Functional modeling style for efficient SW code generation of video codec applications

Sang-Il Han[1)2)] Soo-Ik Chae[1)] Ahmed. A. Jerraya[2)]

SD Group[1)] SLS Group[2)]

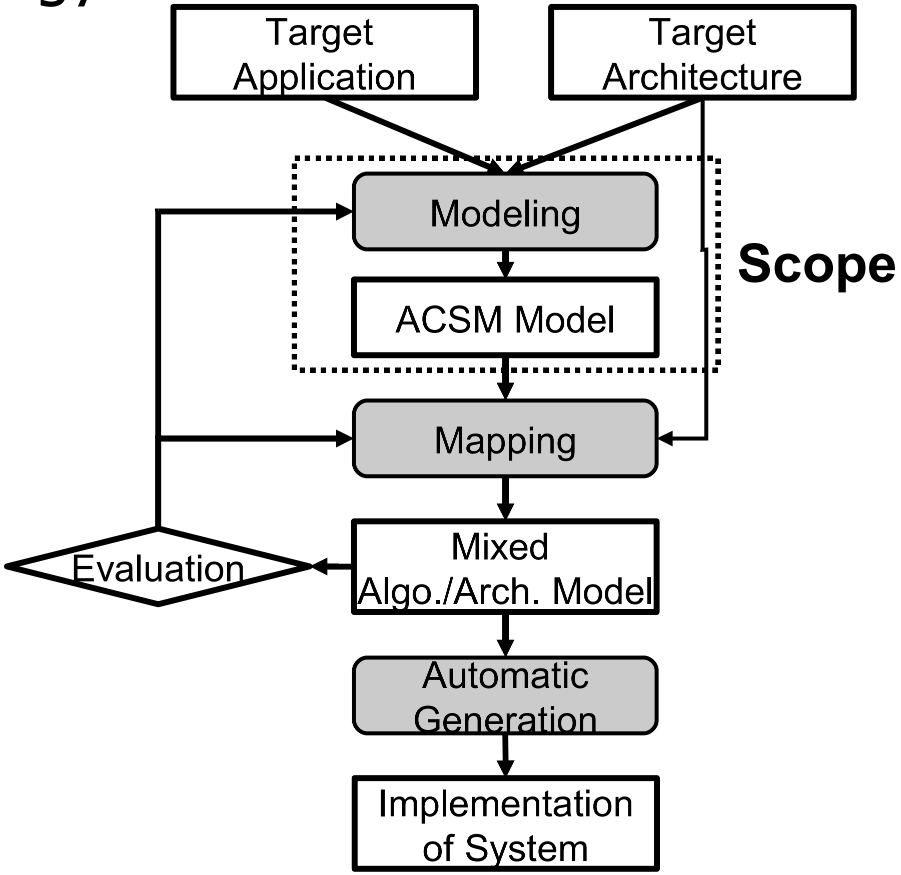
Seoul National Univ., Korea TIMA laboratory, France

Summary

- ❑ **Challenges** in video codec system design
  - ✓ Complex High-performance within short design time
  - ✓ Classical RTL design methodology is too slow
- ❑ **Requirements** in video codec system design
  - ✓ System level design methodology with functional model
  - ✓ Explicit **parallelism** and **conditional** in functional model
- ❑ **Existing modeling styles**
  - ✓ Data-driven model lacks explicit **conditional** (ex: SDF lacks conditionals)
  - ✓ Event-driven model lacks explicit **parallelism** (ex: SM lacks distributed imp.)
- ❑ **Proposed modeling style**
  - ✓ Abstract clock synchronous model (ACSM): An extension of CSM for RTL
  - ✓ Both **parallelism** and **conditional** with an abstract global clock

The scope of this work

- ❑ System-level design methodology



- ❑ Scope
  - ✓ Modeling style to generate efficient SW code.

Contents

❑ **Introduction**

❑ Proposed modeling style

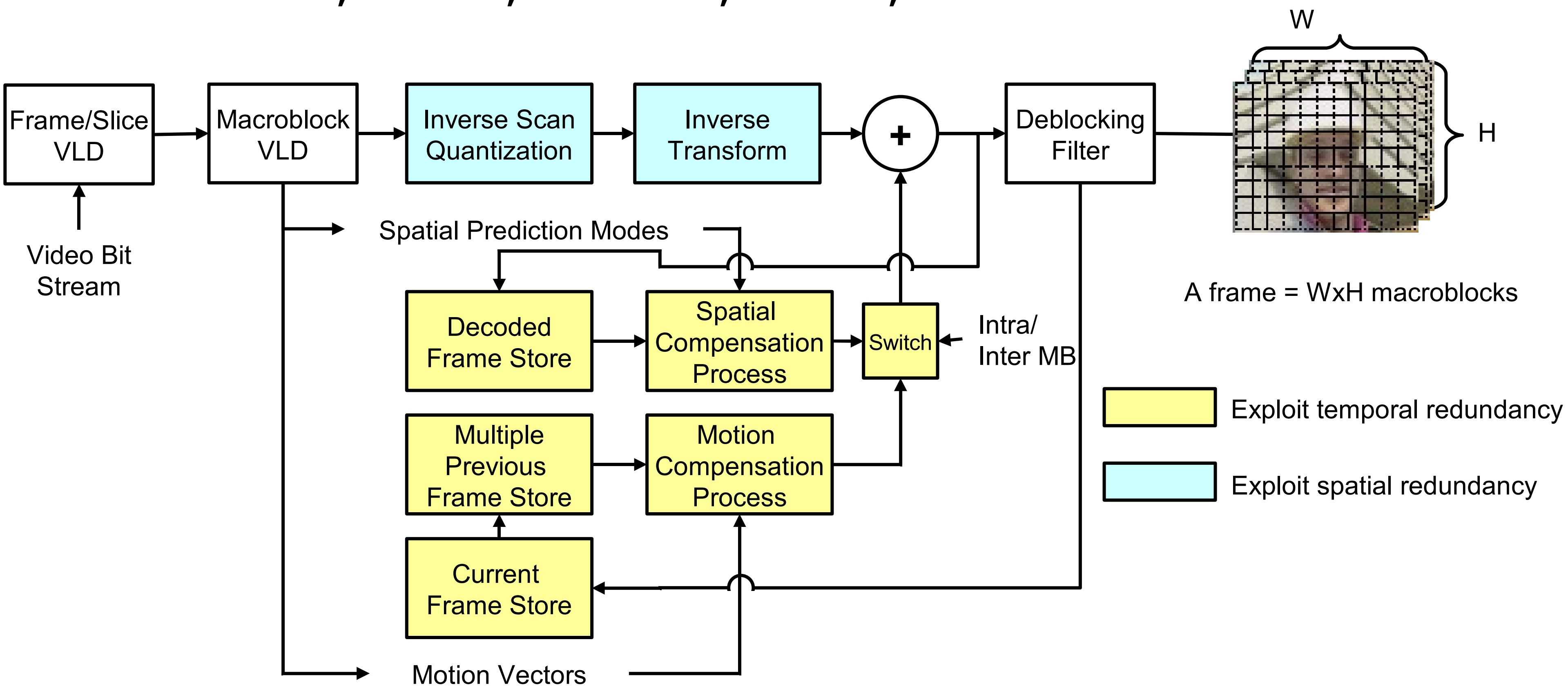
❑ Comparison with existing modeling styles

❑ Experiment

❑ Conclusion and Perspective

Target application

- ❑ Macroblock-based video codecs
  - ✓ MPEG-2, H.263, MPEG-4, H.264, and WMV9


W
H
Frame/Slice VLD
Video Bit Stream
Macroblock VLD
Inverse Scan Quantization
Inverse Transform
+
Deblocking Filter
Spatial Prediction Modes
Decoded Frame Store
Spatial Compensation Process
Switch
Intra/ Inter MB
Multiple Previous Frame Store
Motion Compensation Process
Current Frame Store
Motion Vectors
A frame = WxH macroblocks
Exploit temporal redundancy
Exploit spatial redundancy

Block diagram of an H.264 decoder

Challenges and Requirements (1)

- ❑ **1) Computation Complexity**
  - ✓ Ex: ≈ 10 Gops/sec for 4VGA decoding
- ➢ **Explicit parallelism**
  - ✓ Parallel and pipelined computation execution

- ❑ **2) Communication Complexity**
  - ✓ Ex: ≈ 500 MB/sec external memory bandwidth for 4VGA decoding
- ➢ **Explicit and predictable communication**
  - ✓ Parallel computation and communication execution
  - ✓ Efficient communication scheme e.g. burst transfer

Challenges and Requirements (2)

- ❑ **3) Specification Complexity**
  - ✓ Ex: Curr. MB has dependency with prev., upper., upper prev. MB
- ➢ **Higher level synchronization**
  - ✓ Function- and thread-level synchronization
- ❑ **4) Control Complexity**
  - ✓ Ex: 4x4 intra, 16x16 intra, 4x4 inter, 8x8 inter, … MB prediction modes
- ➢ **Explicit data-dependent comp./comm.**
  - ✓ Sophisticated control operations e.g. memory management

Contribution

- ❑ **Propose modeling style for video codecs**
  - ✓ Abstract clock synchronous model (ACSM)
  - ✓ An extension of clocked synchronous model for RTL
  - ✓ Coarser clock: Abstract clock

- ➢ **1)Partially ordered dependency at function level**
  - ✓ Computation Complexity,
- ➢ **2)Explicit comm. channel with fixed buffer size**
  - ✓ Communication Complexity
- ➢ **3)Coarser clock granularity than physical clock**
  - ➢ Specification Complexity
- ➢ **4)Explicit conditionals through global abstract clock**
  - ➢ Control Complexity

Contents

□ Introduction

□ **Proposed modeling style**

□ Comparison with existing modeling styles

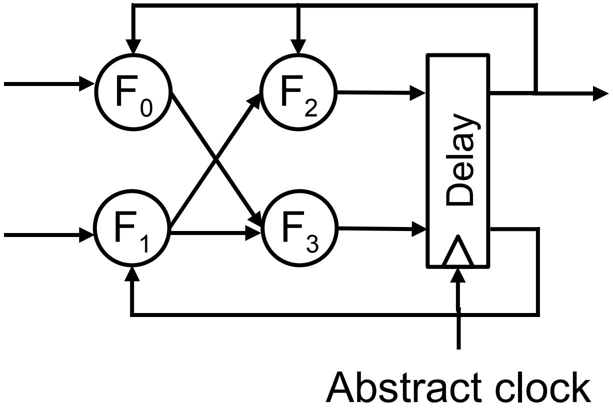
□ Experiment

□ Conclusion and Perspective

Proposed modeling style

❑ Abstract clock synchronous model

- ✓ Structure
  - Network of state-less functions
  - Memory elements
  - Abstract clock

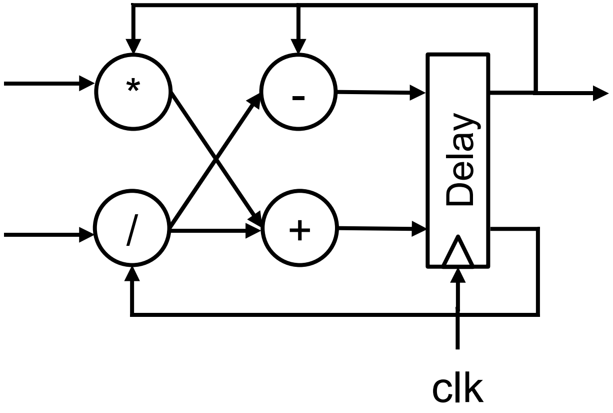


- ✓ Hypothesis : Lock step execution model
  - In every abstract clock cycle, new values propagate in the network

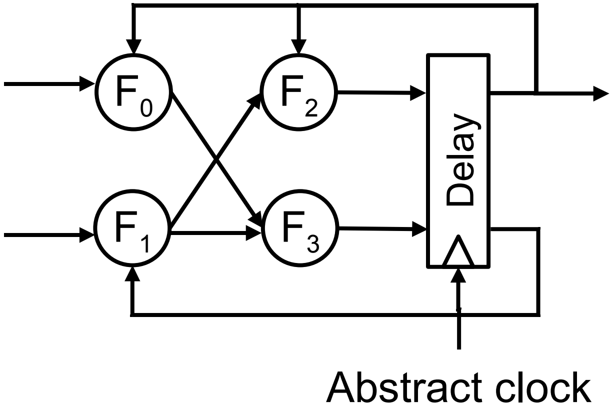
RTL model vs ACSM

Difference: Granularity of clock and components

RTL model



ACSM



❑ Network of combinational gates

❑ Memory elements (delay)

❑ No loops in combinational logics

❑ Synch: Lock step model

❑ Synch. interval: clock

❑ Network of state-less functions

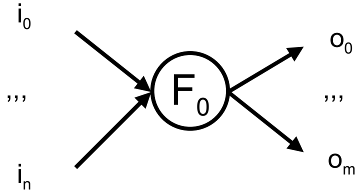
❑ Memory elements (delay)

❑ No loops in network of functions

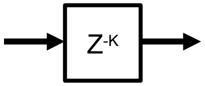
❑ Synch: Lock step model

❑ Synch. interval: abstract clock

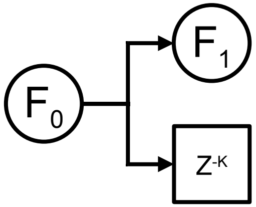
Basic components



(a) module

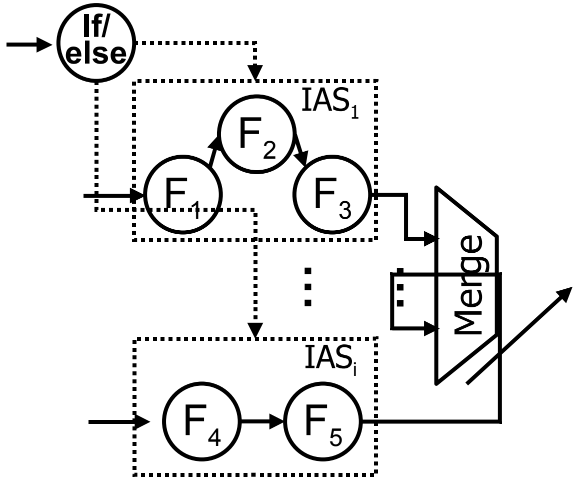


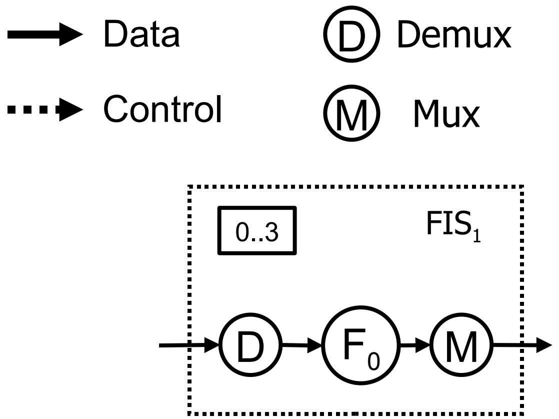
(b) delay



(c) arc

token = (tag, value)





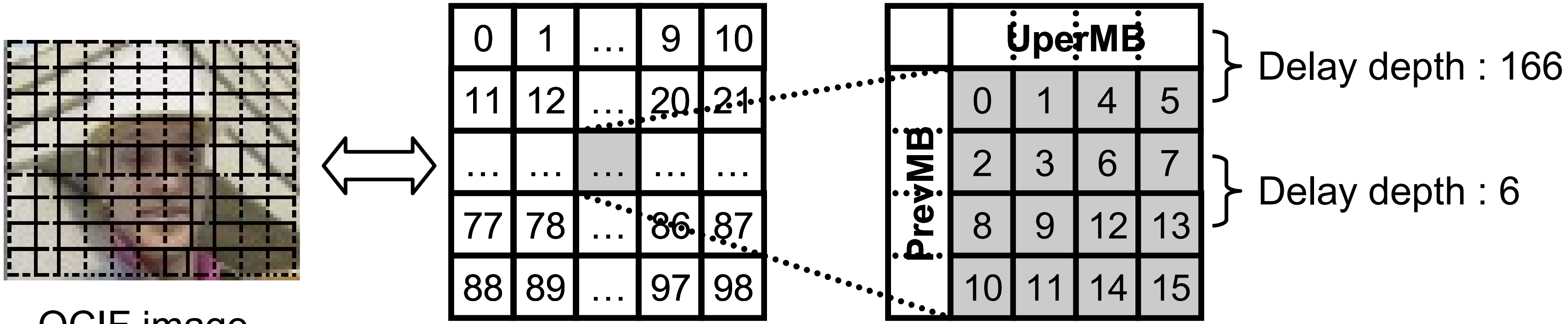
(d) If-action subsystem (IAS)

(e) For-iterator subsystem (FIS)

- □ **Rule on execution**: If one token on **all** input ports, fired.
  - ✓ Except Merge block
- □ **Rule on absent token**: absent token with global abstract clock.
  - ✓ absent token = (tag, empty value)

Abstract clock for video codecs

- ❑ Hierarchical iterations in video decoder and encoder
  - ✓ Frame level index: too coarse to express parallelism.
  - ✓ Sub-macroblock level index: irregular delay depth
  - ✓ Macroblock level index: proper granularity and regular delay depth



QCIF image

0	1	…	9	10
11	12	…	20	21
…	…	…	…	…
77	78	…	86	87
88	89	…	97	98

(a) Decoding order of macroblock

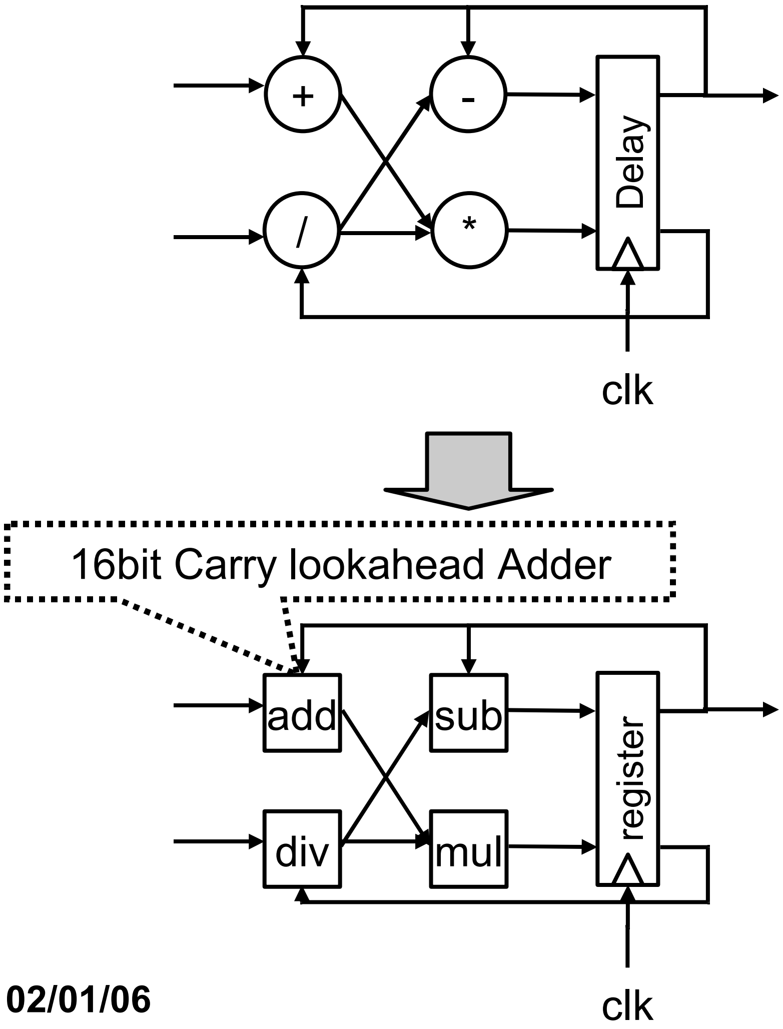
	UperMB			
PrevMB	0	1	4	5
	2	3	6	7
	8	9	12	13
	10	11	14	15

(b) Decoding order of 4x4 block

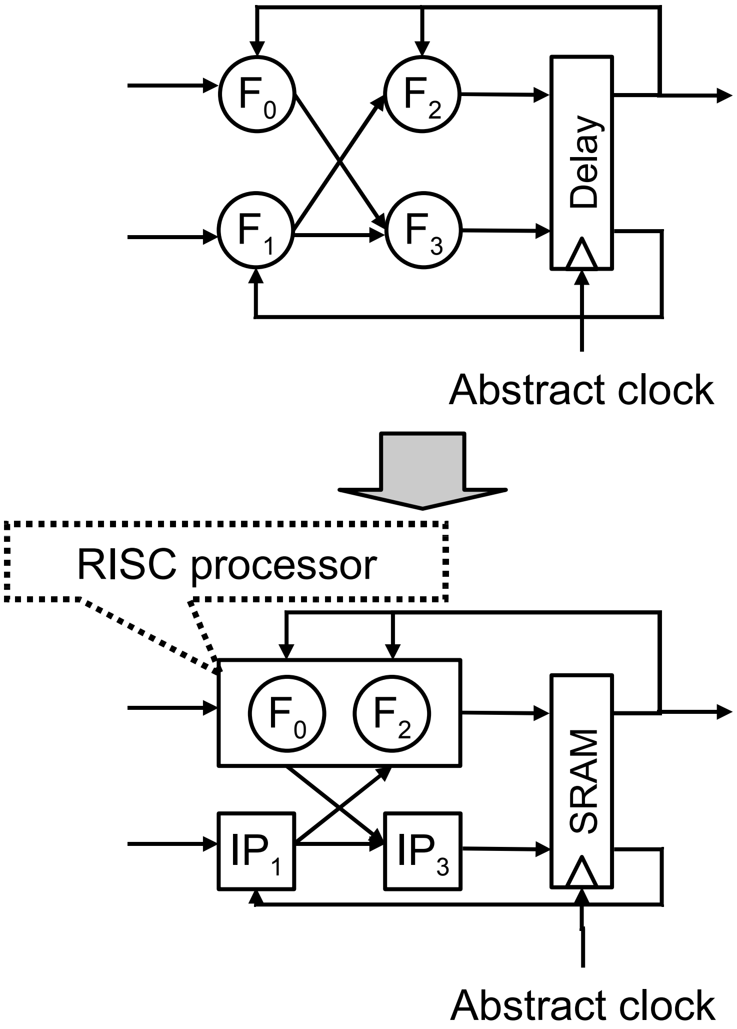
➡ **Abstract clock for video codecs: Macroblock index**

Implementations of RTL model and ACSM

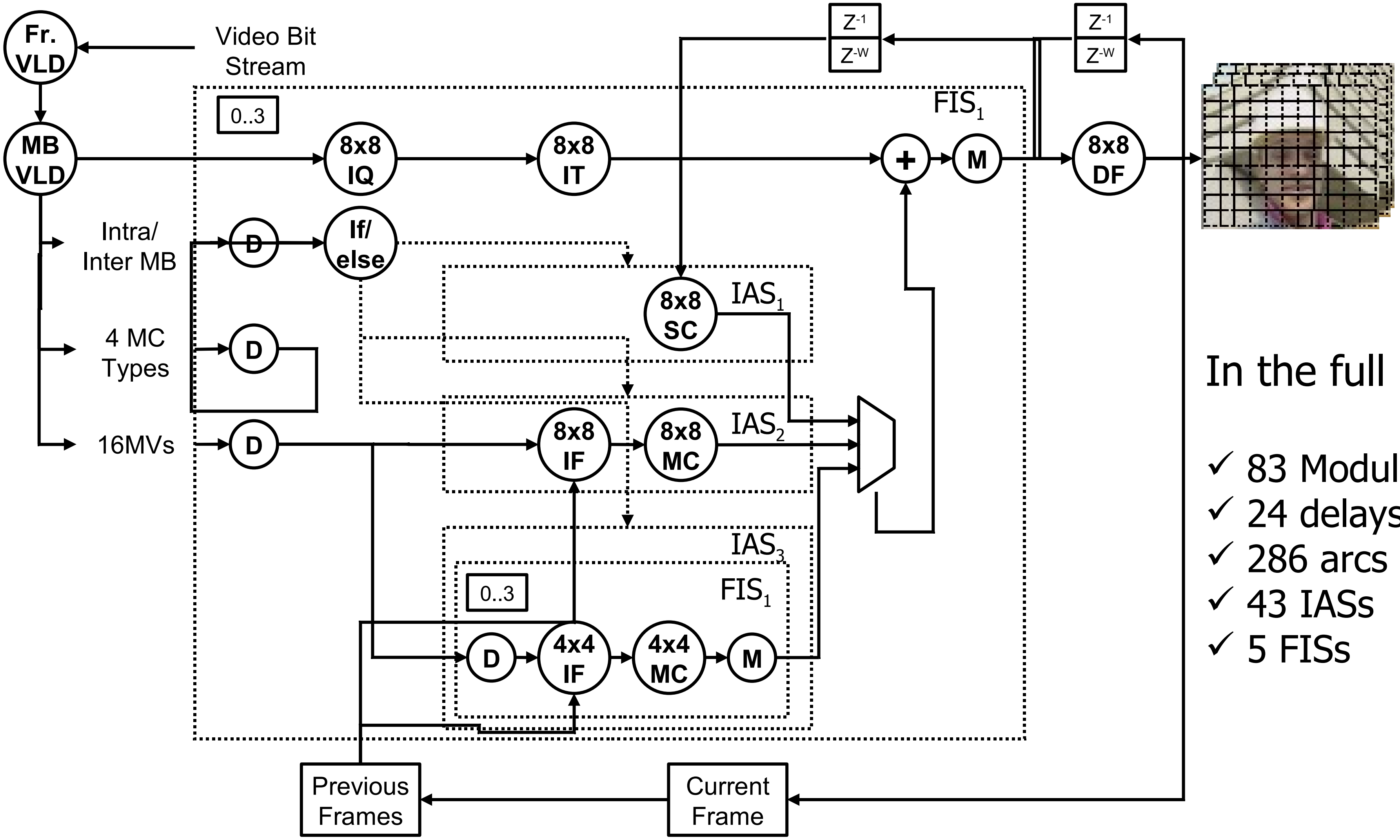
RTL model



ACSM model



A simplified ACSM of H.264 decoder



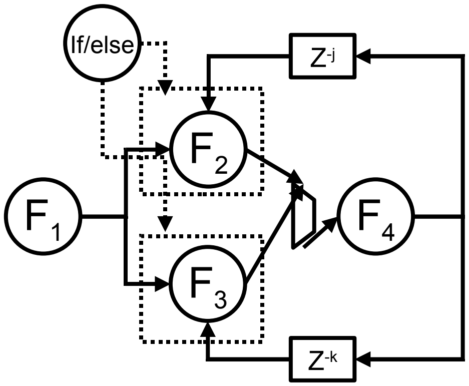
In the full model

- ✓ 83 Modules
- ✓ 24 delays
- ✓ 286 arcs
- ✓ 43 IASs
- ✓ 5 FISs

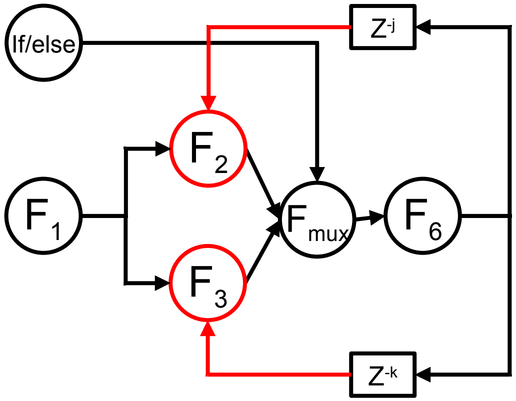
Contents

- Introduction
- Proposed modeling style
- **Comparison with existing modeling styles**
- Experiment
- Conclusion and Perspective

Synchronous Dataflow (SDF)



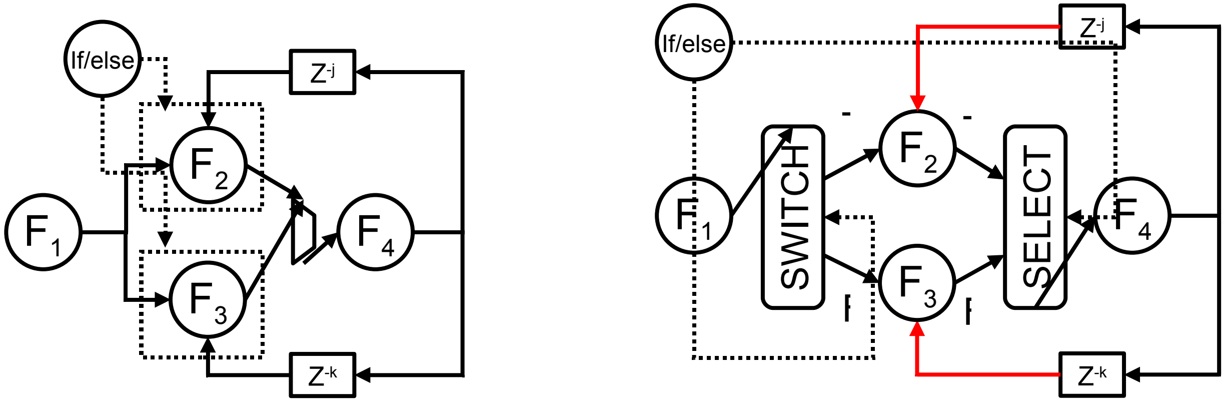
(a) ACSM



(b) SDF

- □ **SDF:** Concurrent actors communicate with one-way FIFO channels
- □ **Rule on execution**: If sufficient non-zero tokens on **all** input ports, fired
- □ **Rule on absent token**: No concept of absent token.
- □ **Difficulty:** No explicit conditionals **(!req. 4)**
  - ✓ Redundant computation : One of F_2 and F_3
  - ✓ Redundant commutation : One of ($Z^{-j} \Rightarrow F_2$) and ($Z^{-k} \Rightarrow F_3$)

Boolean Dataflow (BDF)



(a) ACSM

(b) BDF

- **BDF:** Concurrent actors communicate with one-way FIFO channels
- **Rule on execution**: If sufficient non-zero tokens on **all** input ports, fired
  - ✓ Except SWITCH and SELECT blocks
- **Rule on absent token**: Absent token without global clock.
- **Difficulty:** Buffer overflow problem. **(!req. 4)**
  - ✓ Accumulate tokens between $(Z^{-j} \Rightarrow F_2)$ and $(Z^{-k} \Rightarrow F_3)$
  - ✓ To solve this problem, a lot of additional switches: 138 switches/83 modules in case of H.264 decoder

Other modeling styles

- ❑ Khan Process Network (KPN)
  - ✓ **Difficulty1:** Task-level coarse granularity **(!req 1)**
  - ✓ **Difficulty2:** Comp./comm./control mixed **(!req 2, !req4)**

- ❑ Synchronous Dataflow with embedded control (SDF+eCtrl)
  - ✓ **Difficulty1:** Coarser granularity. **(!req. 1)**
  - ✓ **Difficulty2:** Still Redundant communication **(!req. 4)**

- ❑ Synchronous model (SM)
  - ✓ **Rule on execution**: If one present token on **one or more** input ports, fired
  - ✓ **Rule on absent token**: Absent tokens with virtual clock.
  - ✓ **Difficulty:** distributed implementation **(!req 1)**

Summary of Modeling styles

Models Requirements	KPN	SDF	SDF +eCtrl	BDF	SM	RTL	ACSM
Explicit fine-grain parallelism	Δ	√	Δ	√	×	√	√
Explicit communication	Δ	√	√	√	√	√	√
Higher level synchronization	√	√	√	√	√	×	√
Explicit data-dependent op.	×	×	Δ	Δ	√	√	√
Model Type	**Data-driven**				**Event-driven**	**Clock-driven**	
Clock	No global clock				Virtual clks	Physic clk	Abstract clk
Analysis	No absent token (or without clock) **Explicit conditionals**				Unrestricted absent token **Distributed system**	Restricted absent token with global clock Execution ≈ Data-driven Absent token ≈Event-driven	

Contents

□ Introduction

□ Proposed modeling style

□ Comparison with existing modeling styles

□ **Experiment**

□ Conclusion and Perspective

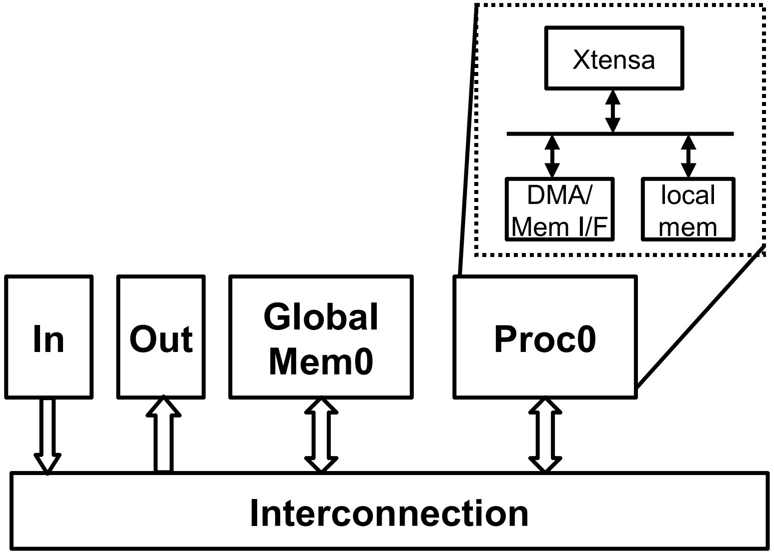
Experiment environment

❑ Target application

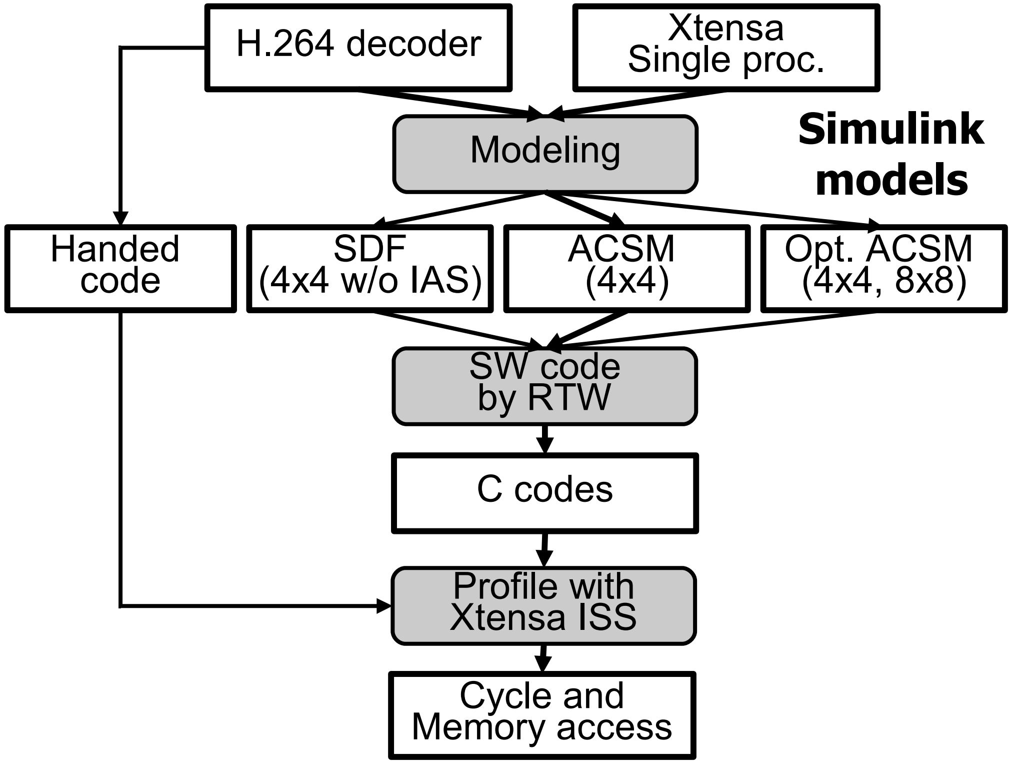
- ✓ H.264 baseline decoder
- ✓ Foreman QCIF format with QP=28 and IntraPeriod=5
- ✓ All search modes are supported.

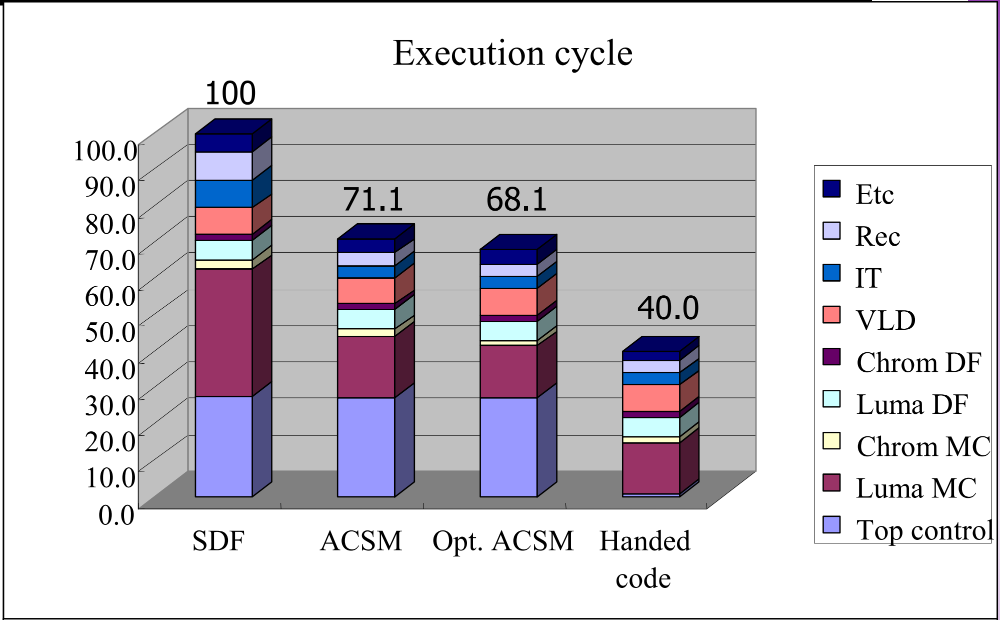
❑ Target architecture

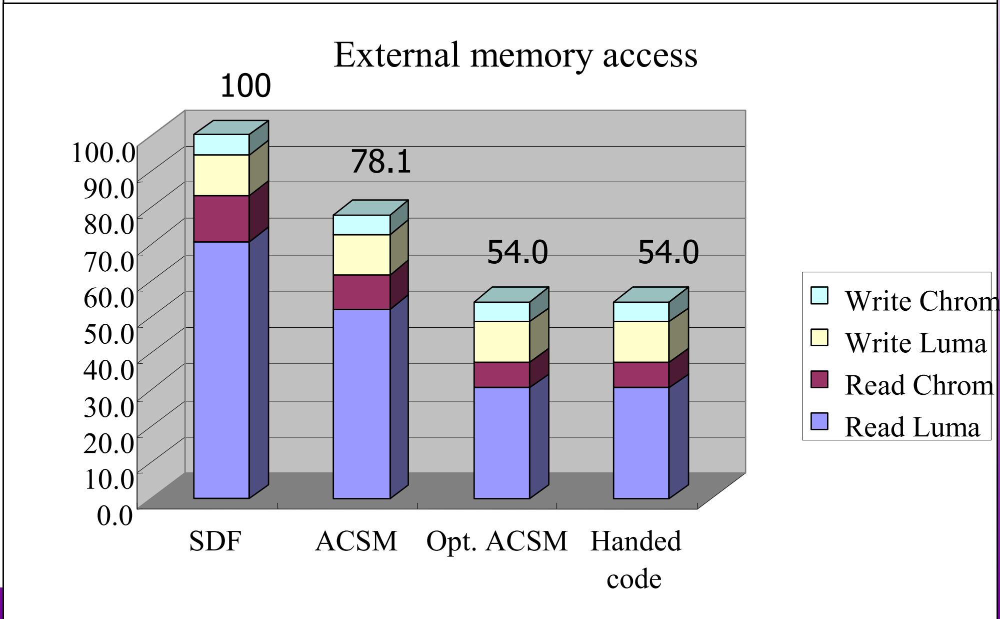
- ✓ Tensilica Xtensa single processor
- ✓ DMA + memory I/F
- ✓ Mapping
  - ◆ Frames are stored in global memory.
  - ◆ Image fetch process is executed on DMA.
  - ◆ All other processes are executed on Xtensa.



Experiment







02/01/06

Experiment result: Comparison

- ❑ Compared to SDF
  - ✓ Reduce 32% execution time
    - Due to Redundant computation of SDF
  - ✓ Reduce 46% external memory access
    - Due to redundant communication of SDF
- ❑ Compared to SDF+eCtrl
  - ✓ Similar execution time, but SDF+eCtrl limits design space.
  - ✓ Reduce 46% external memory access
    - Due to redundant communication of SDF+eCtrl
- ❑ Compared to BDF
  - ✓ Buffer overflow,
  - ✓ or a lot of switches required (138 switches/83 modules)

Contents

□ Introduction

□ Contribution

□ Basics of target application and architecture

□ Proposed modeling style

□ Comparison with existing modeling styles

□ Experiment

□ **Conclusion and Perspective**

Conclusion and perspective

❑ Conclusion

- ✓ ACSM suitable for video codec application
  - An high level RTL model
  - Satisfies the requirements of functional model

- ✓ Efficient SW code generation from Simulink with RTW
  - Reduce 32% execution time than SDF
  - Reduce 46% external memory access than SDF

❑ Perspective

- ✓ SW generation tool for distributed implementation.

Thank you!!!

Question?