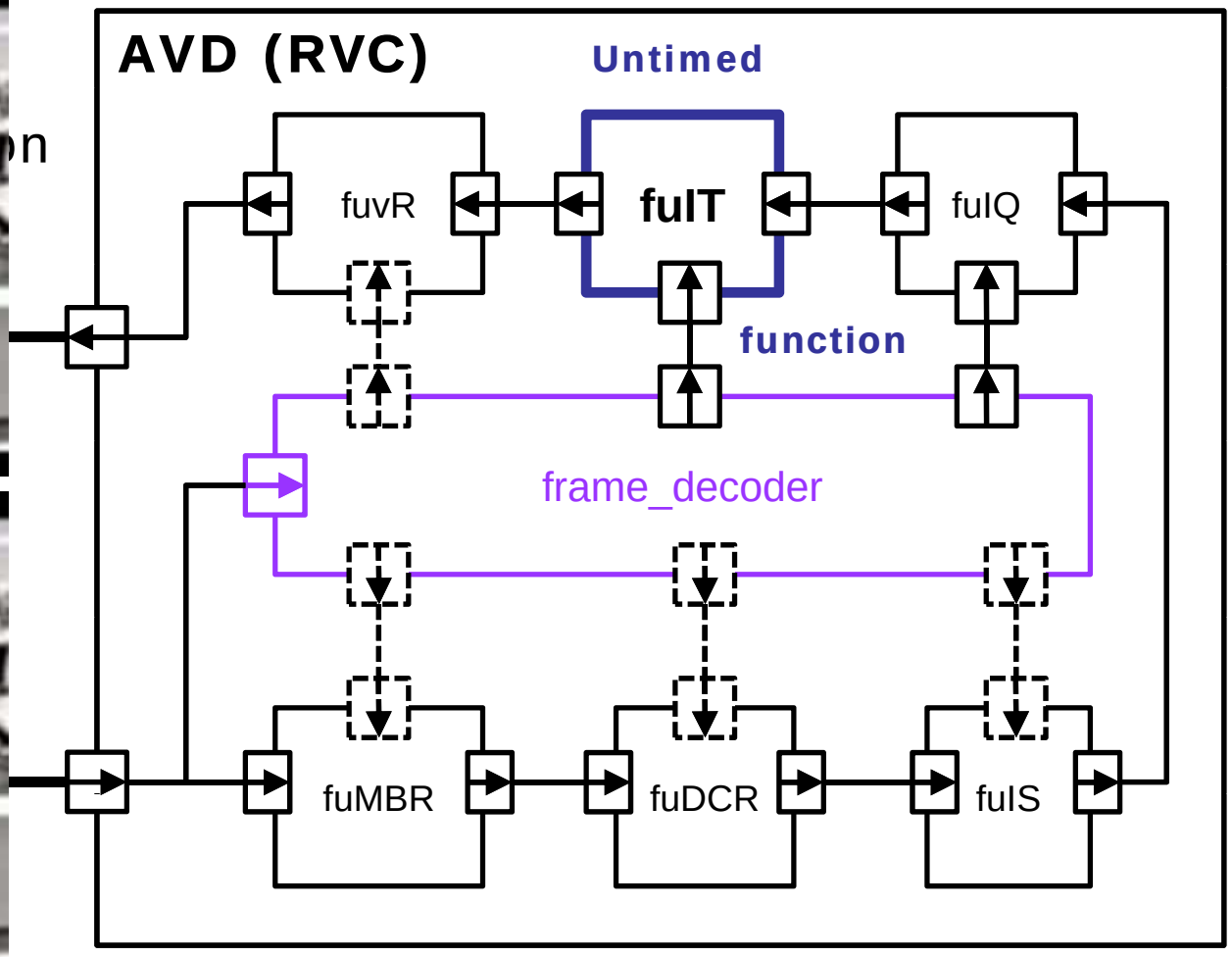
- Code transformations for efficiency

Use Case: fuIT Refinement in AVD

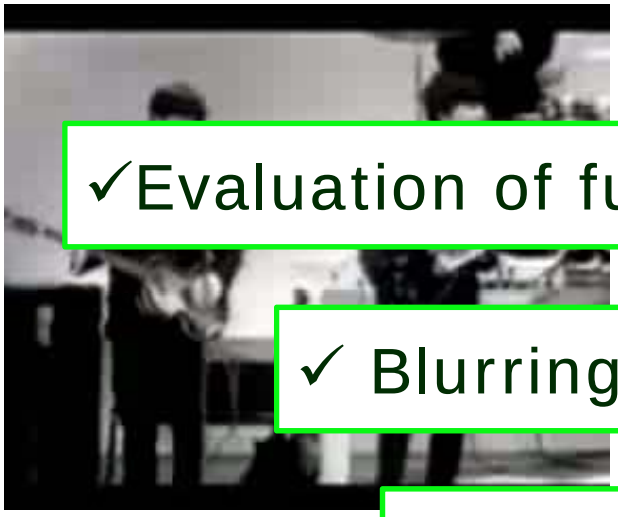
- Functional Validation

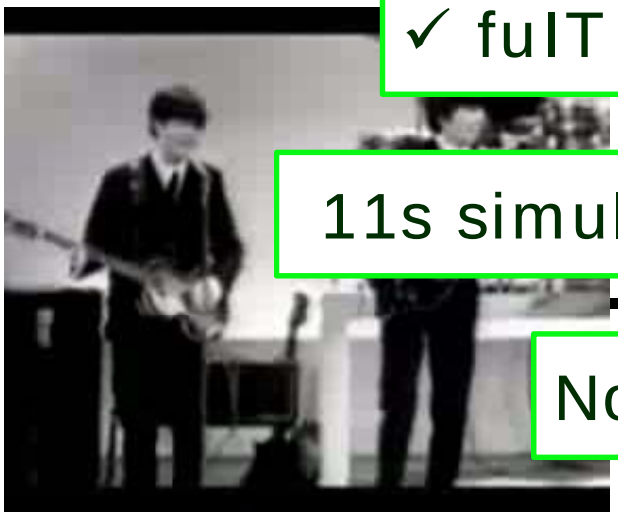


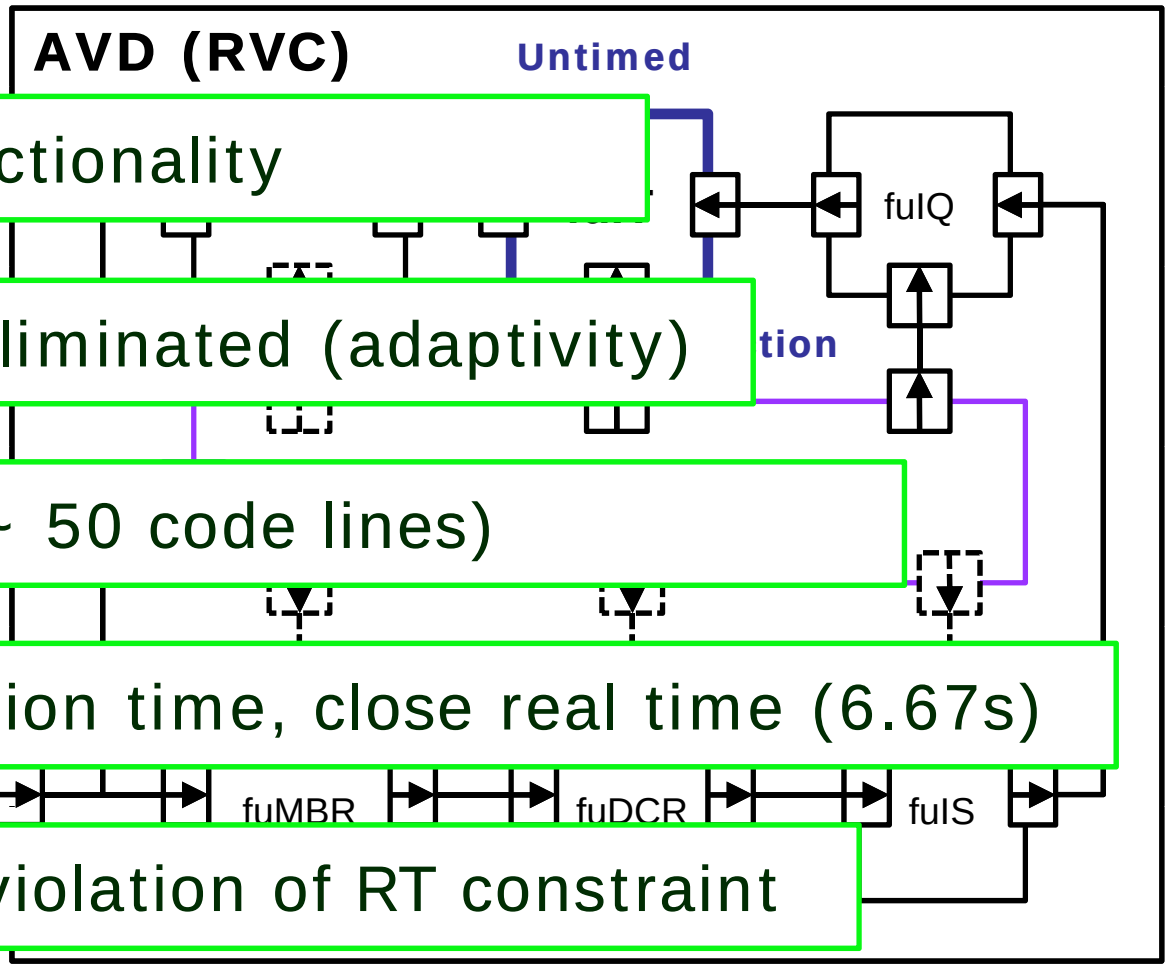
Results: fuIT Refinement in AVD



Results: fuIT Refinement in AVD







✓Evaluation of functionality

✓ Blurring eliminated (adaptivity)

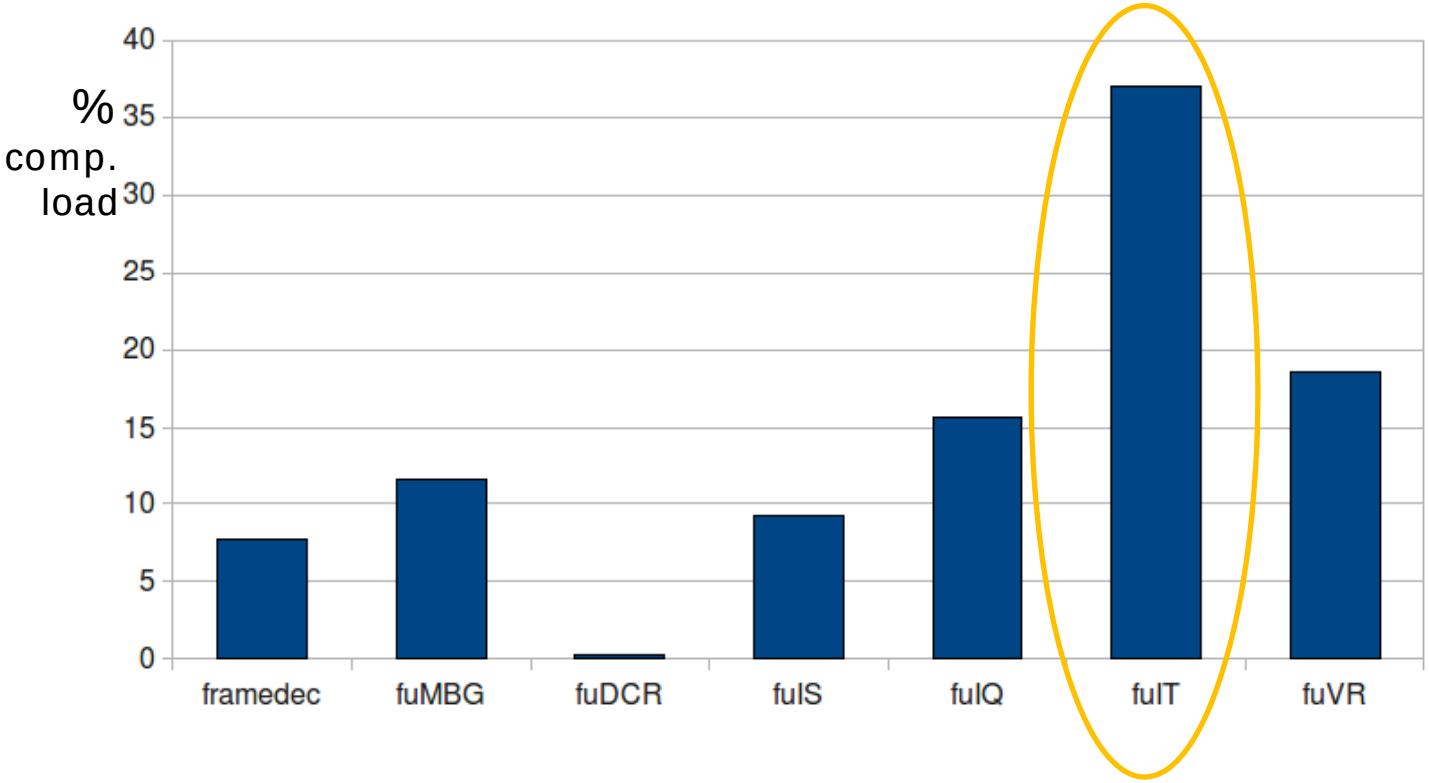
✓ fuIT (~ 50 code lines)

11s simulation time, close real time (6.67s)

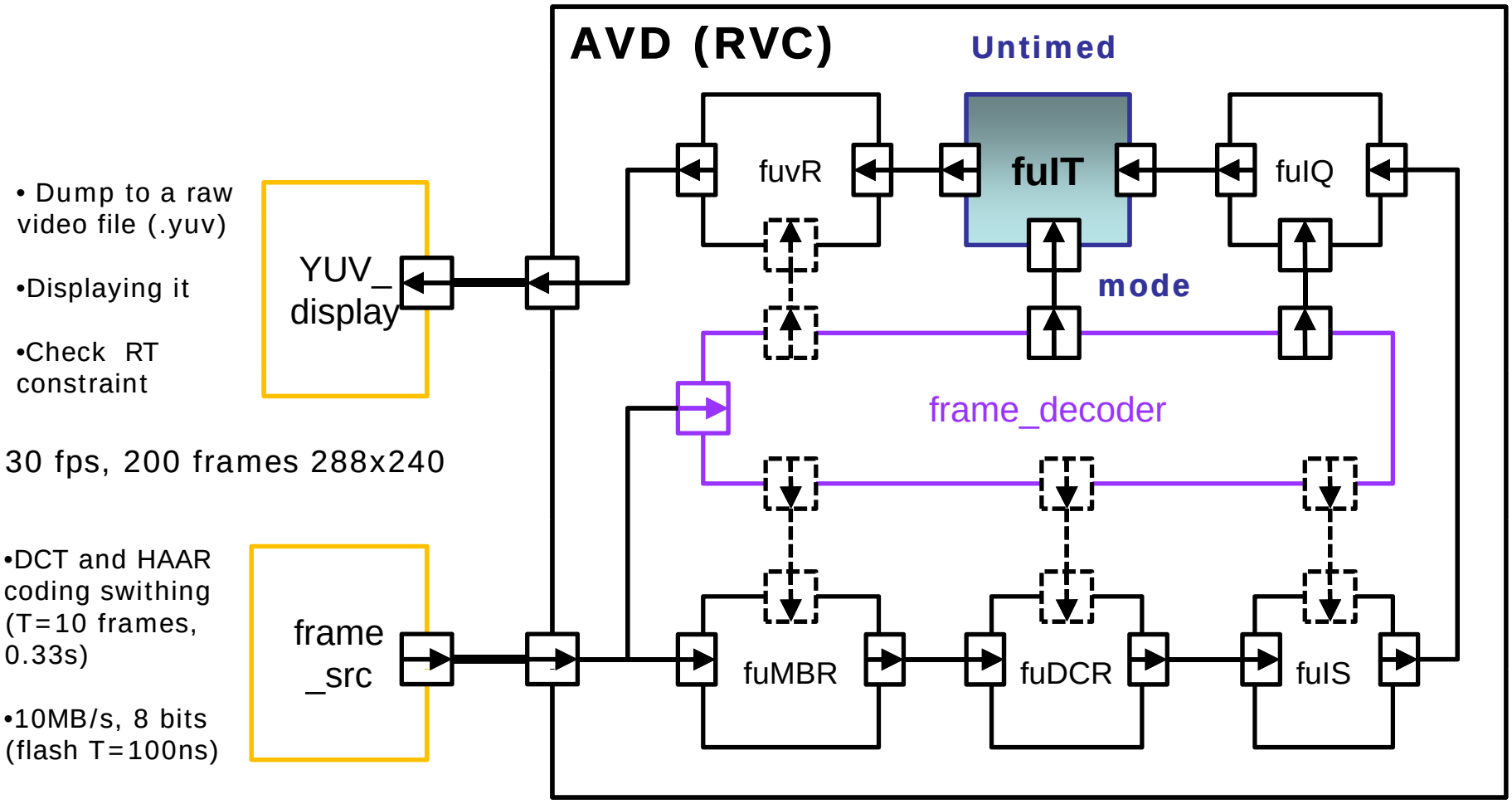
No violation of RT constraint

Results: Computational load

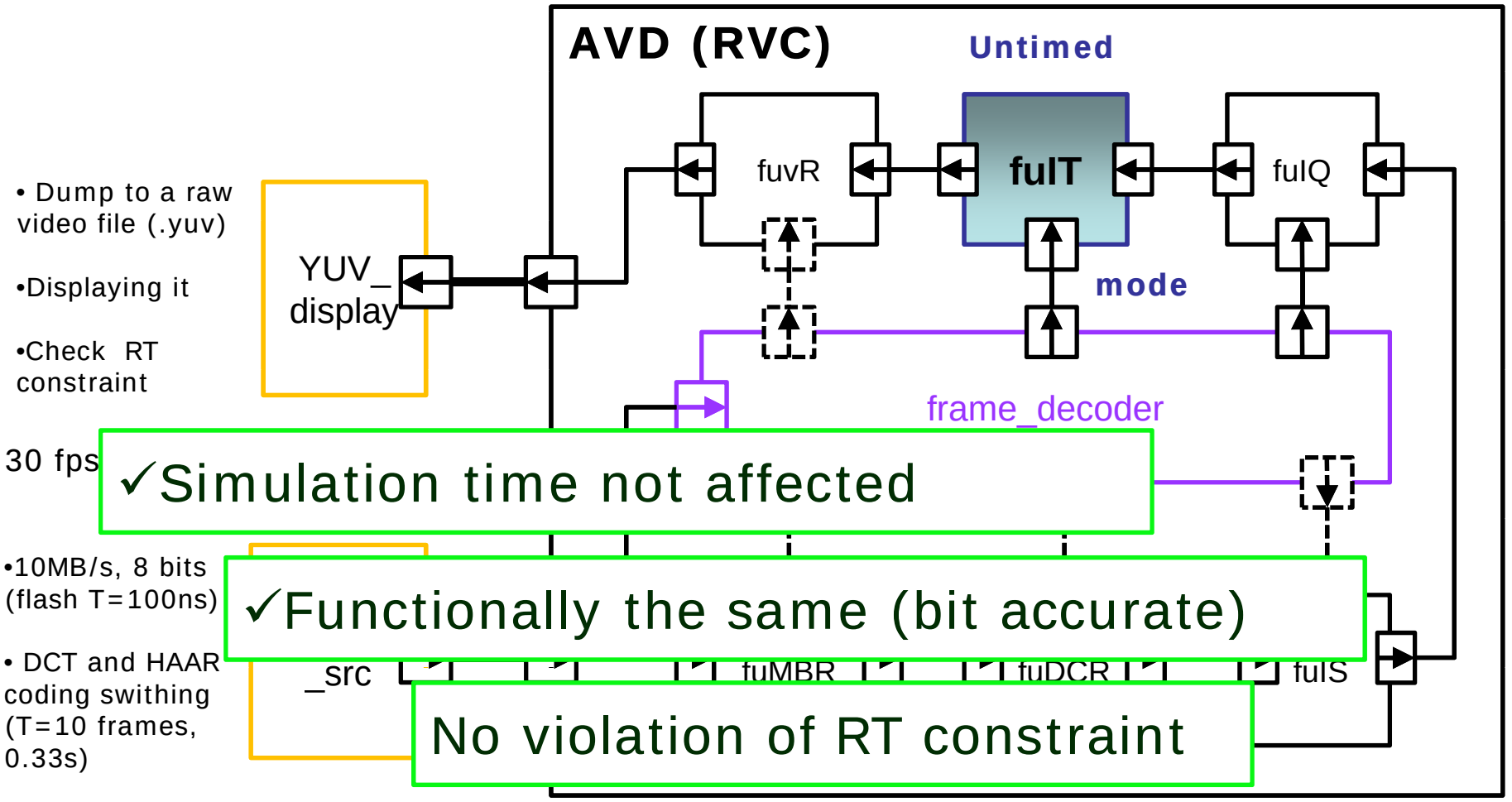
- Refinement of the fuIT worthy



Use Case: R1 Refinement

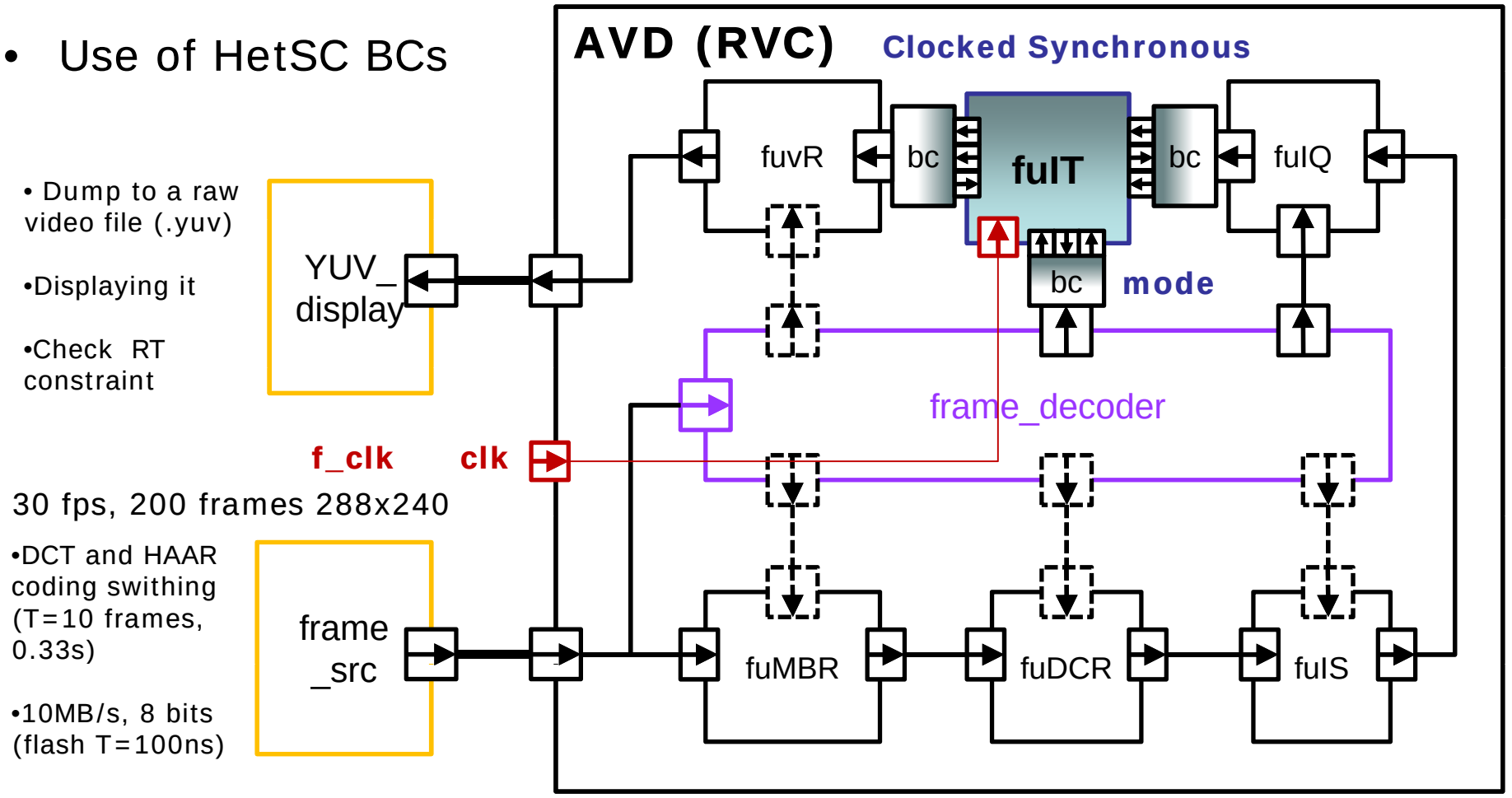


Use Case: R1 Refinement



Use Case: R2 Refinement

- Use of HetSC BCs



• Dump to a raw video file (.yuv)

•Displaying it

•Check RT constraint

30 fps, 200 frames 288x240

•DCT and HAAR coding swithing (T=10 frames, 0.33s)

•10MB/s, 8 bits (flash T=100ns)

Use Case: impact of Time Domain Refinement (R2)

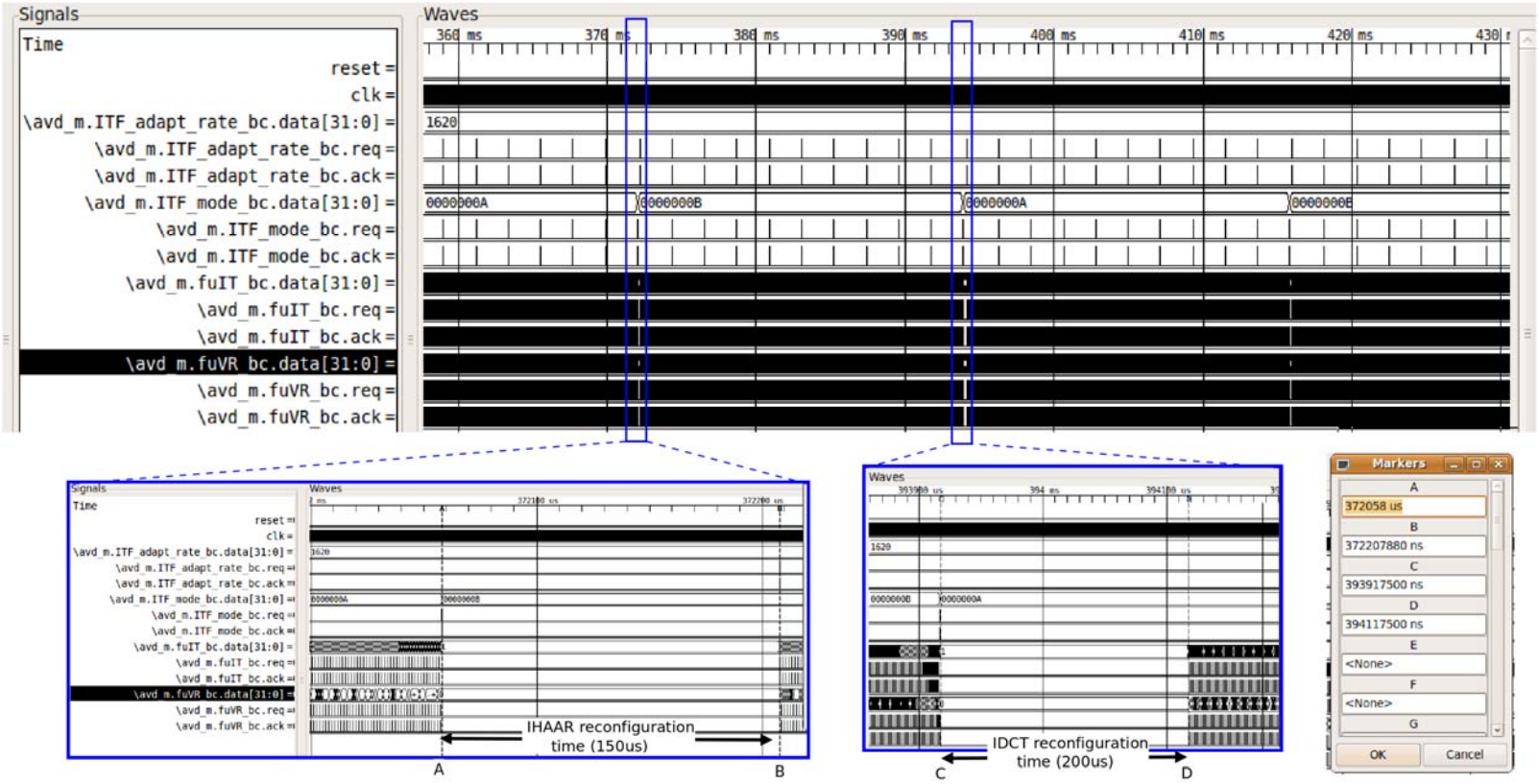
- Study of minimum frecuency for fuIT computational load vs frecuency
  - Assuming a Real Time execution

F(MHz)	100	10	7.5	6.67	6.5
fuIT t(s)	0.44	1.99	5.78	6.65	6.69
fuIT load (%) Vs RT exec.	6.6	29.8	86.6	97.4	>100

- Slow down of simulation speed
  - 81s (almost 8 times!)
  - with wave tracing, 160s (more than 14 times!)

Results: Reconfiguration times after 3rd Refinement

- Reconfiguration times considered in simulation
- In this case, +3.5ms (not noticeable in time performance)
- Simulation performance similar



Results: 4th Ref. Step

- Around 1500 VHDL lines of synthesizable code
- Post-synthesis validation

Conclusions

- Systematic Refinement of A-HetSC HAPs into OSSS+R constructs
- Linking abstract adaptive model with efficient implementation as DRHW
- Refinement in the SystemC domain
  - Smooth approach
  - High-Level analysis

- Future: automation