Methodology and Tools for Performance Analysis of Multiprocessor Embedded Systems

Ismail Assayad and Sergio Yovine

VERIMAG, FRANCE

ARTIST WS 2007, Berlin

Presentation plan

- Motivation
- Current practices
- Related work
- Methodology
- Framework
- Applications
  - Video encoding platform application
- Conclusion

Motivation

- Many embedded applications such as video compression, HDTV and packet routing require higher and higher performance $\Longrightarrow$
  1. Hardware becomes multiprocessor
  2. Software becomes parallel
- Significant growth in the demand and workload of embedded architectures $\Longrightarrow$
  1. Need to be able to predict the mutual impact of software and hardware on their performance
  2. Framework supporting joint (rather than separate) software and hardware analysis

Current practices

- Modelling approaches:
  - Analytical models
  - Simulation
    - Wrapper-based external timing models
    - Annotated timing models
- Modelling scope:
  - Software-based
    - Analysis of hardware performance is not considered
  - Hardware-based
    - Focuses on hardware design without taking into account software development or analysis
  - Platform-based design
    - Abstraction levels for modelling both software and hardware architectures

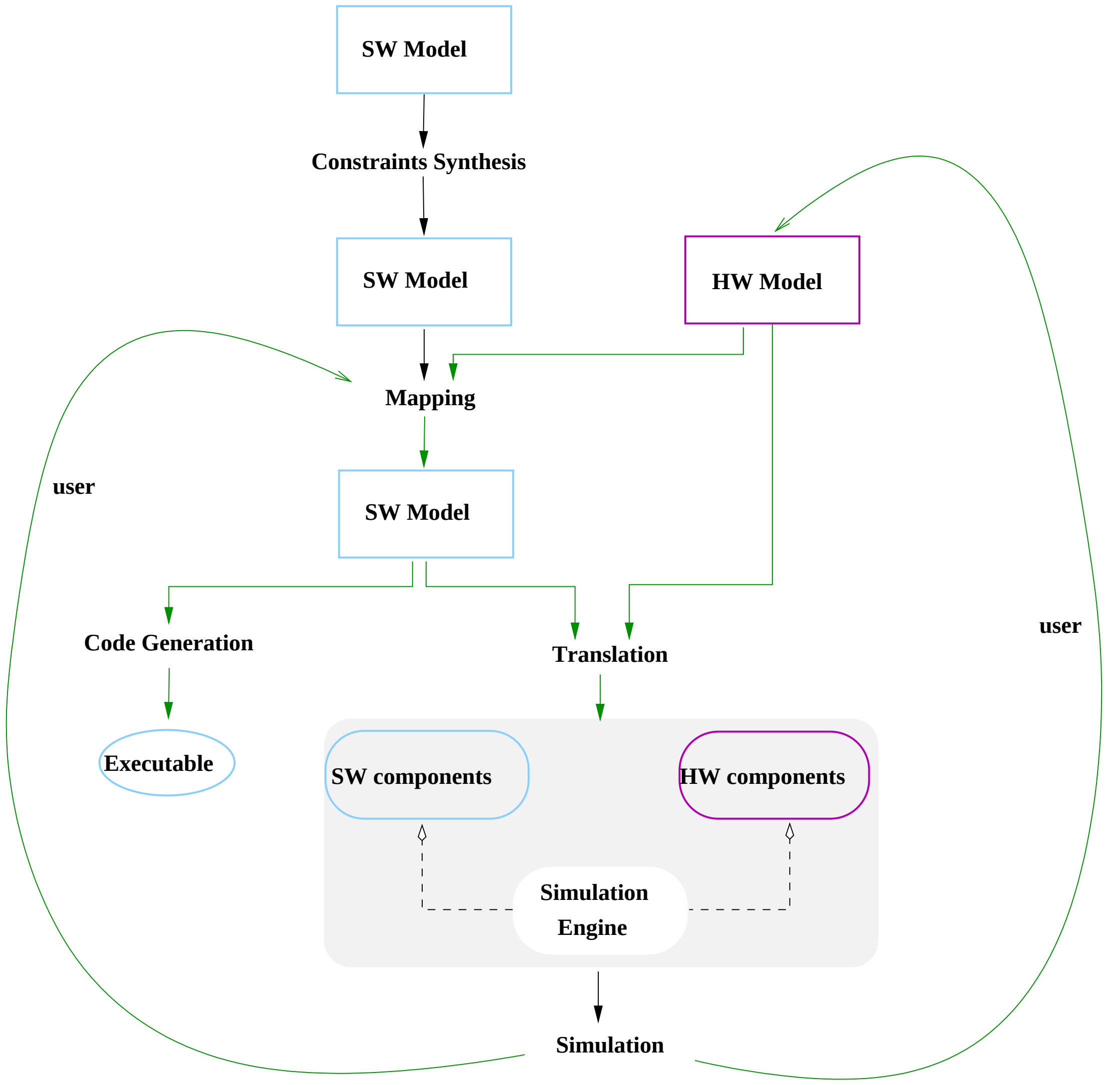
Related work

- Analytical (Thiele et al):
  - Specific to the application domain of packet processing
  - Network calculus theory for reasoning about interleaved streams of packets
- MPARM (Benini et al)
  - Wrapper-based
  - Multiprocessor simulation platform for analysis of hardware design tradeoffs
- ARTS (Madsen et al):
  - Annotated DAG for software and communication latencies for hardware
  - Software timing models are not resolved to micro-architectures ones
- Metropolis (Vincentelli et al), MESH (Paul et al):
  - General-purpose approaches for concurrency modelling at unfixed abstraction level
- SystemC/TLM (Ghenassia et al):
  - Industrial practices use wrapper-based models (PV, PVT) but does not propose a method

Our approach

- Our approach is a simulation and platform based one, it provides:
    - Methodology for concurrency and performance modelling of micro-architectures
    - Modelling hardware at a transaction level and software at a task level
    - Wrapper-less annotated-method based timed models for components
- Advantages:
    - Semantics and methodology for components construction, connexion, and performance prediction
    - Joint software and hardware model-based performance analysis support
- An implementation of our framework is built using SystemC and TLM

Methodology

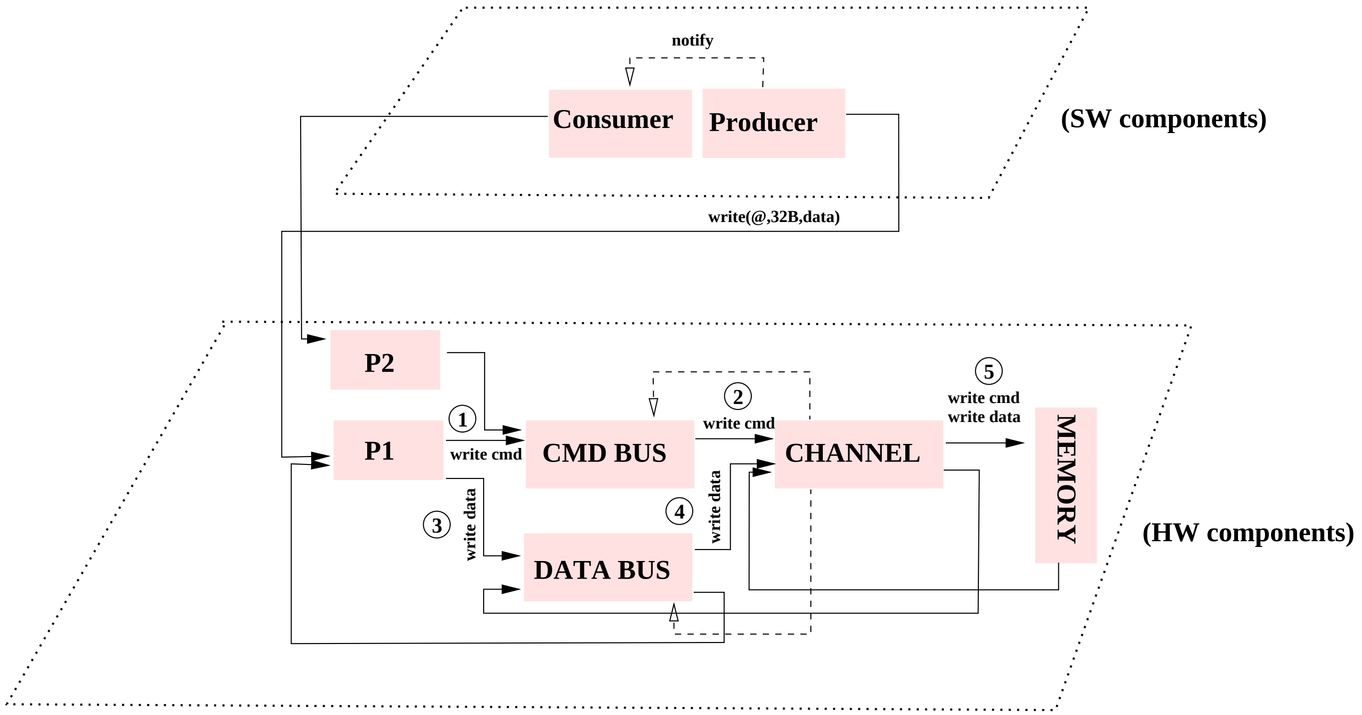


Presentation plan

- Motivation
- Current practices
- Related work
- Methodology
- Framework
  - Modelling
  - Hardware meta-models
  - Example
  - Software meta-models
  - Tools
- Application
- Conclusion

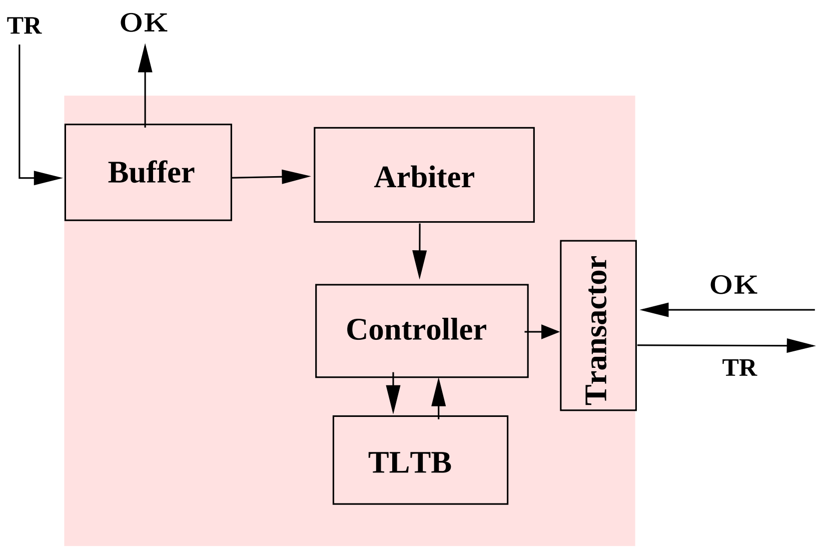
Framework - Modelling

- Components
  - They model transactions behavior of the system
  - They communicate through transaction requests and state-change based events
  - They support profiling of predicted performance (eg. used bandwidth, conflicts, execution and communication times, etc)



Framework - Hardware meta-models

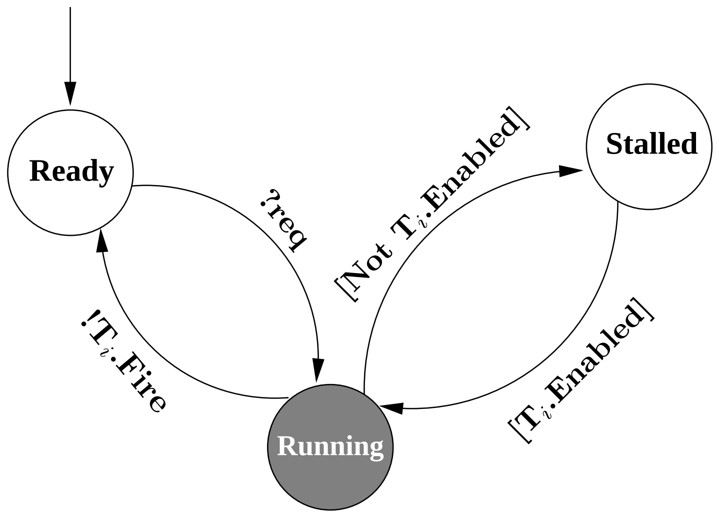
- Component meta-models
  - can be instantiated for modelling hardware micro-architectures at transaction-level
  - are performance-centric and take into account arbitration policies, transaction-level latencies and generated transaction request traffic
  - are composed of following blocks:



Buffered component

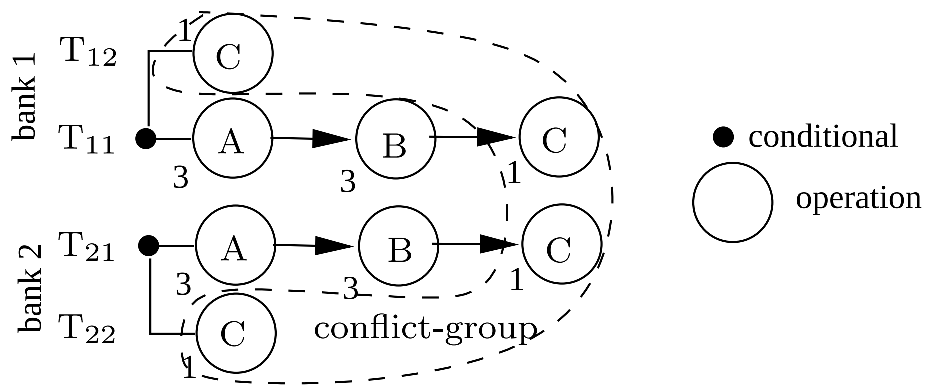
Framework - Hardware meta-models

- Component behavior
  - Blocks are described by automata whose states are instantiated with hardware specific functions
  - Examples: some variables and functions associated to “Running” and “Executing” states of controller and transaction automata are hardware dependent



Controller automaton

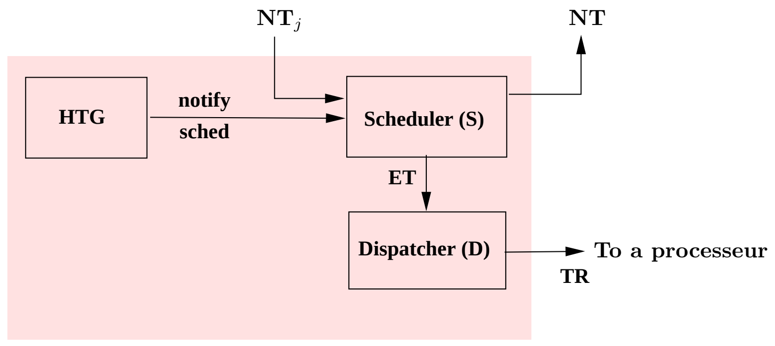
Framework - DRAM example



TLTB block model of a two-bank DRAM (instance of the meta-model)

- In the “Running” state of the controller block:
  - Upon reception of TR $= (i, j, k)$ concerning a memory access for data in column i, row j and bank k do:
    - test if transactions T_{k1} and T_{k2} are enabled
    - if it is the case, fire either T_{k1} or T_{k2} according to the preceding transaction request (TR_k^{last}) of bank b_k:
      - if $\mathrm{TR}_k^{last}.j = \mathrm{TR}.j$ then fire T_{k2}
      - otherwise fire T_{k1}

Framework - Software meta-model



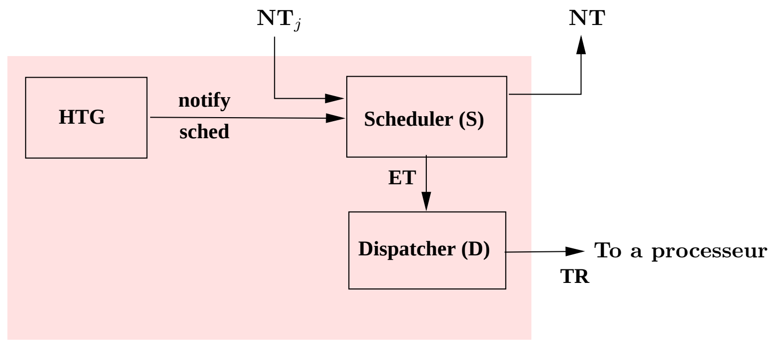
ET: Enabled tasks

NT_j: Completed tasks in other components

Software component

- It is composed of:
  - A hierarchical task graph (HTG)
  - Tasks scheduler
  - Transaction requests dispatcher

Framework - Software meta-model



ET: Enabled tasks

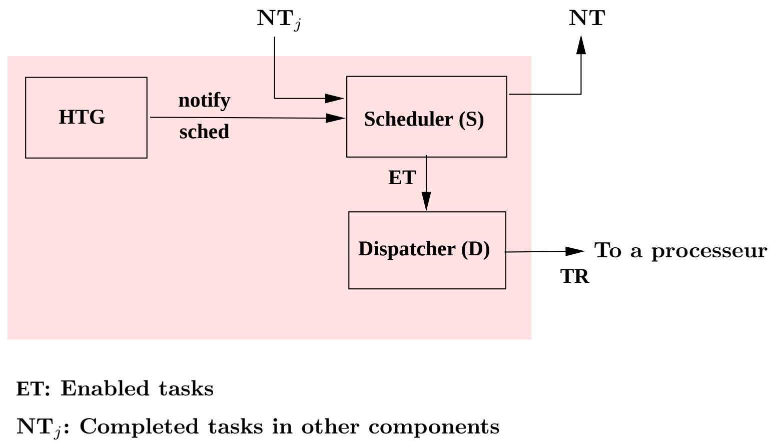
NT$_j$: Completed tasks in other components

Software component

- A task is a sequence of transaction requests, example:

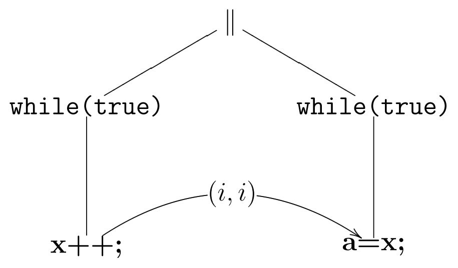
$$\text{« x = x+1 »} \longrightarrow \begin{cases} \text{read(@x, 32B);} \\ \text{[...] //increment x} \\ \text{write(@x, 32B);} \end{cases}$$

Framework - Software meta-model



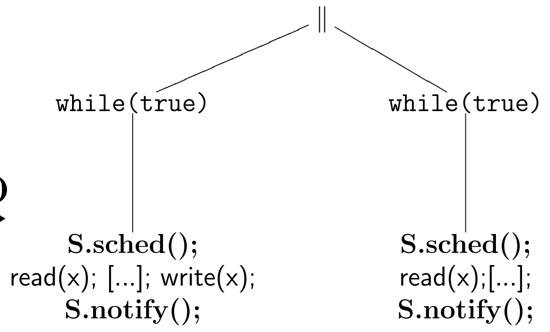
Software component

- Tasks behavior is described using FXML and implemented by the HTG, example:



FXML

$\xrightarrow{\text{Translated to}}$



HTG

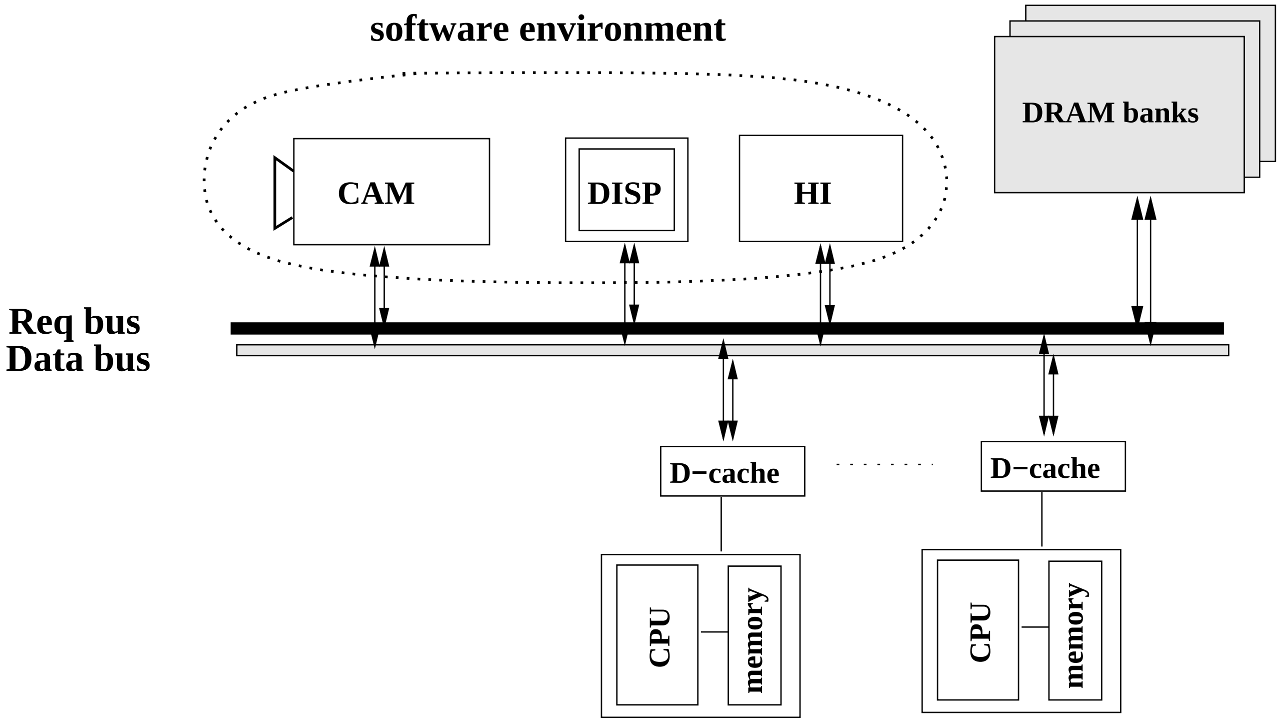
Framework - Tools

- Jahuel:
  - Describes software and hardware models in FXML
  - Synthesizes executable code for software model (eg : C+posix)
- P-Ware:
  - Takes hardware and software meta-models instances as input from Jahuel
  - Provides joint software and hardware performance prediction by simulation

Presentation plan

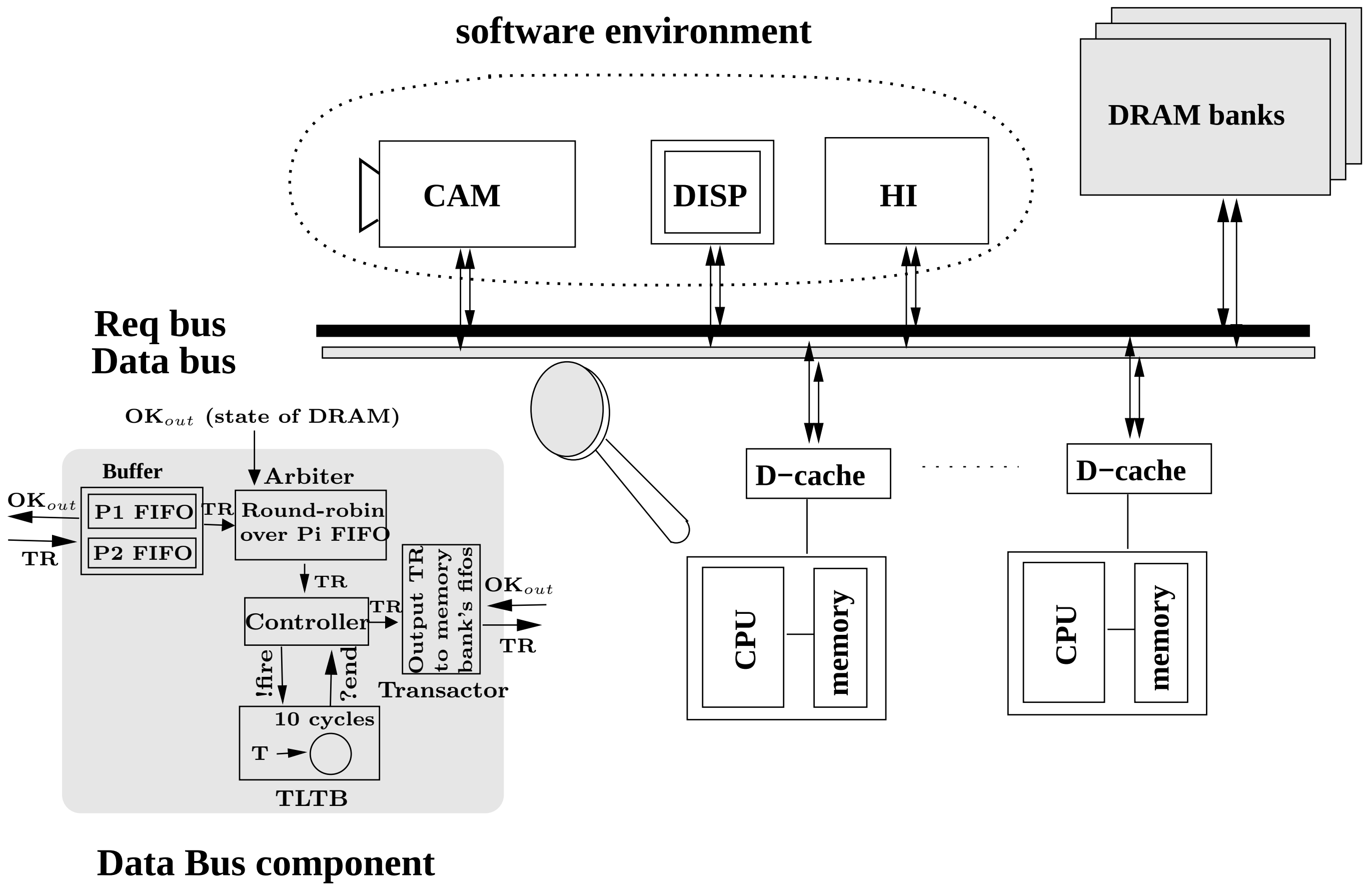
- Motivation
- Current practices
- Related work
- Methodology
- Framework
- Video encoding platform application
  - Constraints synthesis results
  - VE Performance results
- Conclusion

VE platform - Hardware components



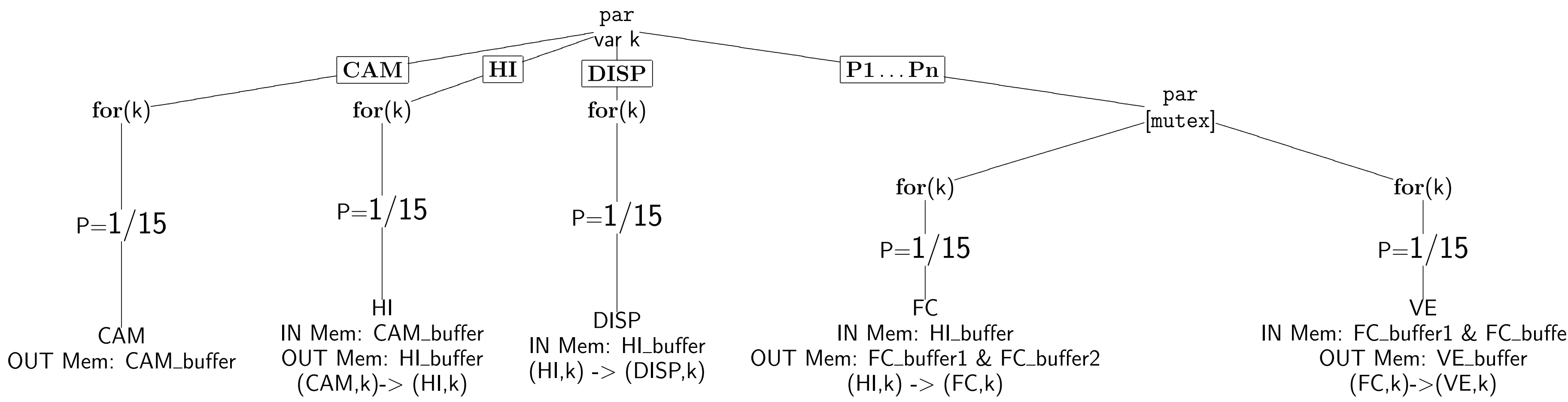
Hardware architecture components

VE platform - Hardware components



Hardware architecture components

VE platform - System constraints

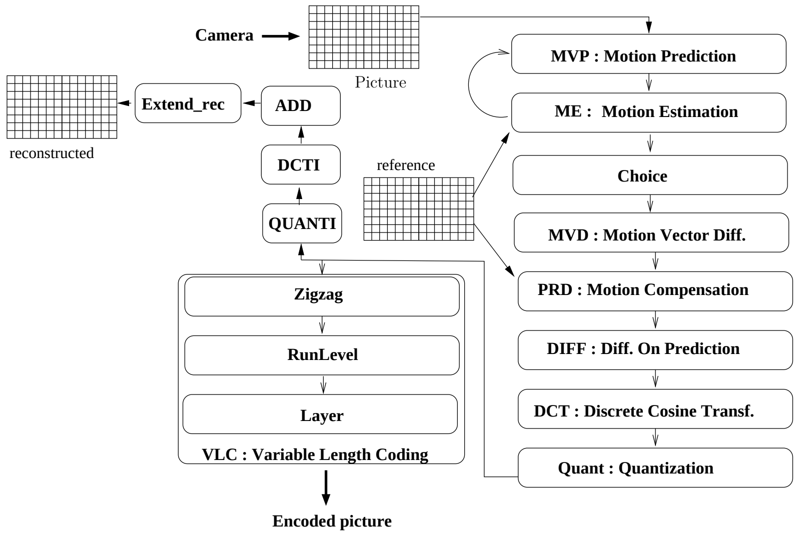


- Taking into account WCET of CAM, HI, and DISP ($\delta_{\mathrm{CAM}} = \delta_{\mathrm{HI}} = \delta_{\mathrm{DISP}} = \frac{1}{30}$) we synthesize:

$$\begin{cases} b_{\mathrm{HI}} = b_{\mathrm{CAM}} + \frac{1}{30} & \text{HI activated } \frac{1}{30} \text{ after CAM} \\ b_{\mathrm{DISP}} = b_{\mathrm{HI}} + \frac{1}{30} & \text{DISP activated } \frac{1}{30} \text{ after HI} \\ b_{\mathrm{FC}} = b_{\mathrm{HI}} + \frac{1}{30} & \text{FC activated } \frac{1}{30} \text{ after HI} \\ b_{\mathrm{VE}} = b_{\mathrm{FC}} + \frac{1}{30} & \text{VE activated } \frac{1}{30} \text{ after FC} \\ \delta_{\mathrm{FC}} + \delta_{\mathrm{VE}} \leq \frac{1}{15} & \text{Execution time of VE and FC smaller than } \frac{1}{15} \end{cases}$$

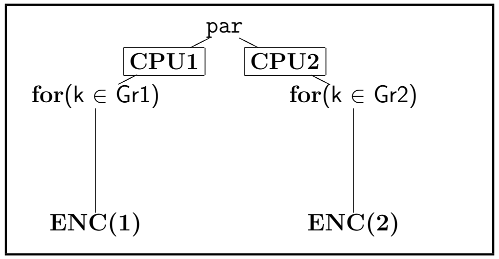
- With a $\frac{2}{3} \times \frac{1}{25} s$ format converter ($\delta_{\mathrm{FC}} = \frac{2}{3} \times \frac{1}{25}$): $\boxed{\delta_{\mathrm{VE}} \leq \frac{1}{25}}$

Implementations of the MPEG-4 VE

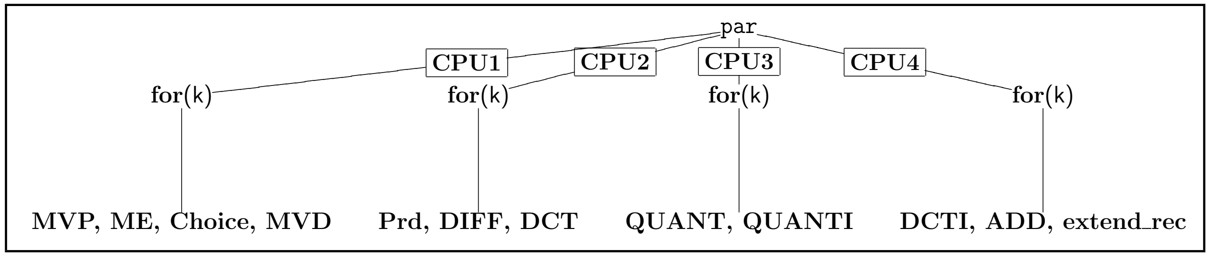


MPEG-4 VE block diagram

- Two Implementations of ENC:



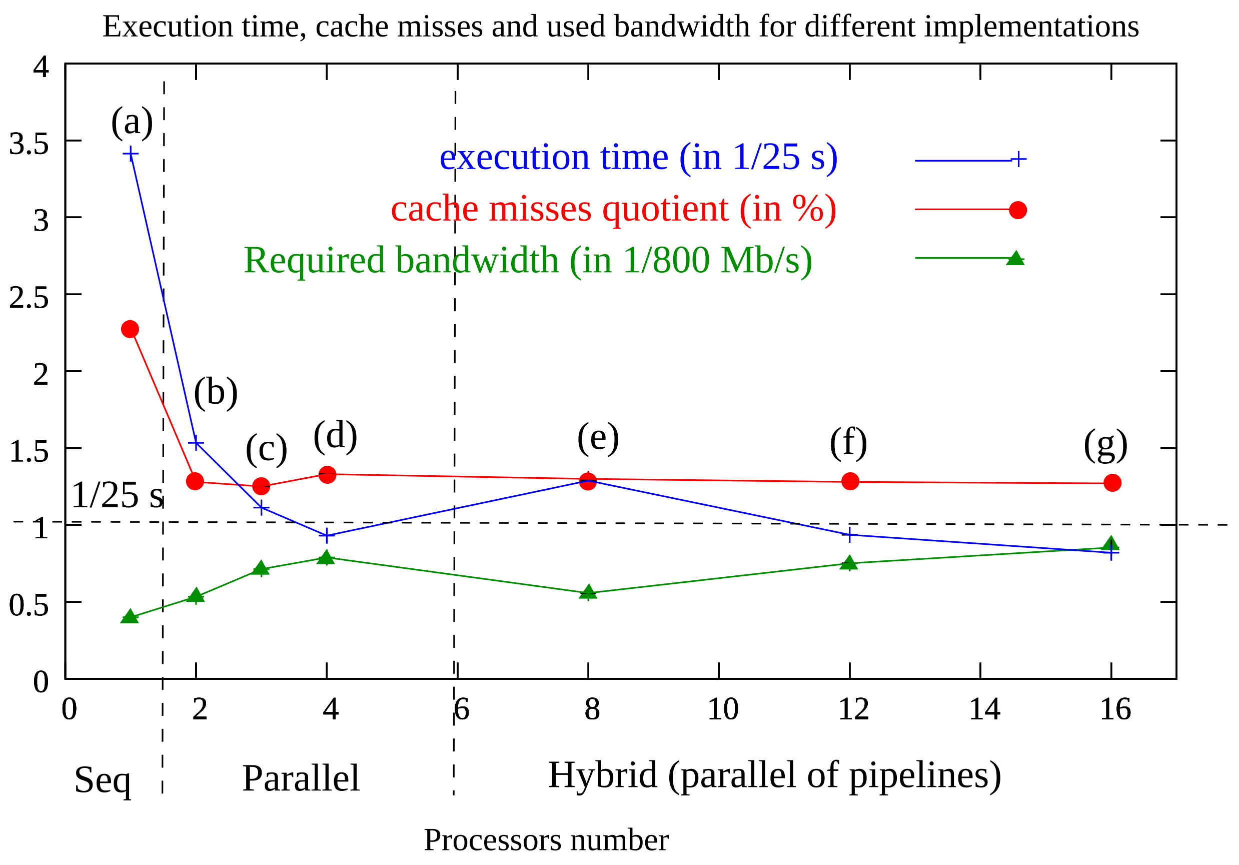
HTG with a 2-thread partition of ENC



HTG with a pipeline partition of ENC

Performance results of VE

- Results using P-Ware:



- (d), (f) and (g) satisfy execution time constraints
- Hybrid ones, i.e. (f) and (g), produce an increase of bandwidth usage
- The best compromise seems to be (d), consisting of 4 MPEG-threads

Conclusion - Framework

- Component-based modelling framework combining transaction-level HW and programmer-level SW models
- Joint HW and SW modelling and performance analysis allowing for predicting:
  1. Impact of HW on SW performance
  2. Ability of HW to accommodate future services
- A programming and simulation tools supporting the framework
  1. Jahuel
  2. P-Ware

Conclusion - Applications

- Application to real-life industrial systems:
  - MPEG-4, IPv4, Philips WASABI NoC, and Intel's IXP2800
- Models expressiveness:
  - Data granularity is easily set-up using the imlementation of software dispatchers and/or hardware transactors: bus-packet (eg. a line of pixels) suited for MEPG-4, and bus-size for IPv4.
- Tool performance:
  - Scalable prototype: eg. dual IXP NP with **768** memory bank component
  - Fast simulation speeds: average **300 000** cycles/s
- Models precision:
  - Precise performance results: eg. values are within **5%** of the ones obtained by the IXP2800 cycle-accurate simulator for several data granularities
  - Correct performance trends
- Joint software and hardware modelling environment:
  - Automatization using Jahuel
  - Fast and efficient system design with user-driven joint software and hardware performance tracking using P-Ware

Thank you!