Hardware Accelerators

Shao-Yi Chien

Ref: W. Wolf, “Chap 7: Hardware Accelerator,” *Computers as Components*, Academic Process, 2001.

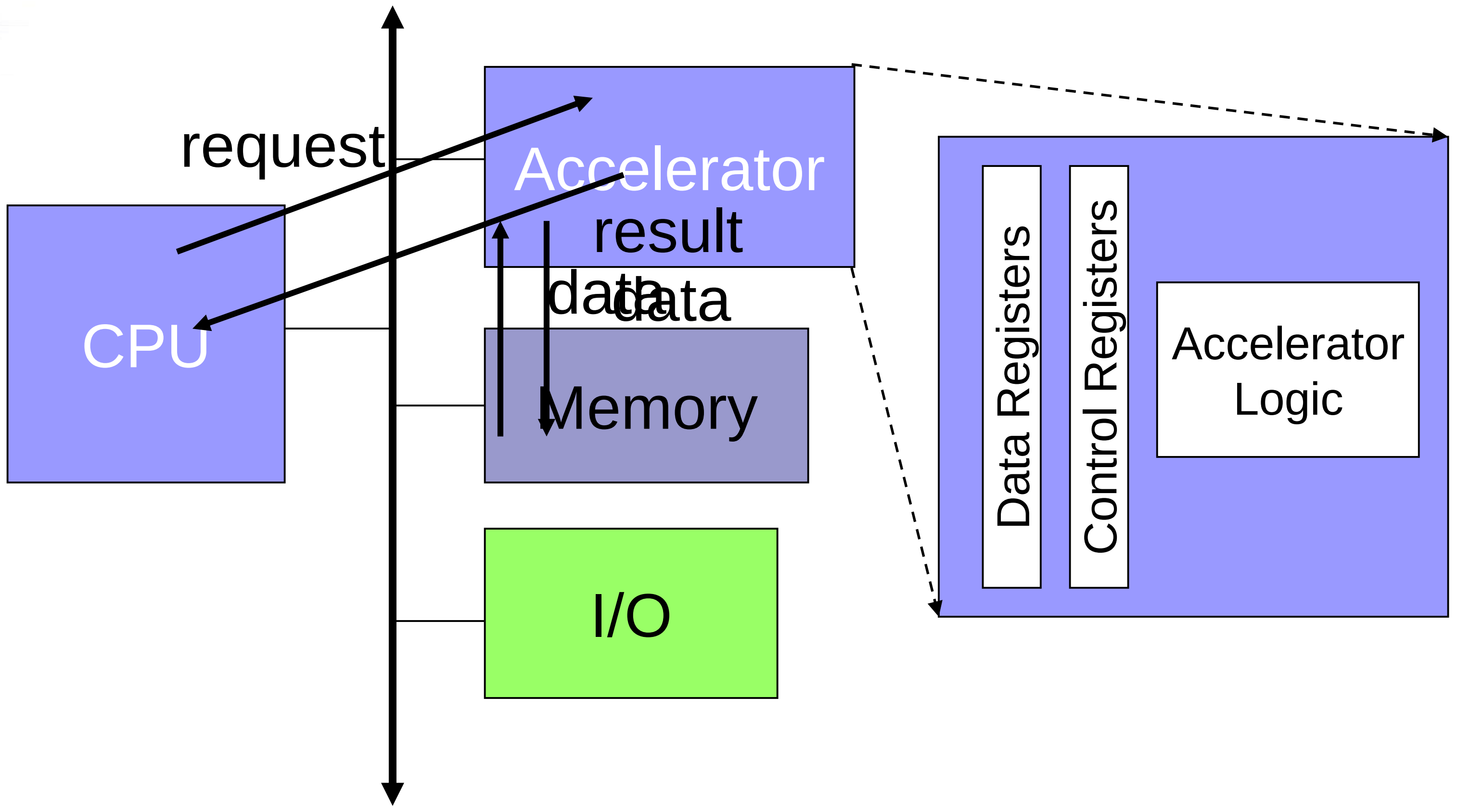
Outline

- CPU and accelerators
- Why accelerators?
- Accelerated system design
- Important concepts
- Design examples

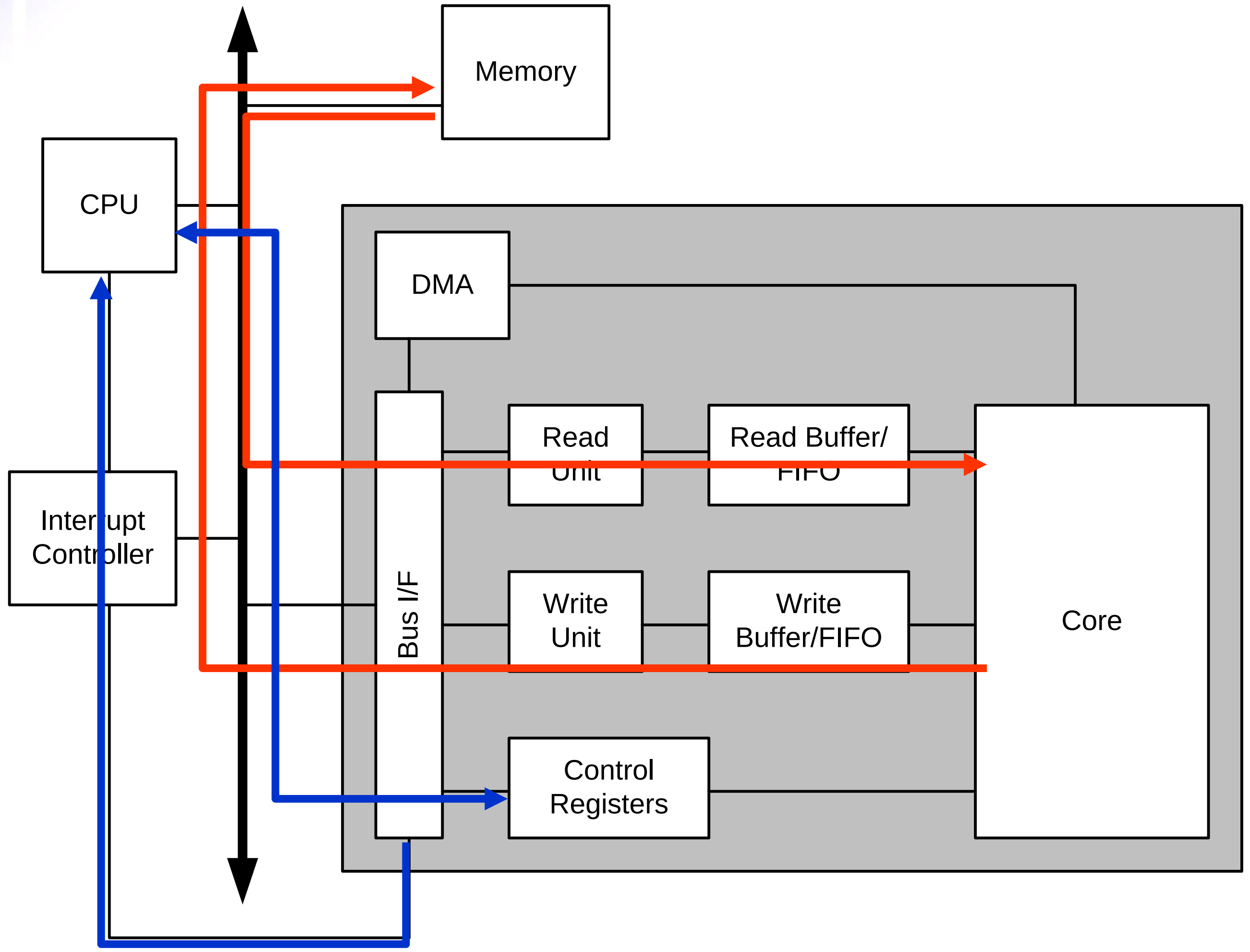
Accelerated Systems

- Use additional computational unit dedicated to some functions?
  - Hardwired logic
  - Extra CPU
- Hardware/software co-design: joint design of hardware and software architectures.

Typical Accelerated System Architecture



Accelerator Architecture Framework



Accelerator vs. Co-Processor

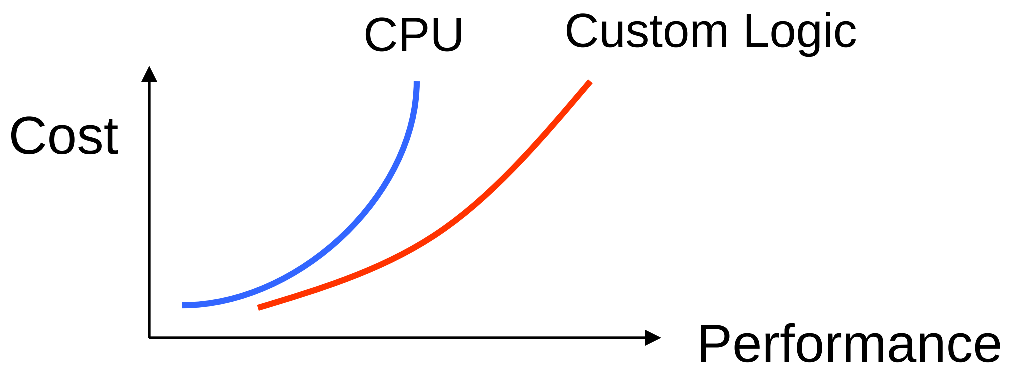
- A **co-processor** executes instructions
  - Instructions are dispatched by the CPU
  - **Tightly coupled connection**
- An **accelerator** appears as a device on the bus
  - The accelerator is controlled by registers
  - **Loosely coupled connection**

System Design Tasks

- Design a heterogeneous multiprocessor architecture.
    - Processing element (PE): CPU, accelerator, etc.
- Program the system

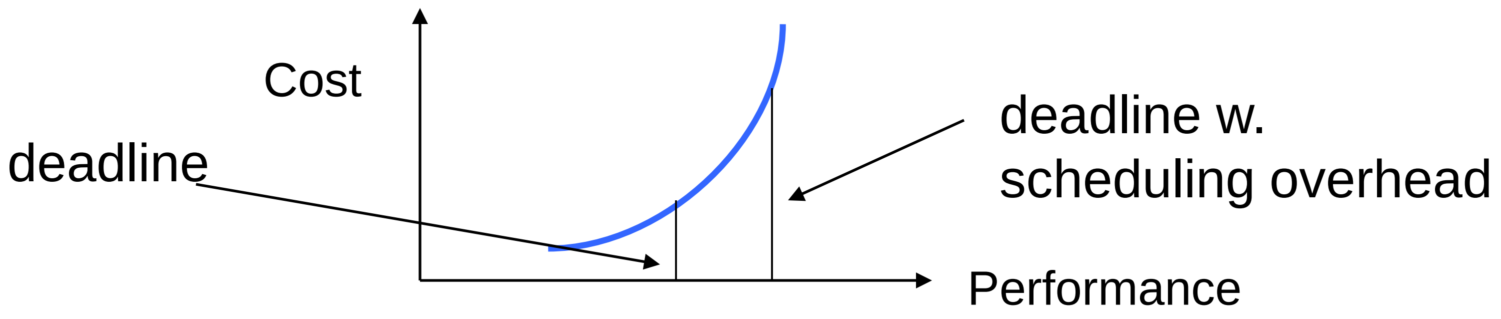
Why Accelerators?

- Better cost/performance.
  - Custom logic may be able to perform operation faster than a CPU of equivalent cost
  - CPU cost is a non-linear function of performance



Why Accelerators?

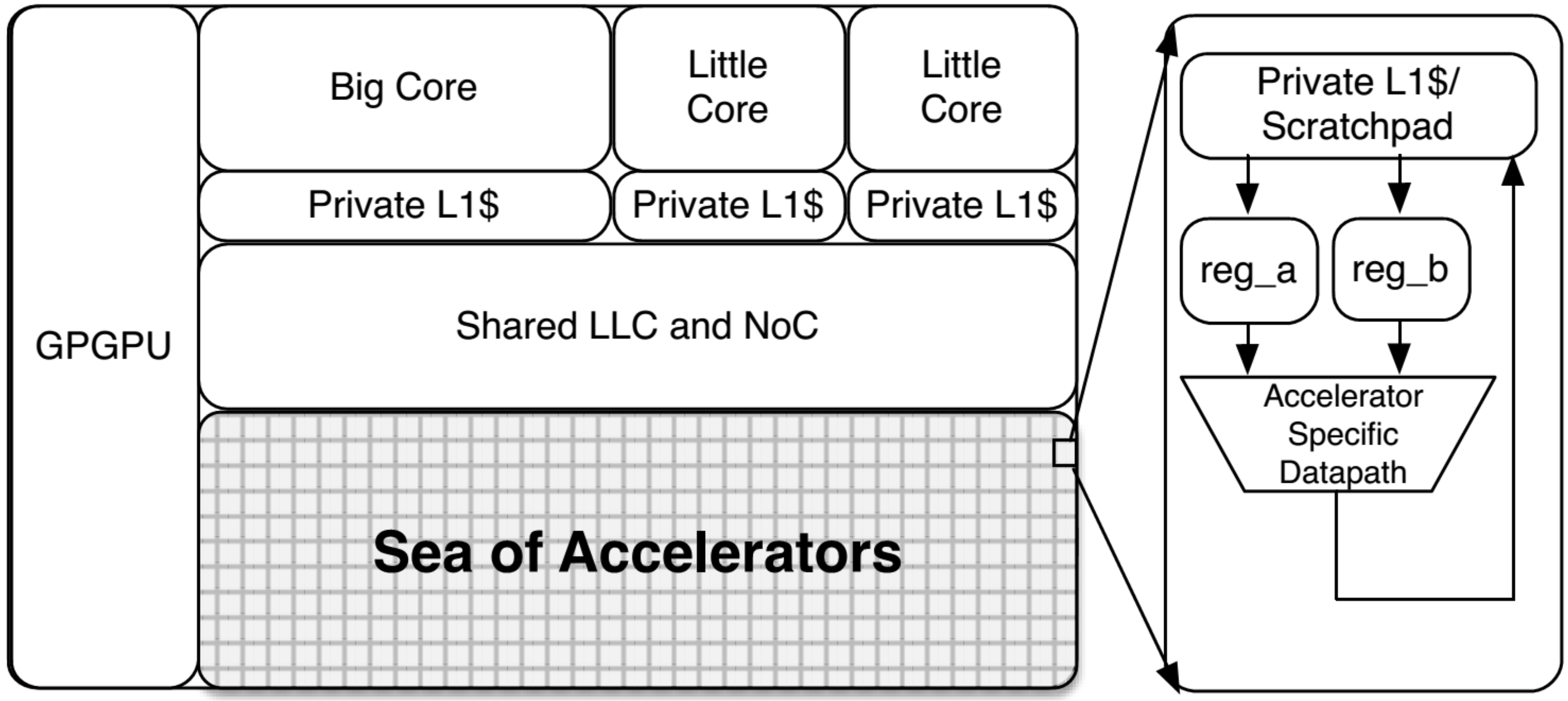
- Better real-time performance
  - Put time-critical functions on less-loaded processing elements
  - Remember RMS utilization---extra CPU cycles must be reserved to meet deadlines.



Why Accelerators?

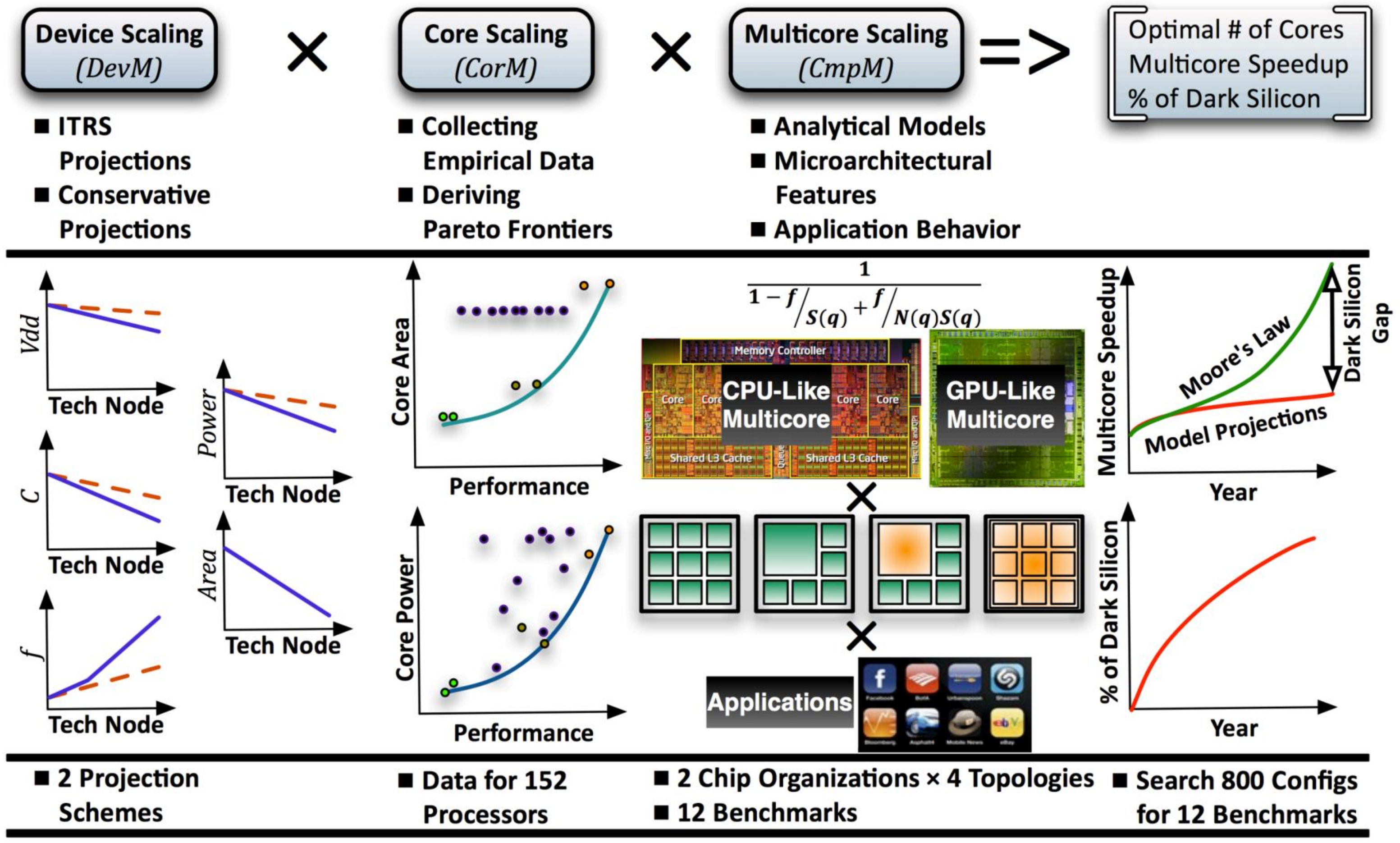
- Good for processing I/O in real-time
- **May consume less energy**
- May be better at streaming data
- May not be able to do all the work on even the largest single CPU

Why Accelerators?



Ref: Yakun Sophia Shao, Brandon Reagen, Gu-Yeon Wei, David Brooks, "Aladdin: A Pre-RTL, Power-Performance Accelerator Simulator Enabling Large Design Space Exploration of Customized Architectures," in Proc. International Symposium on Computer Architecture (ISCA), 2014.

Dark Silicon



Ref: H. Esmaeilzadeh, E. Blem, R. St. Amant, K. Sankaralingam, and D. Burger, “Dark silicon and the end of multicore scaling,” Micro, IEEE, 2012.

Types of Applications Suited to Hardwired Accelerators

- Functions requiring operations that do **not map well onto a CPU's data operation**
  - Bit level operations
  - Operations requiring too many registers
  - To control the precision of the arithmetic
- Highly responsive **input and output operations** may be best performed by an accelerator with an attached I/O unit
- **Streaming data**, such as wireless and multimedia

Accelerated System Design

- First, determine that the system really needs to be accelerated
  - How much faster is the accelerator on the **core function**?
  - How much **data transfer overhead**?
- Design the accelerator itself
- Design CPU interface to accelerator

Performance Analysis

- Critical parameter is speedup: how much faster is the system with the accelerator?
- Must take into account:
    - Accelerator execution time
    - Data transfer time
    - Synchronization with the master CPU

Accelerator Execution Time

- Total accelerator execution time:
  - $t_{accel} = t_{in} + t_x + t_{out}$

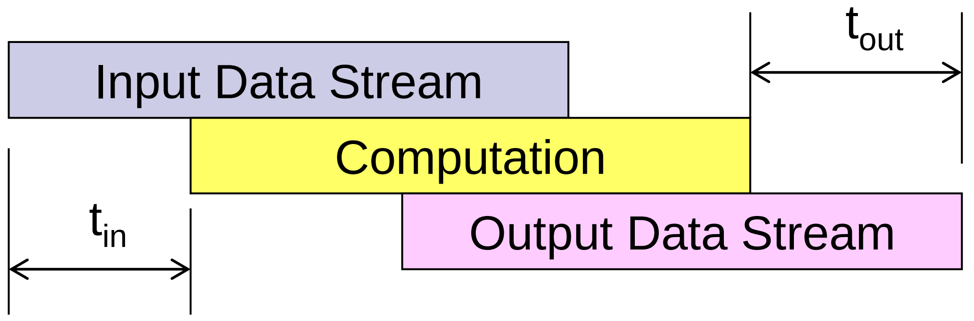
Data input

Accelerated computation

Data output

Accelerator Execution Time

- A more sophisticated accelerator could try to overlap input and output with computation
- t_{in}
  - Non-overlapped read time
  - Determined by the amount of data read in before starting computation
- t_{out}
  - Non-overlapped write time
  - The length of time between the last computation and the last data output



Data Input/Output Times

- Bus transactions include
  - Flushing register/cache values to main memory;
  - Time required for CPU to set up transaction;
  - Overhead of data transfers by bus packets, handshaking, etc.

Accelerator Speedup

- Assume a loop is executed n times
- Compare accelerated system to non-accelerated system:
  - $S = n(t_{CPU} - t_{accel})$
    $= n[t_{CPU} - (t_{in} + t_x + t_{out})]$

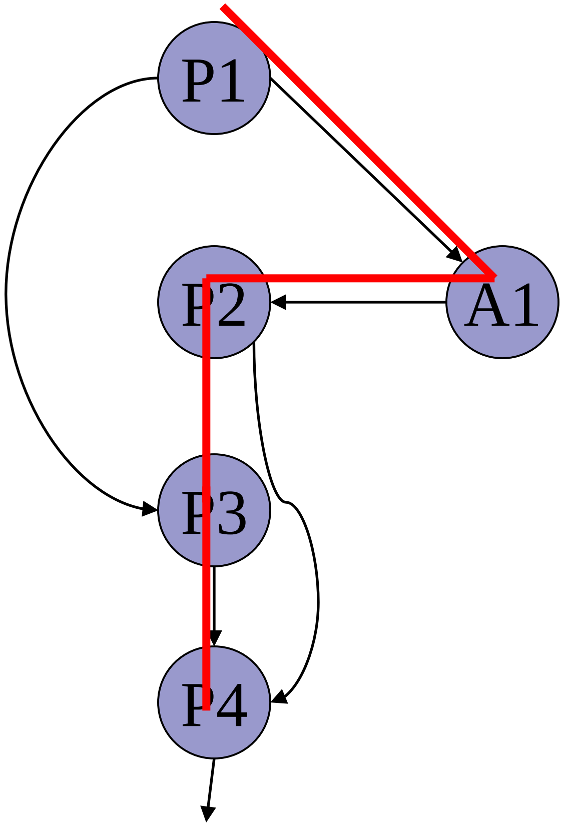
Execution time on CPU

Single- vs. Multi-threaded

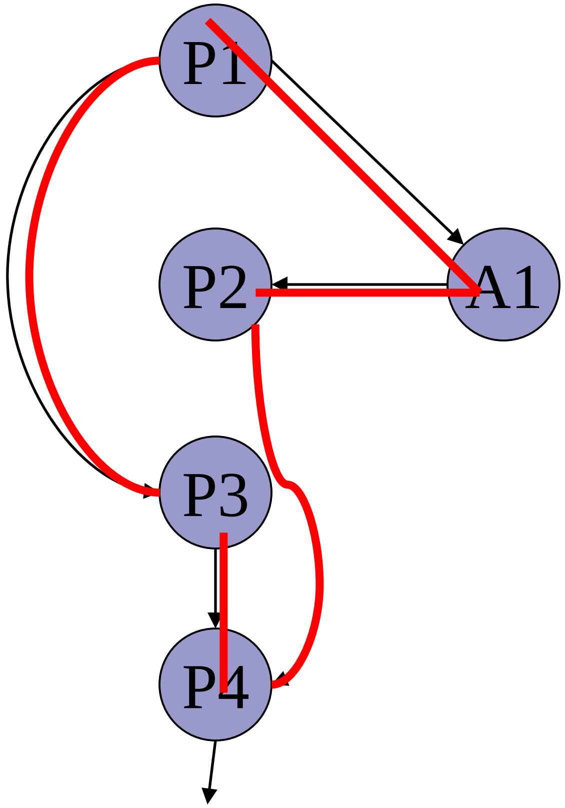
- One critical factor is available parallelism:
  - Single-threaded/blocking: CPU waits for accelerator;
  - Multithreaded/non-blocking: CPU continues to execute along with accelerator.
- To multithread, CPU must have useful works to do
  - But software must also support multithreading

Total Execution Time

- Single-threaded:



- Multi-threaded:



Execution Time Analysis

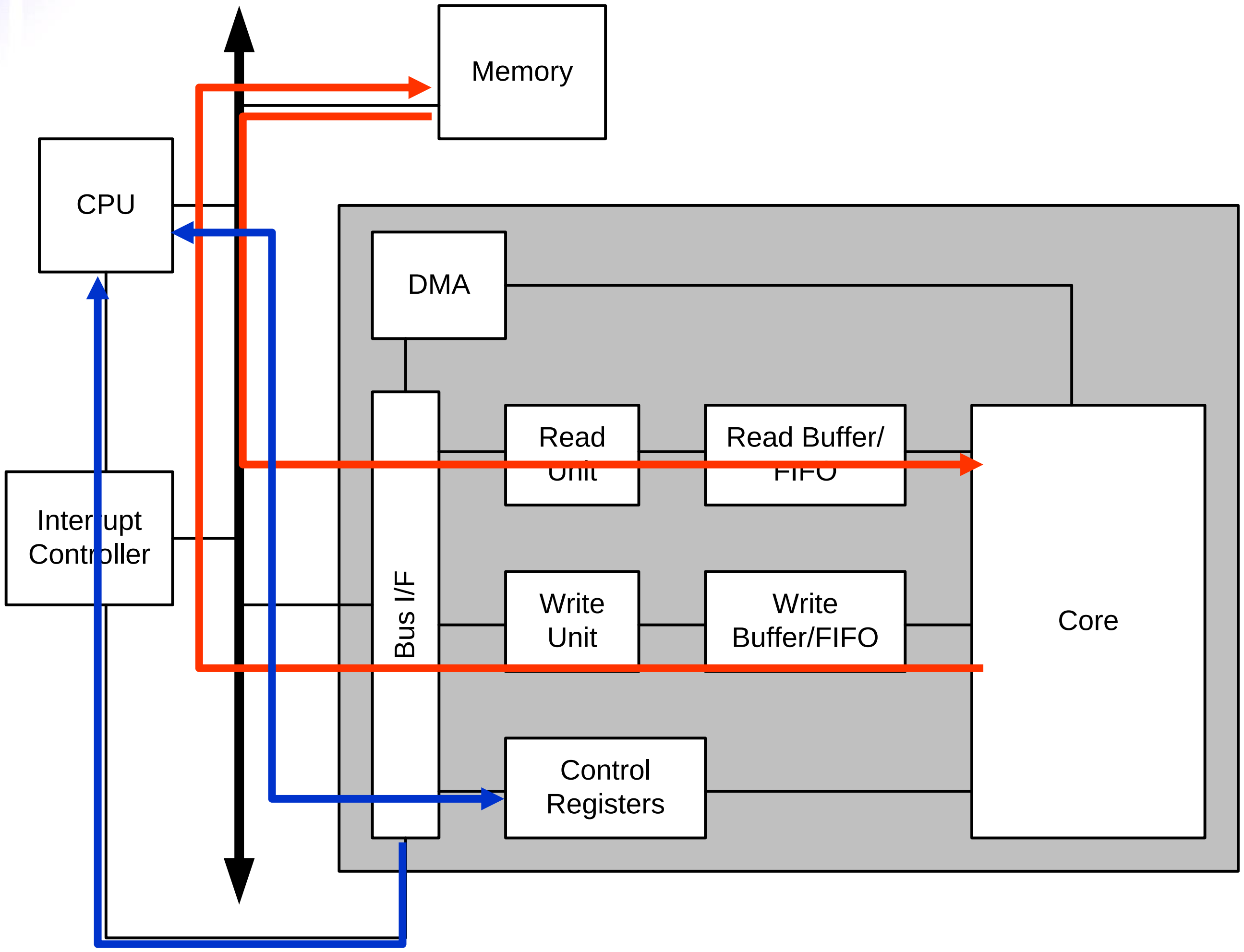
- Single-threaded:
  - Count execution time of all component processes

- Multi-threaded:
  - Find longest path through execution

Sources of Parallelism

- Overlap I/O and accelerator computation
  - Perform operations in batches, read in second batch of data while computing on first batch
- Find other works to do on the CPU
  - May reschedule operations to move work after accelerator initiation

Accelerator Architecture Framework



Accelerator/CPU Interface

- Accelerator registers provide control registers for CPU
- Data registers can be used for small data objects
- Accelerator may include special-purpose read/write logic
  - Especially valuable for large data transfers

Caching Problems

- Main memory provides the primary data transfer mechanism to the accelerator.
- Programs must ensure that caching does not invalidate main memory data
  - CPU reads location S
  - Accelerator writes location S
  - CPU again reads location S

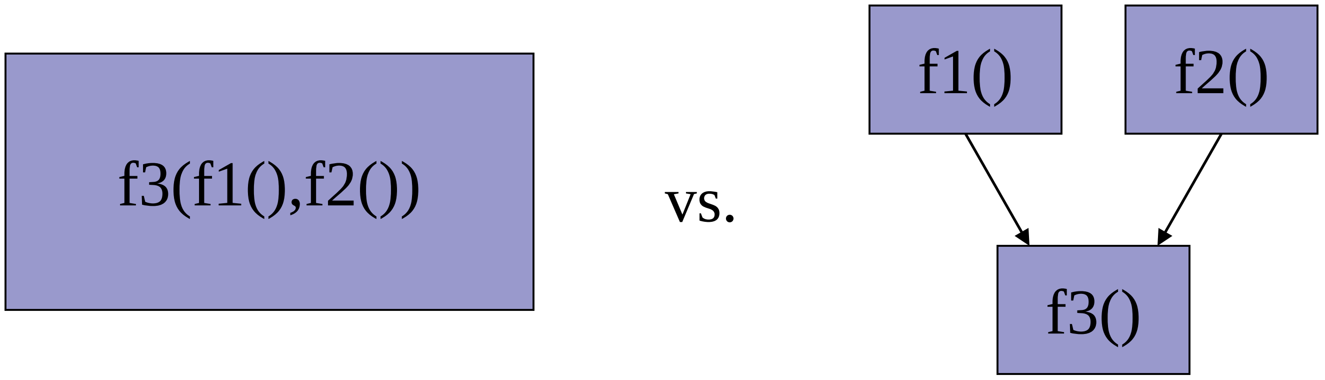
Solutions for the Conflicts

- Exploit cache invalidation instructions
- Remove the location from the cache by reading another location that is mapped to the same cache line

- For memory access conflicts
  - Apply test-and-set scheme

Partitioning/Decomposition

- Divide functional specification into units.
  - Map units onto PEs
  - Units may become processes
- Determine proper level of parallelism



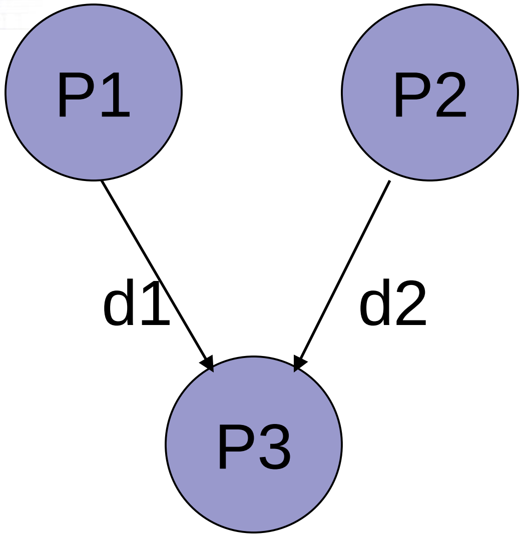
Partitioning/Decomposition

- Partitioning should be driven by performance analysis
- Partitioning should identify possible partitions from which the designer can choose

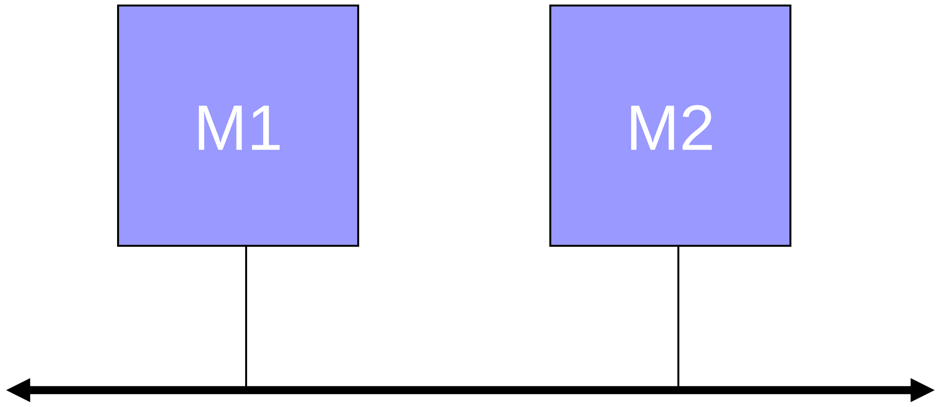
Scheduling and Allocation

- Must:
  - Schedule operations in time
  - Allocate computations to processing elements
- Scheduling and allocation interact, but separating them helps
  - Alternatively allocate, then schedule

Example: Scheduling and Allocation



Task graph



Hardware platform

Example Process Execution Times

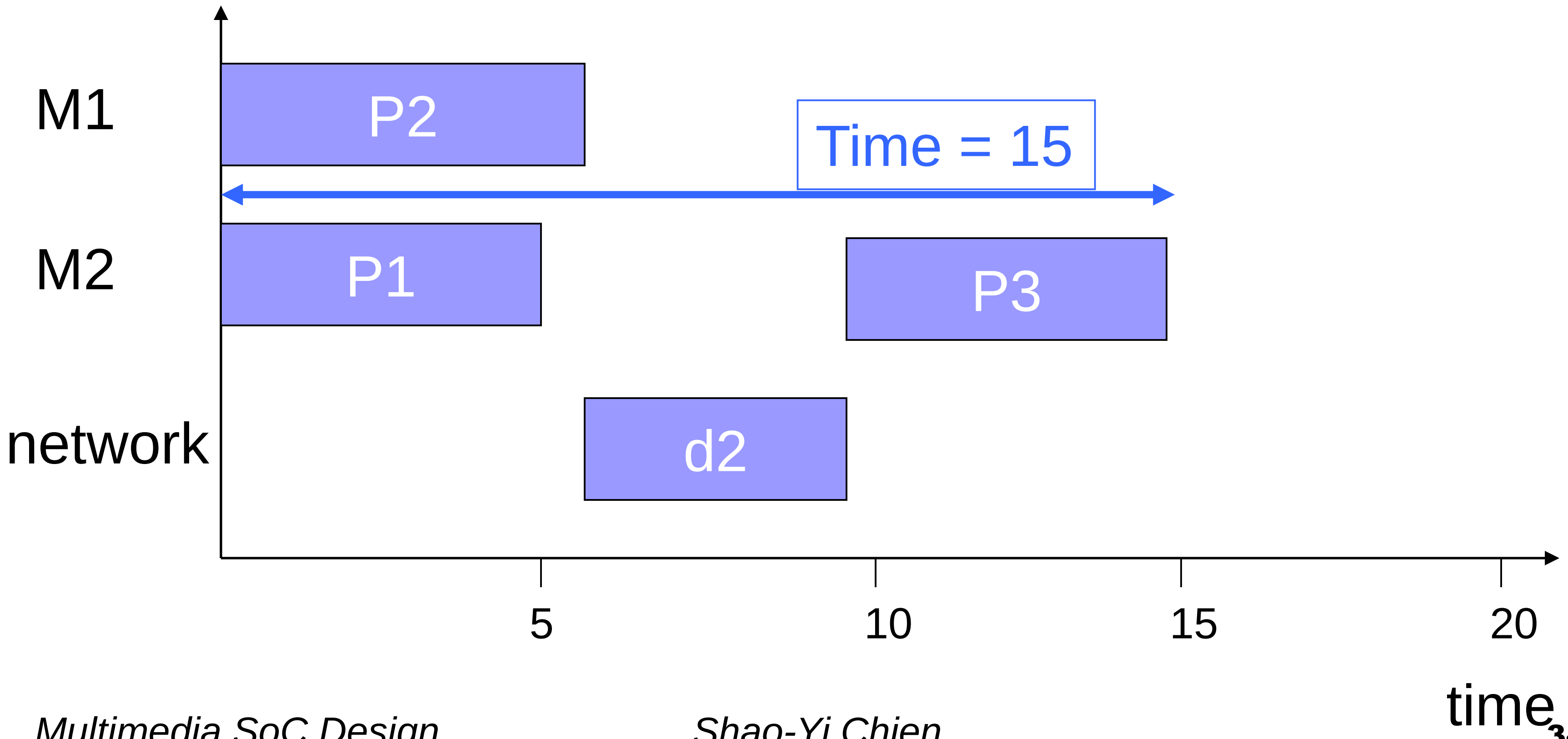
	M1	M2
P1	5	5
P2	5	6
P3	-	5

Example Communication Model

- Assume communication within PE is free
- Cost of communication from P1 to P3 is d1 =2; cost of P2->P3 communication is d2 = 4

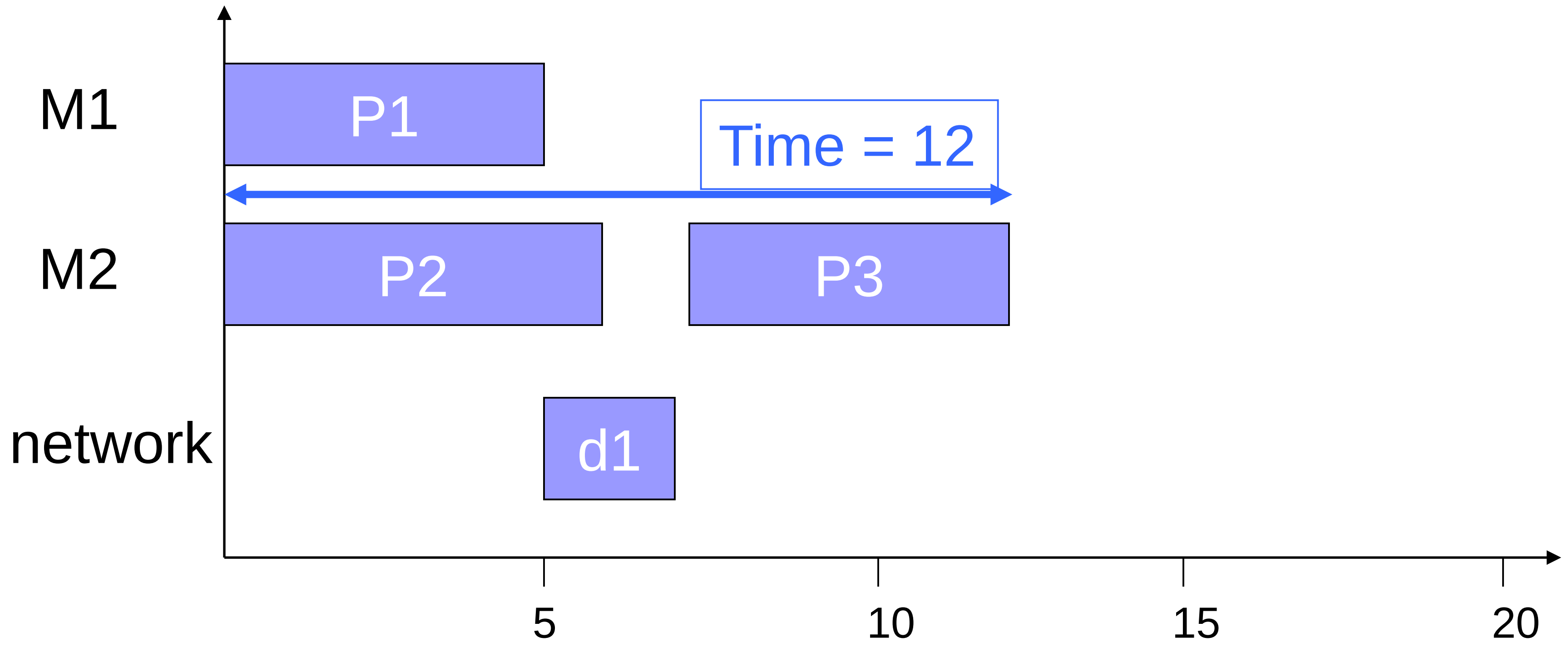
First Design

- Allocate P2 -> M1; P1, P3 -> M2.



Second Design

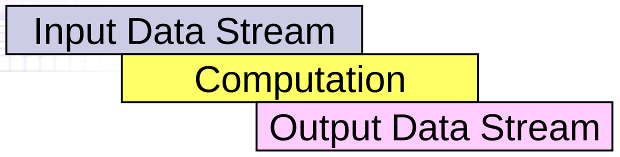
- Allocate P1 -> M1; P2, P3 -> M2:

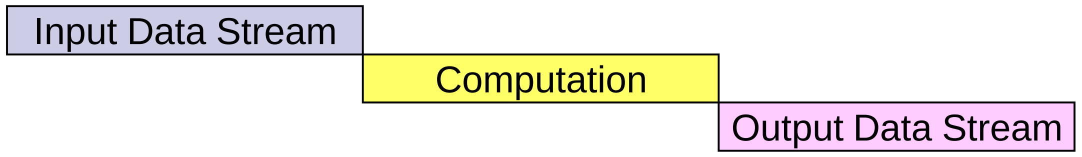


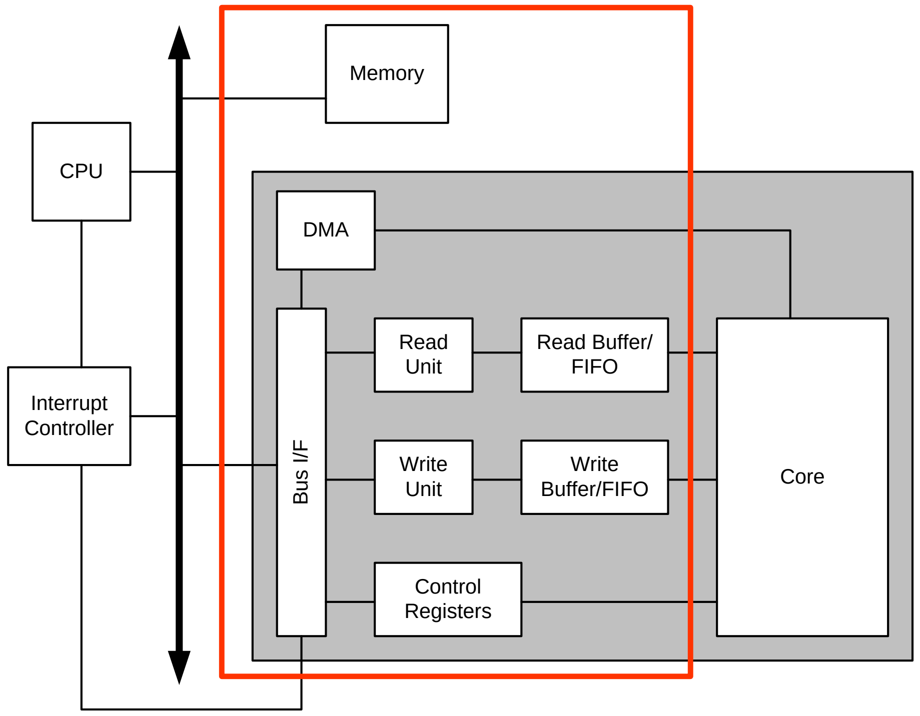
System Integration and Debugging

- Try to debug the CPU/accelerator interface separately from the accelerator core
- Build scaffolding to test the accelerator
- Hardware/software co-simulation can be useful

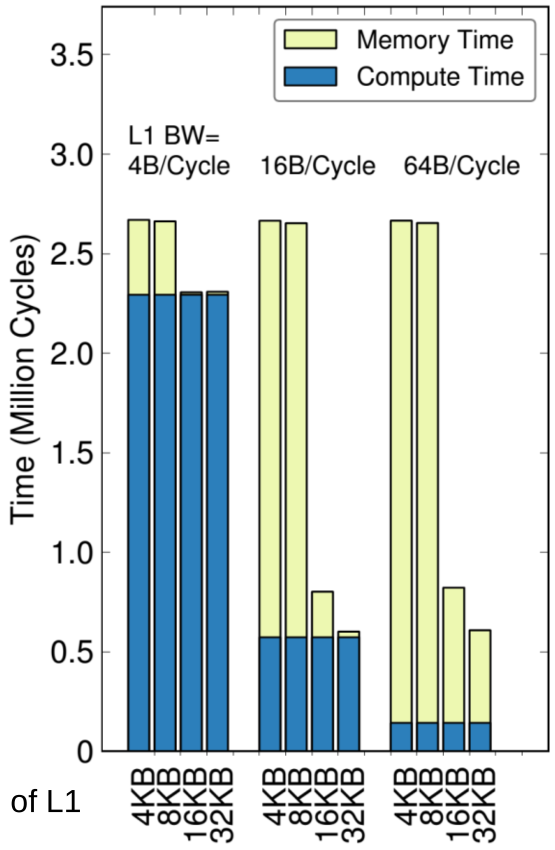
Memory Design is the Key



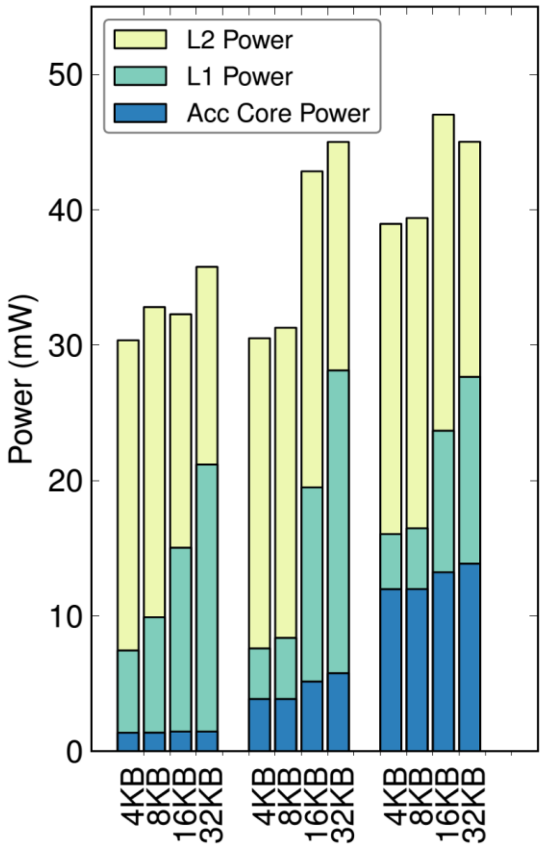




Memory Design is the Key

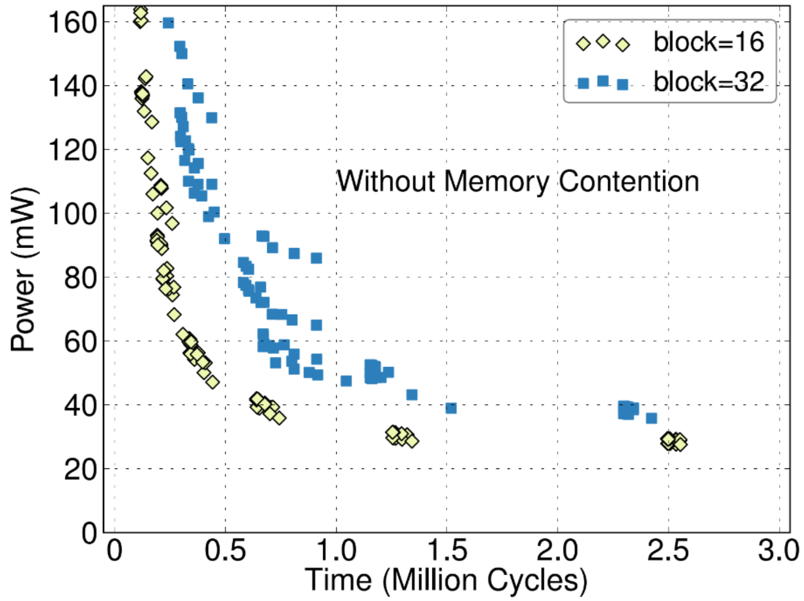


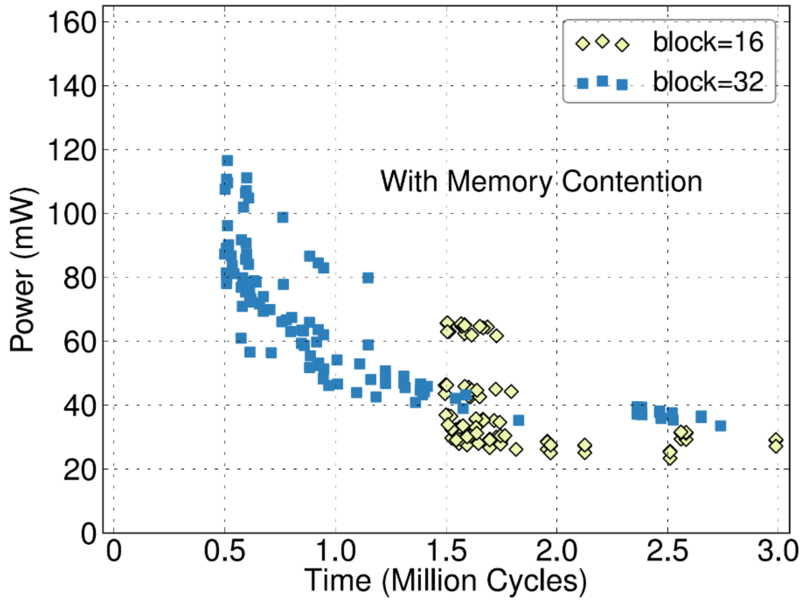
Size of L1



Ref: Yakun Sophia Shao, Brandon Reagen, Gu-Yeon Wei, David Brooks, “Aladdin: A Pre-RTL, Power-Performance Accelerator Simulator Enabling Large Design Space Exploration of Customized Architectures,” in Proc. International Symposium on Computer Architecture (ISCA), 2014.

Memory Design is the Key

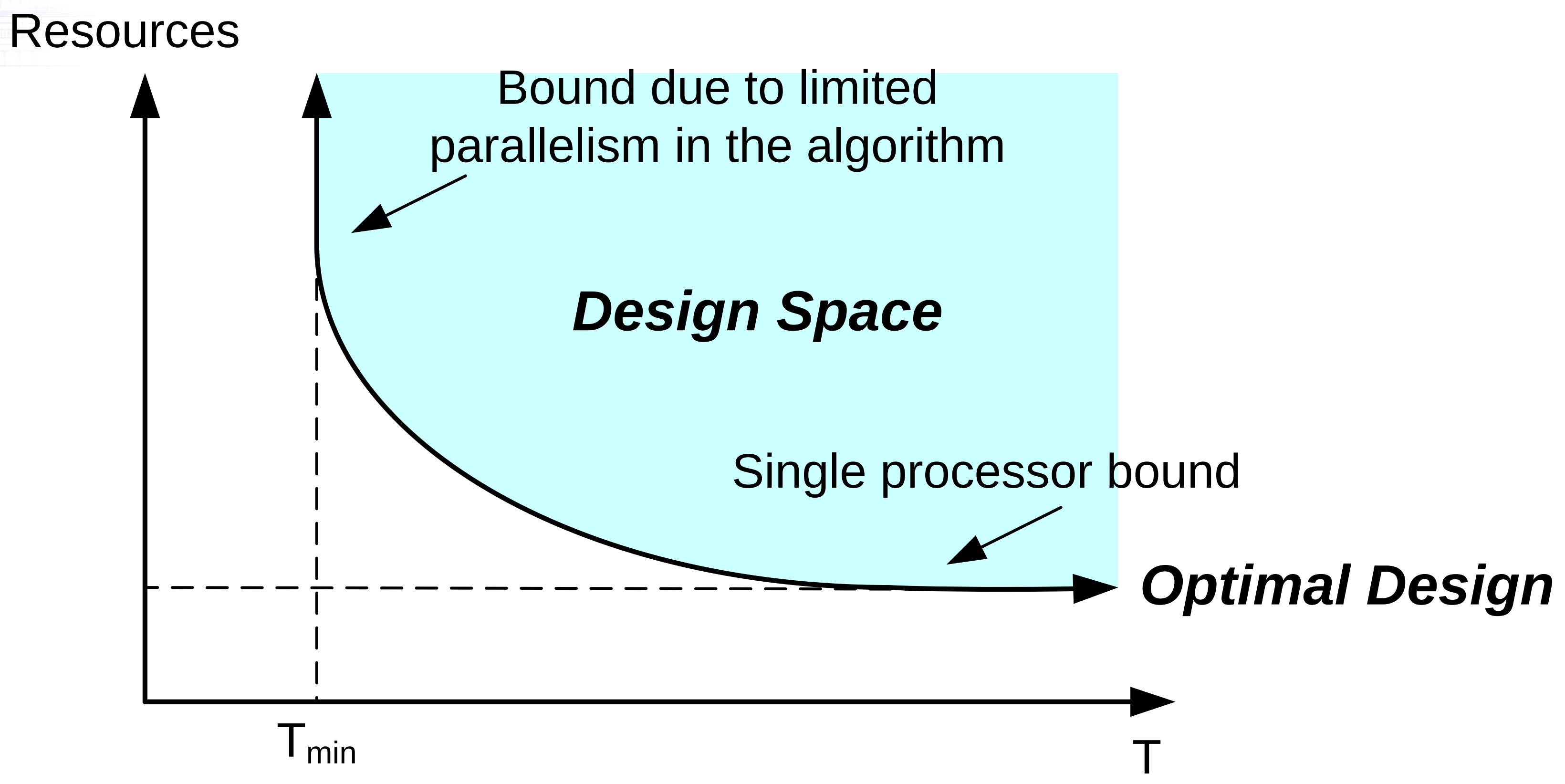




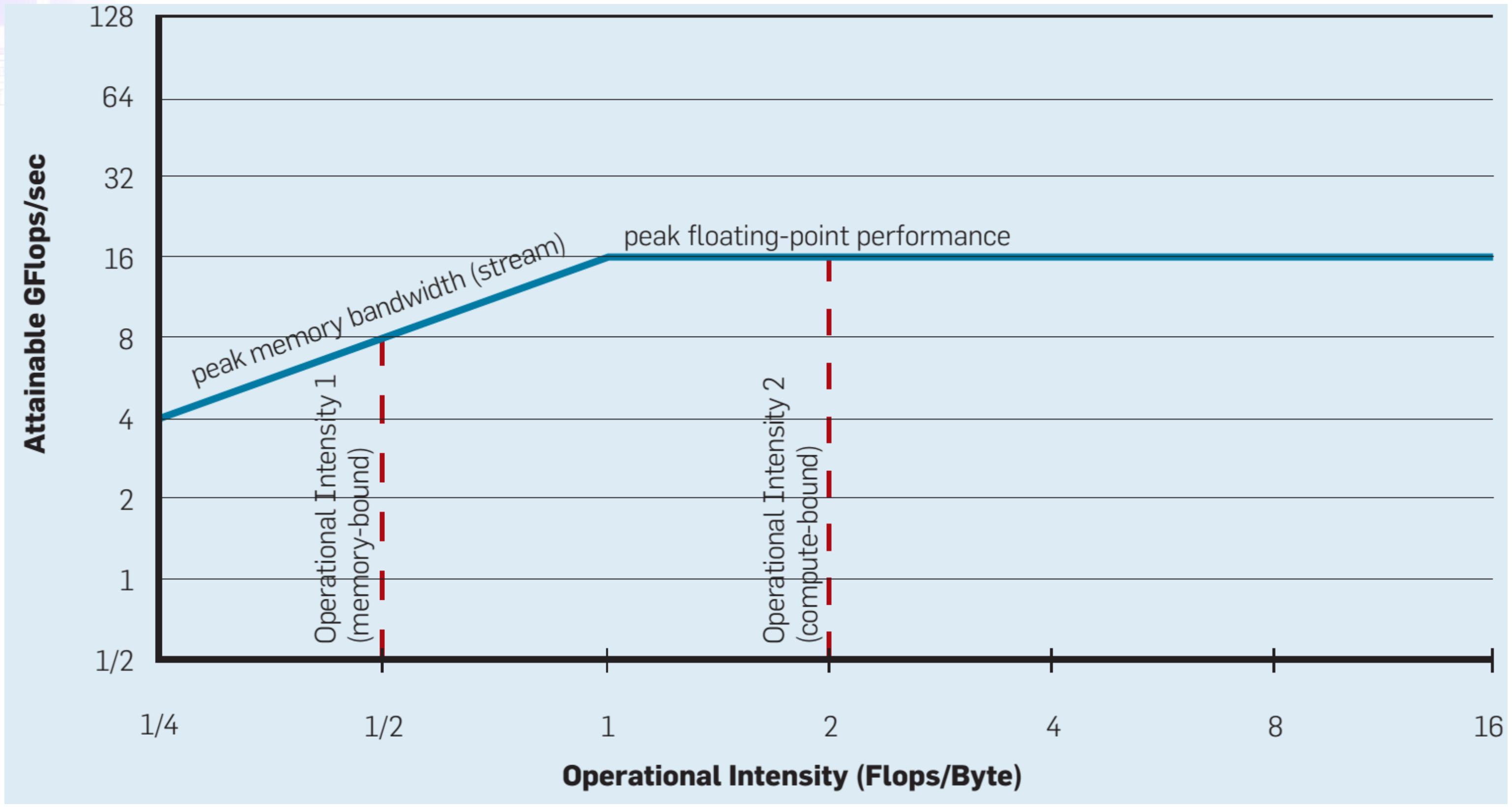
Ref: Yakun Sophia Shao, Brandon Reagen, Gu-Yeon Wei, David Brooks, "Aladdin: A Pre-RTL, Power-Performance Accelerator Simulator Enabling Large Design Space Exploration of Customized Architectures," in Proc. International Symposium on Computer Architecture (ISCA), 2014.

Important Concepts

Design Space: Time and Resources

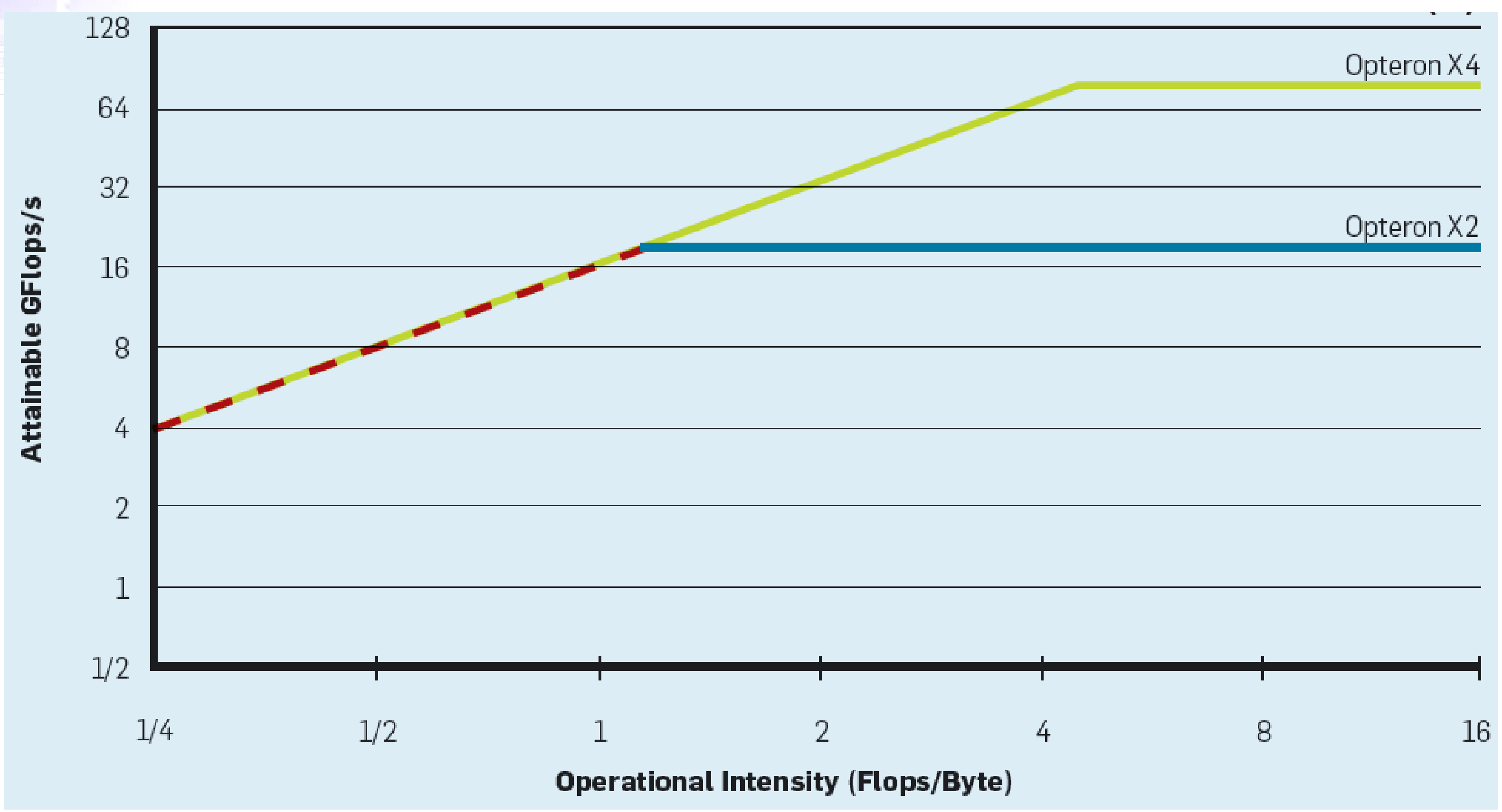


Roofline Model

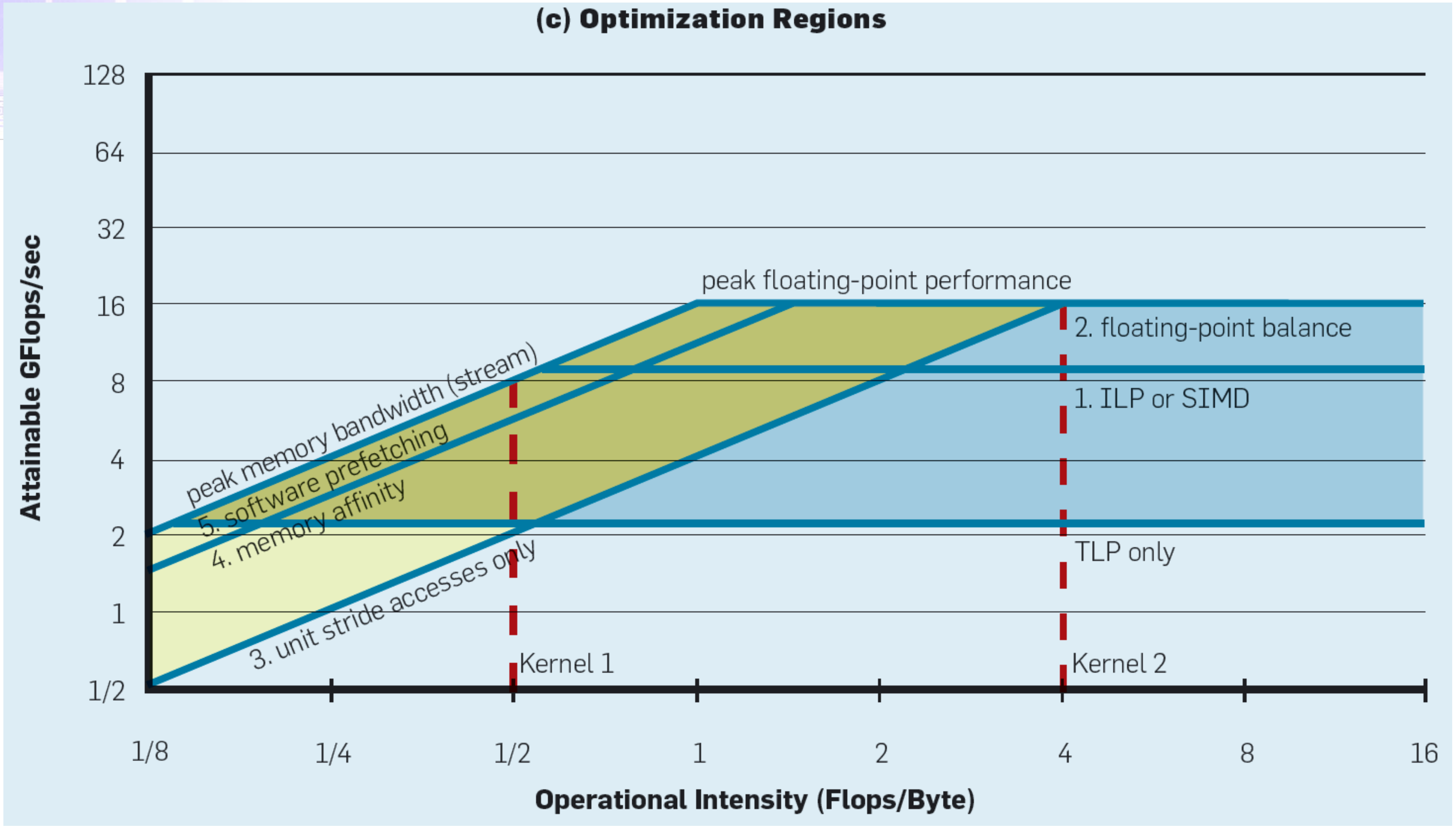


Ref: S. Williams, A. Waterman, and D. Patterson, "Roofline: An Insightful Visual Performance Model for Multicore Architectures," Commun. ACM, April 2009.

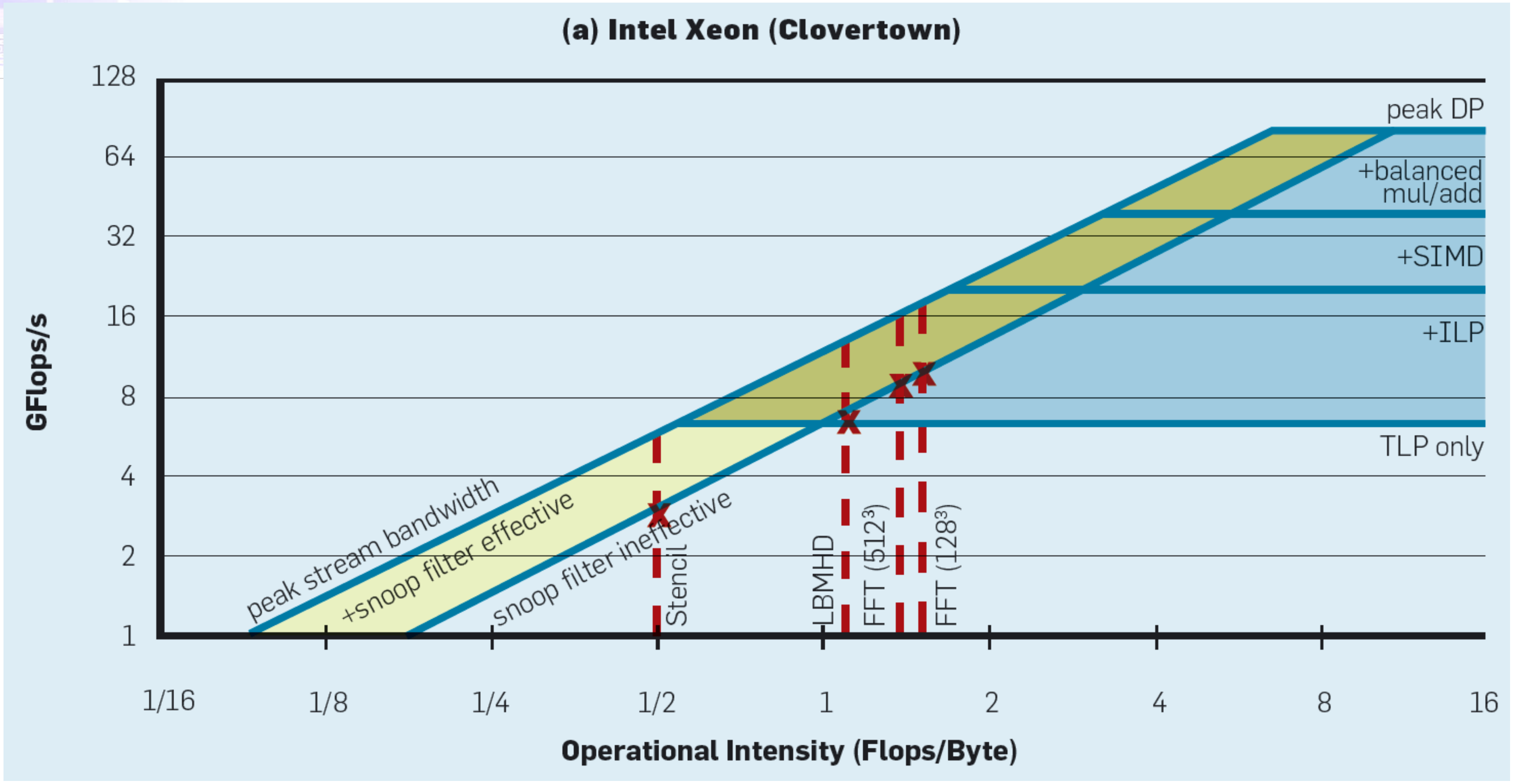
Roofline Model



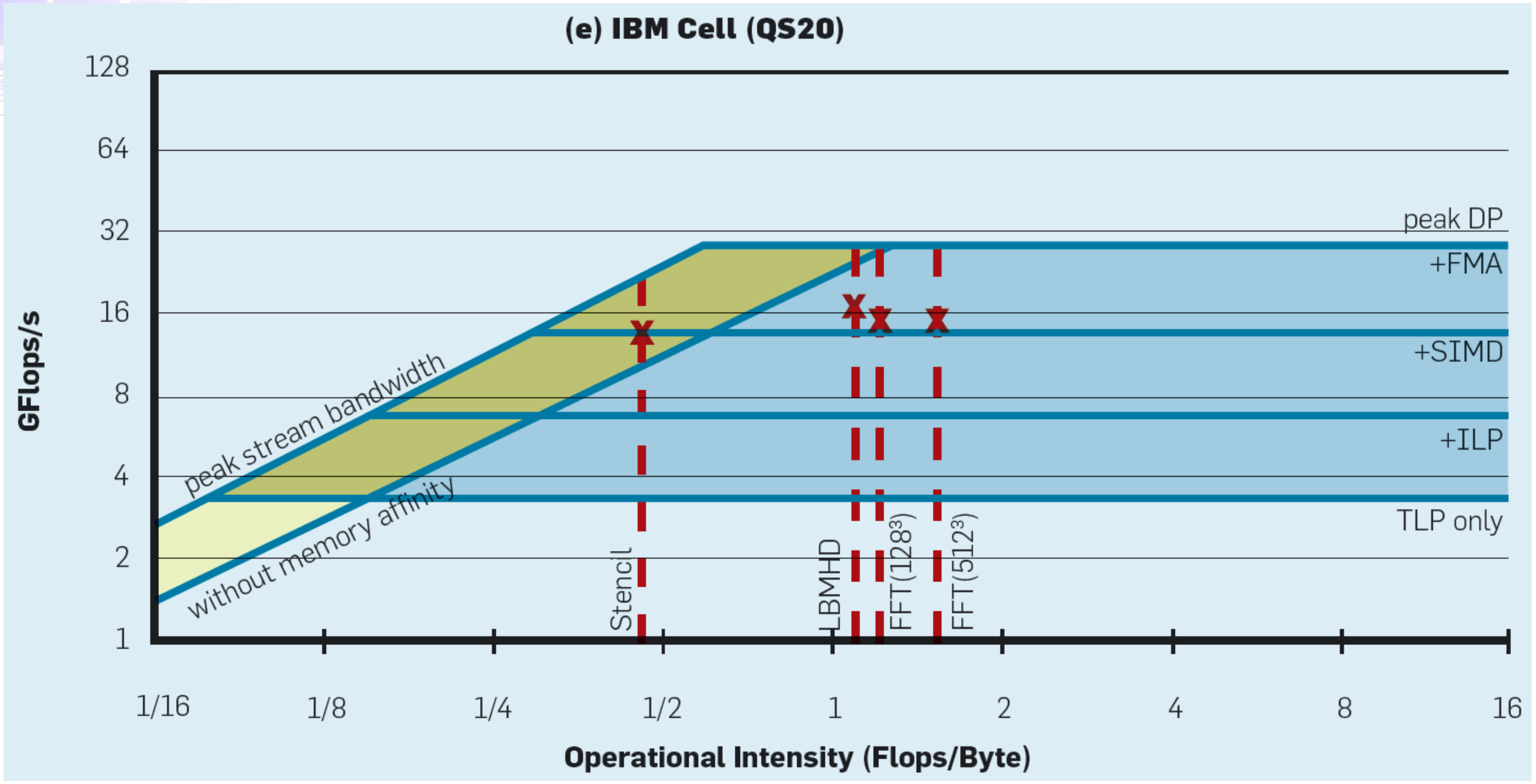
Roofline Model



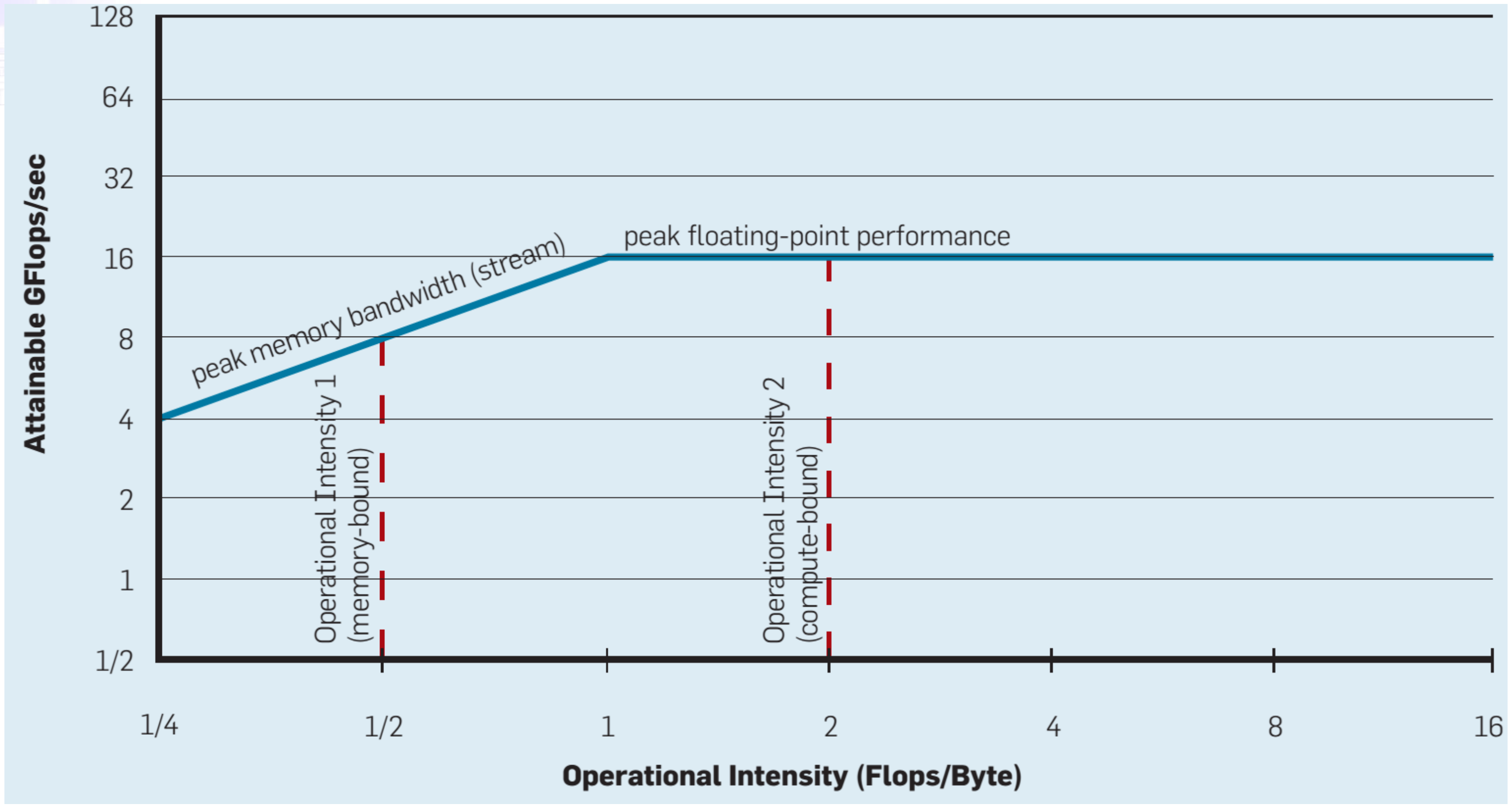
Roofline Model



Roofline Model



Roofline Model for Accelerators?



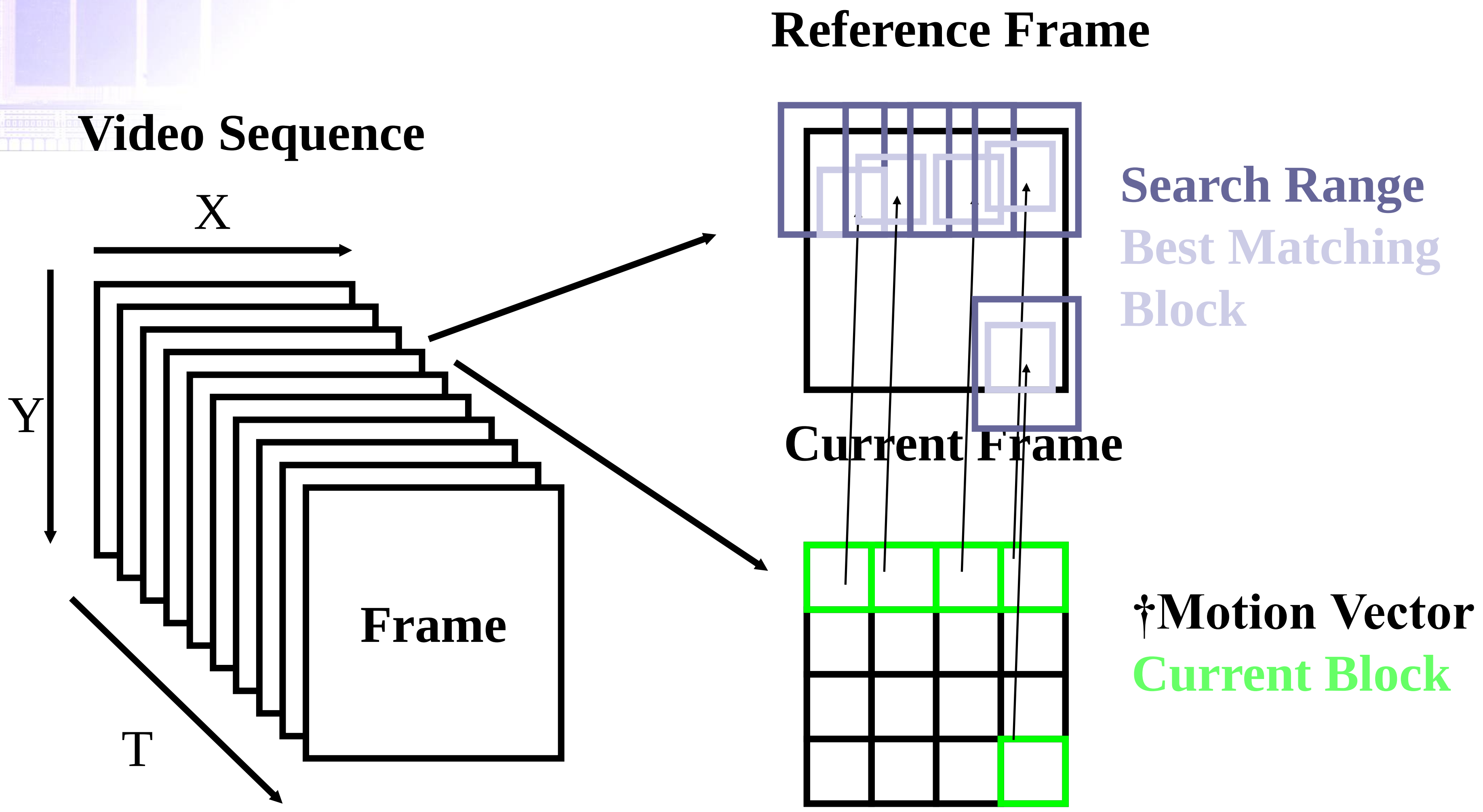
Ref: S. Williams, A. Waterman, and D. Patterson, “Roofline: An Insightful Visual Performance Model for Multicore Architectures,” Commun. ACM, April 2009.

Accelerators

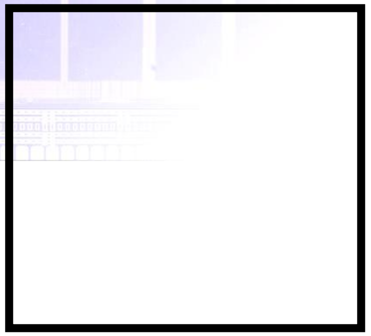
- Example: data reuse scheme for motion estimation

  - Ref: Jen-Chieh Tuan, Tian-Sheuan Chang, and Chein-Wei Jen, “On the data reuse and memory bandwidth analysis for full-search block-matching VLSI architecture,” *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 12, no. 1, pp. 61-72, Jan. 2002.

Block Matching Algorithm (BMA)



Full-Search Block Matching Algorithm

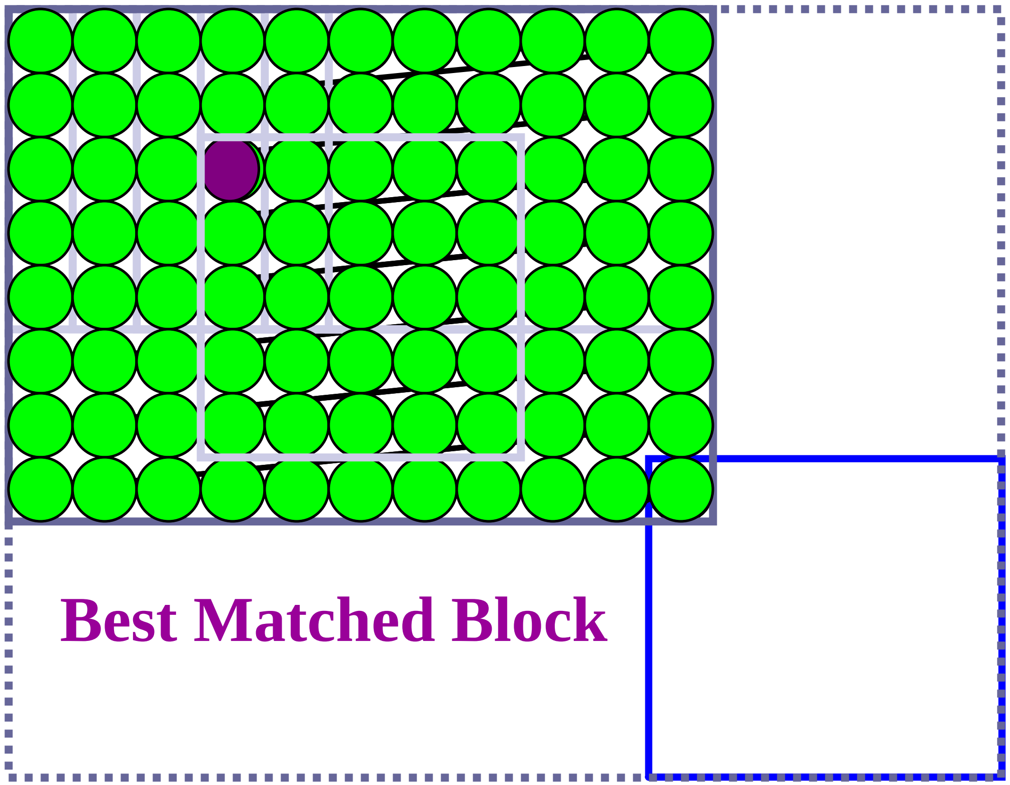


Current Block

Search Range

Reference Block (Candidate Block)

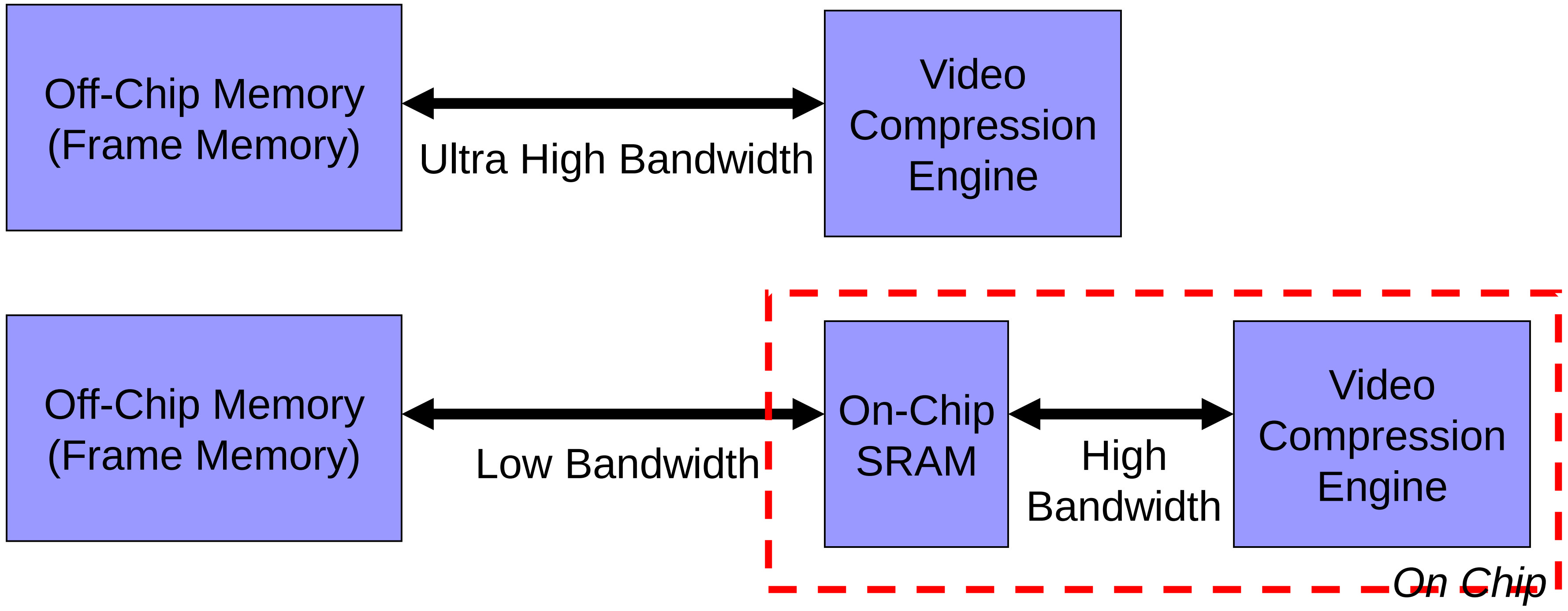
Candidate Search Position (Search Location)



$$SAD(i,j) = \sum_{k=1}^{N} \sum_{l=1}^{N} \left| x_t(k,l) - x_{t-1}(k+i, l+j) \right|$$

On-Chip SRAM

- The off-chip memory bandwidth can be dramatically reduced with on-chip memory



On-Chip SRAM

- If we can buffer current block pixels and search area pixels on the on-chip SRAM, we can **significantly decrease the required bandwidth on system bus** (external RAM)
  - **Data reuse** of search area pixels can further reduce the bandwidth of system bus
- Act **like cache** memory in CPU
- This is a **trade-off between area and bandwidth**
- In the following discussions, we assume block size is N x N, and search range is [-P, +P-1]

Different Schemes of Data Reuse for Search Area Pixels

- Data reuse between different rows of candidates in one column of a block (scheme A)
- Data reuse between adjacent columns of candidates in a block (scheme B)
- Data reuse between adjacent blocks in one row of block (scheme C)
- Data reuse between different rows of block (scheme D)
- In today’s technology, scheme C is mostly used.

Illustration of Scheme A

- Data reuse between different rows of candidates in one column of a block

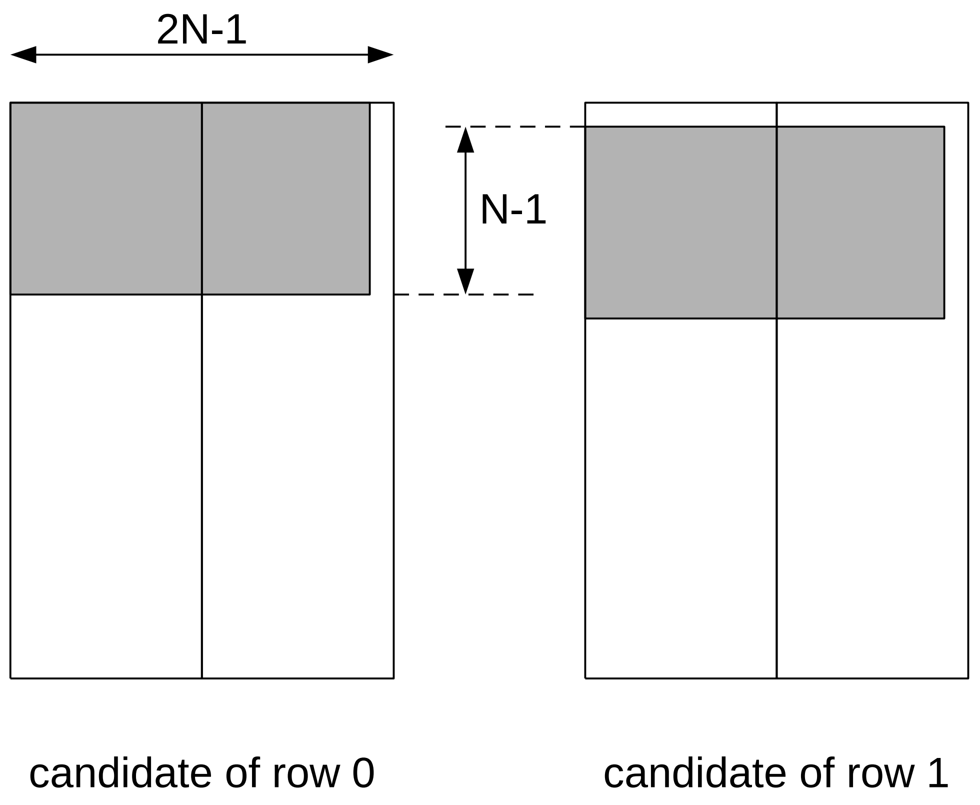


Illustration of Scheme B

- Data reuse between adjacent columns of candidates in a block

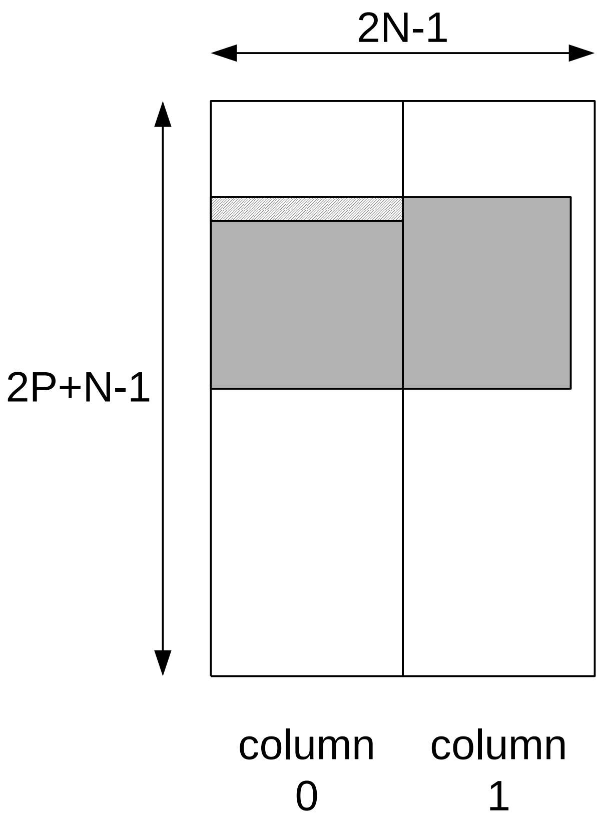


Illustration of Scheme C

- Data reuse between adjacent blocks in one row of block

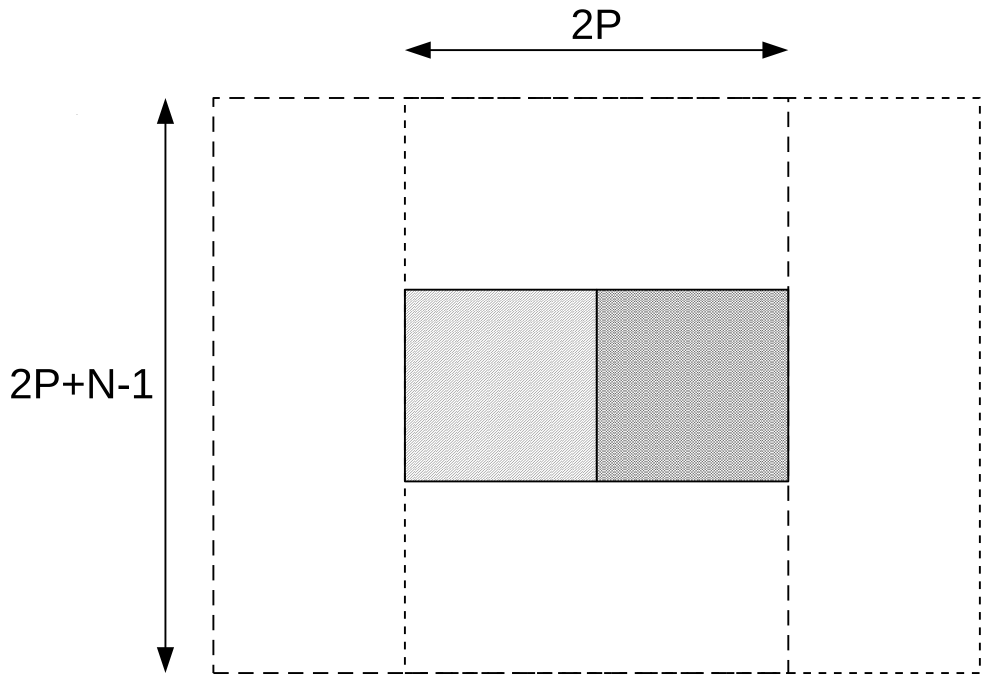
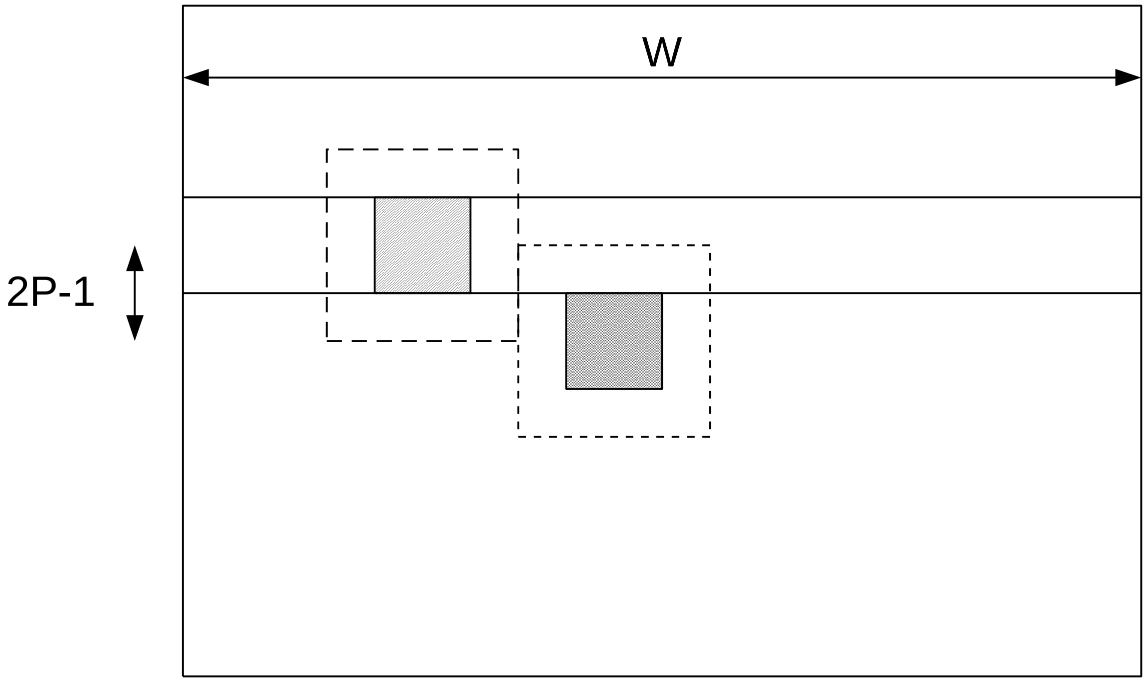


Illustration of Scheme D

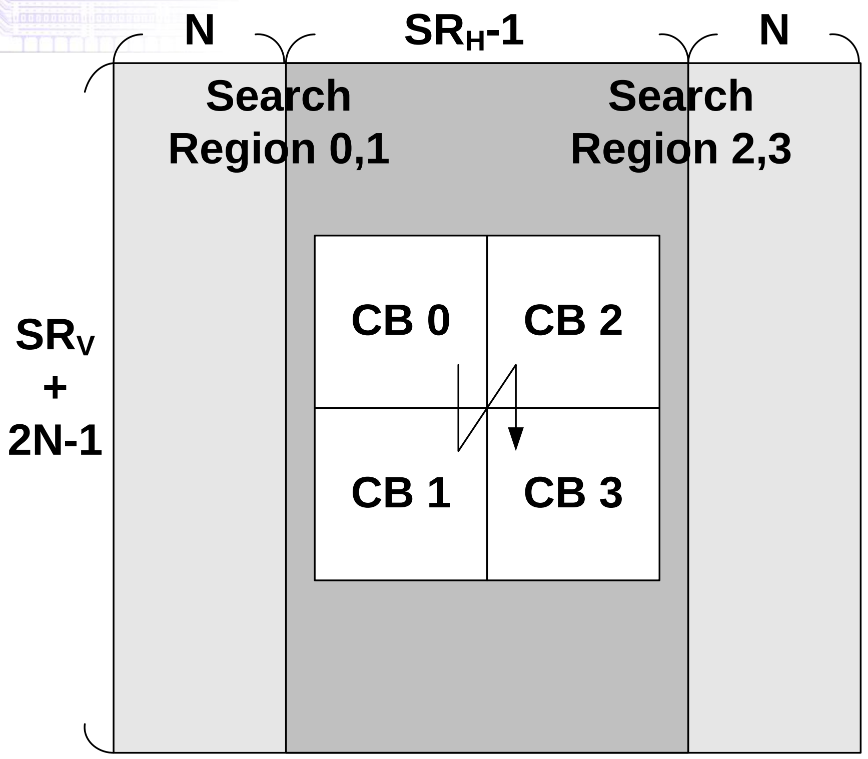
- Data reuse between different rows of block



Comparison of Different Schemes of Search Area Data Reuse

	Scheme A	Scheme B	Scheme C	Scheme D
On-chip buffer size (bytes)	(2N-1) x (N-1)	N x (2P+N-1) + N x (N-1)	Max{2N, 2P} x (2P+N-1)	W x (2P-1) + 2P x N
Off-chip to on-chip (times/pixel)	$(2P/N+1)^2$ x (2P/N)	(2P/N+1) x (2P/N)	2P/N+1	1
On-chip to core (times/pixel)	2NP / (2P+N-1)	2NP / (2P+N-1) x 2	2NP / (2P+N-1) x (2P/N+1)	2P x (2P/N+1)

Level C+ Data Reuse



- Conventional data reuse schemes are based on raster scan
- By use of stripe scan
  - Stitch n successive vertical MBs (n-stitched)
  - Load their searching ranges
  - Partially reuse vertical data

Ref: Ching-Yeh Chen, Chao-Tsung Huang, Yi-Hau Chen, and Liang-Gee Chen, "Level C+ data reuse scheme for motion estimation with corresponding coding orders," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 16, no. 4, pp. 553--558, April 2006.

Comparison

Data Reuse Scheme	Bandwidth (Ea)	SRB
Level C scheme	$1+\frac{SR_V}{N}$	$(SR_H+N-1)(SR_V+N-1)$
Level C+ scheme	$1+\frac{SR_V}{nN}$	$(SR_H+N-1)(SR_V+nN-1)$
Level D scheme	1	$(SR_H+W-1)(SR_V-1)$

- System memory bandwidth (equivalent access factor)

$$Ea_{ME}=\frac{Total\ \ memory\ \ bandwidth\ \ for\ \ reference\ \ frame}{processed\ \ current\ \ pixels}$$

- On-chip memory size (SRB)

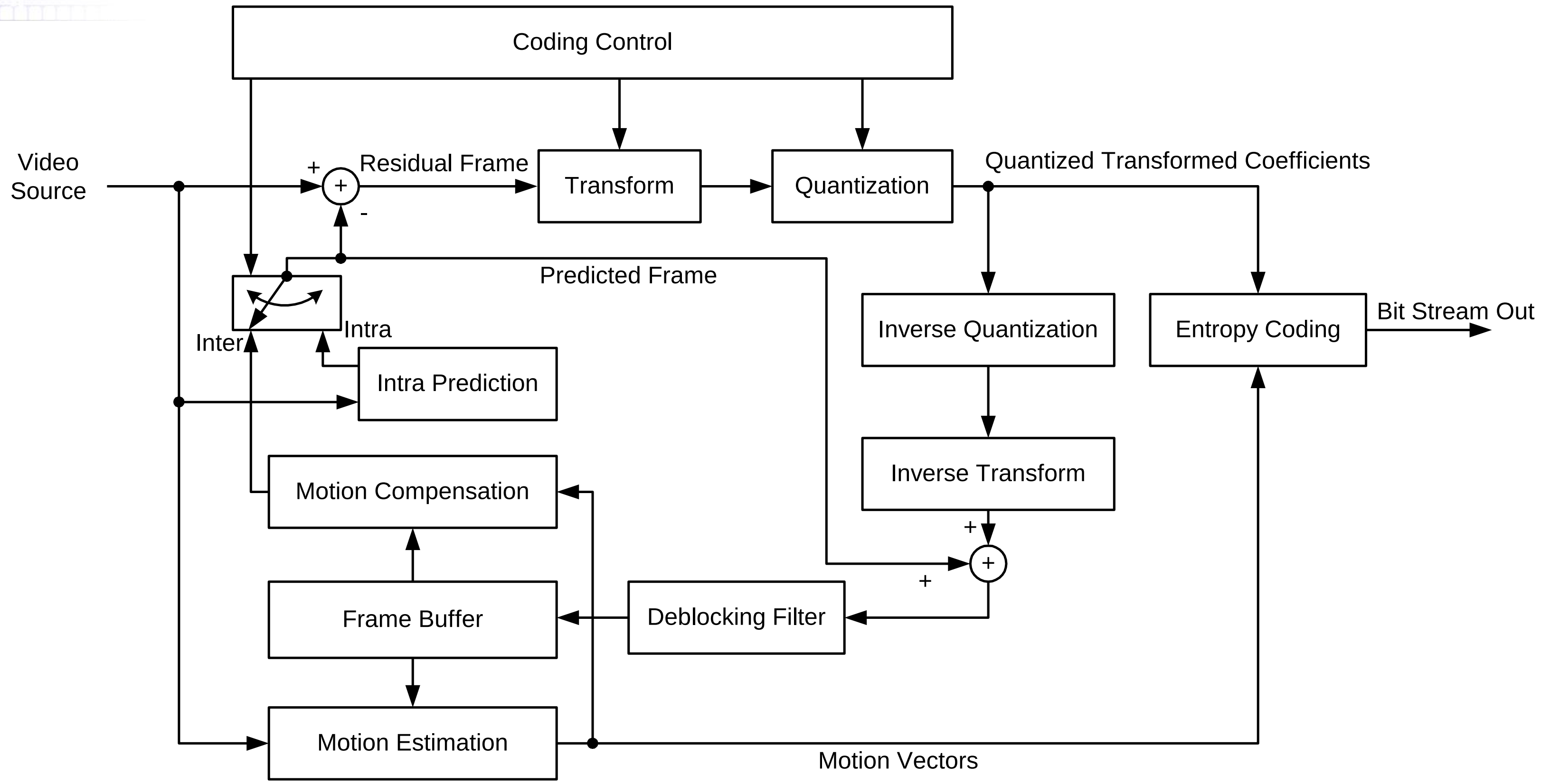
Accelerators

- Example: video coding accelerator

Ref: Shao-Yi Chien, Yu-Wen Huang, Ching-Yeh Chen, Homer H. Chen, and Liang-Gee Chen, "Hardware architecture design of video compression for multimedia communication systems," *IEEE Communications Magazine*, vol. 43, no. 8, pp. 122—131, Aug. 2005.

All the Standards are Based on Hybrid Coding!!

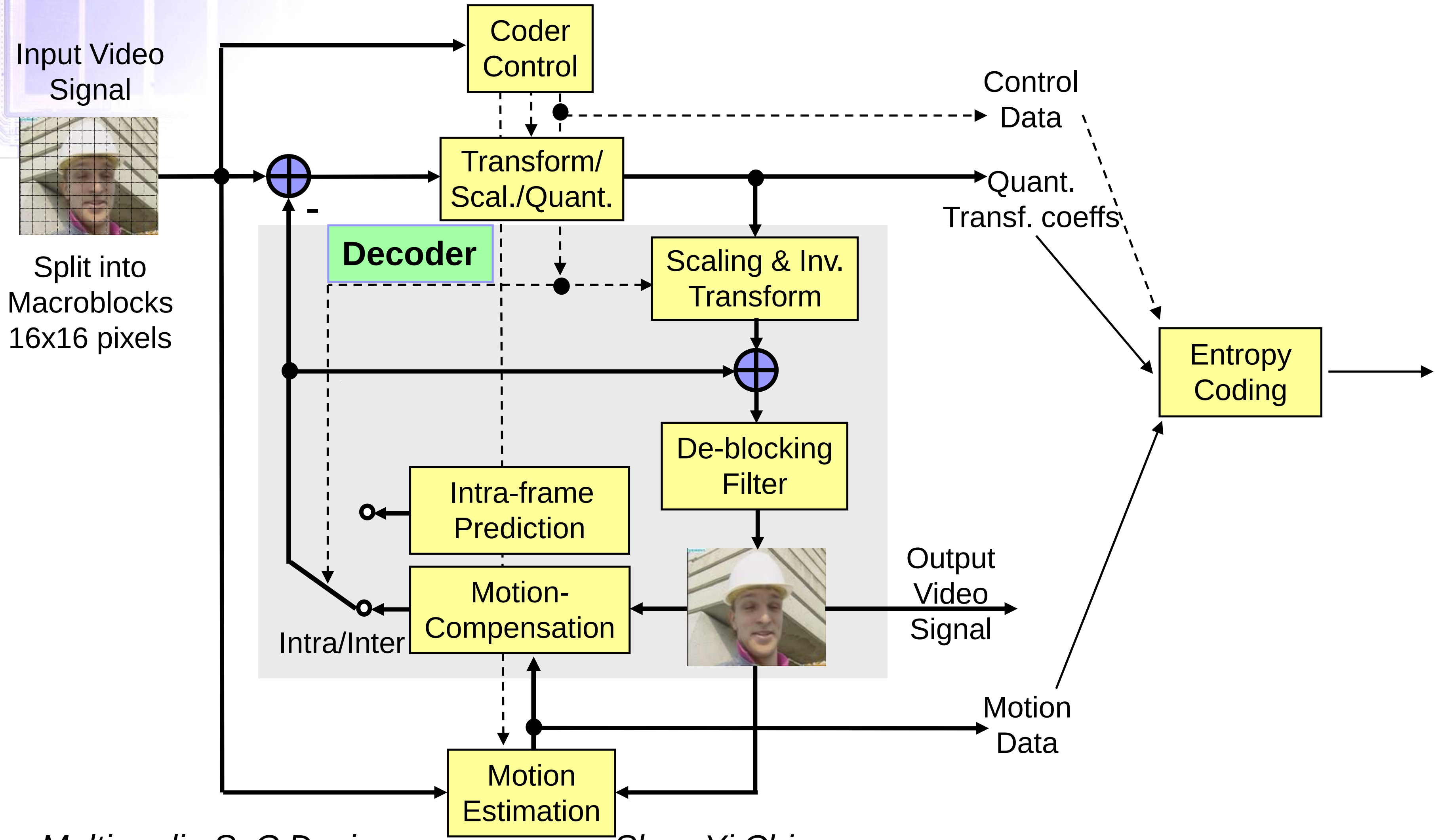
- A more complete hybrid coding model

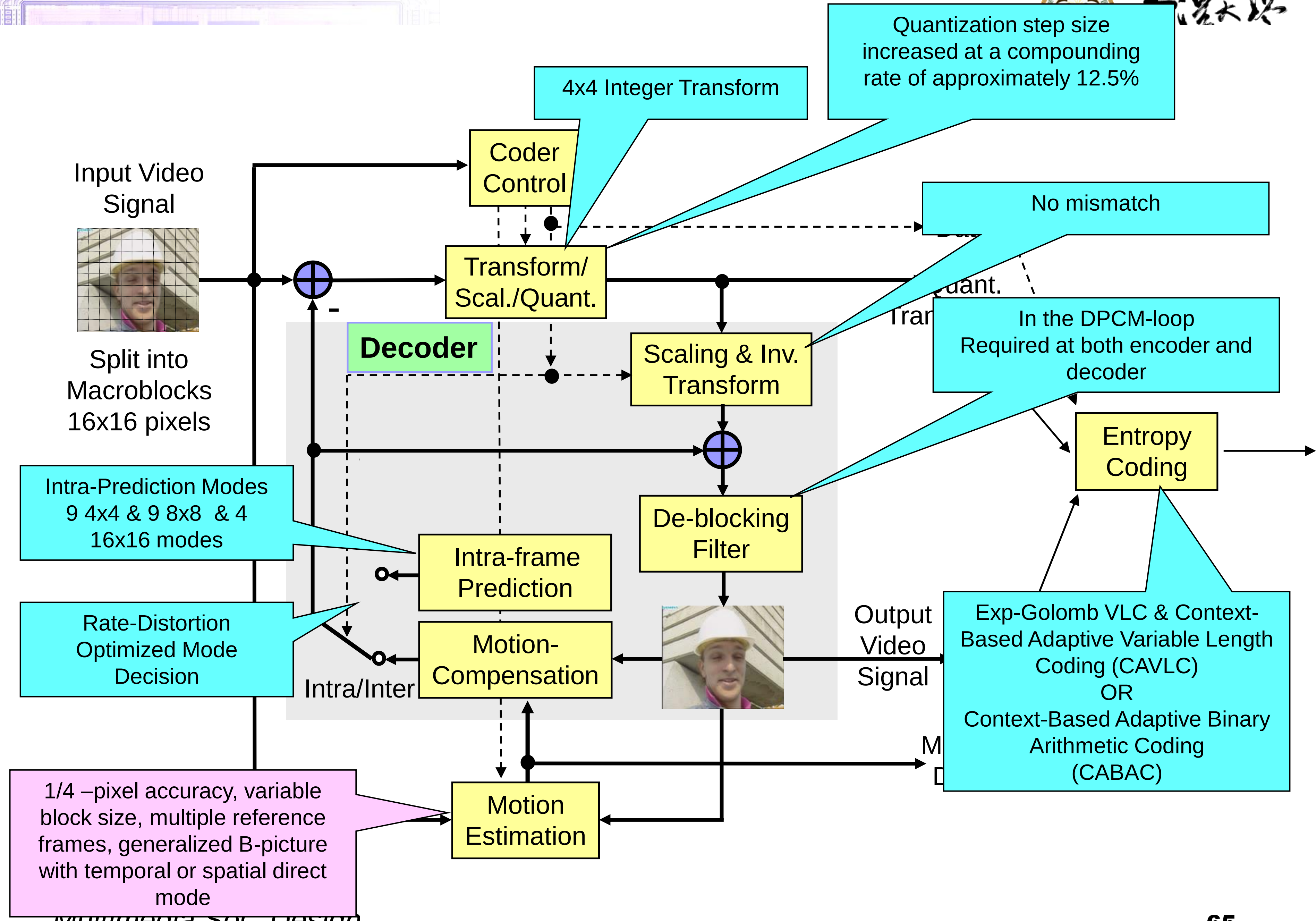


Comparison between Different Video Coding Standards

	Standards		
Modules	**MPEG-2**	**MPEG-4 ASP**	**H.264 Baseline Profile**
ME/MC			
-Block size	16x16	16x16 and 8x8	16x16, 16x8, 8x16, 8x8, 8x4, 4x8, and 4x4
-Quarter-pel precision	No	Yes	Yes
-Multiple reference frame	Up to 2	Up to 2	Yes (5 reference frames)
Intra-Prediction	DC prediction	AC/DC prediction	Yes (9 modes for 4x4 blocks and 4 modes for 16x16 blocks)
Rate-Distortion Optimization (*)	No	No	Yes
Transform	8x8 DCT	8x8 DCT	4x4 integer transform
Entropy Coding	VLC	VLC	VLC and CAVLC
In-Loop Deblocking Filter	No	No	Yes

H.264: the State-of-the-Art Video Coding Standard



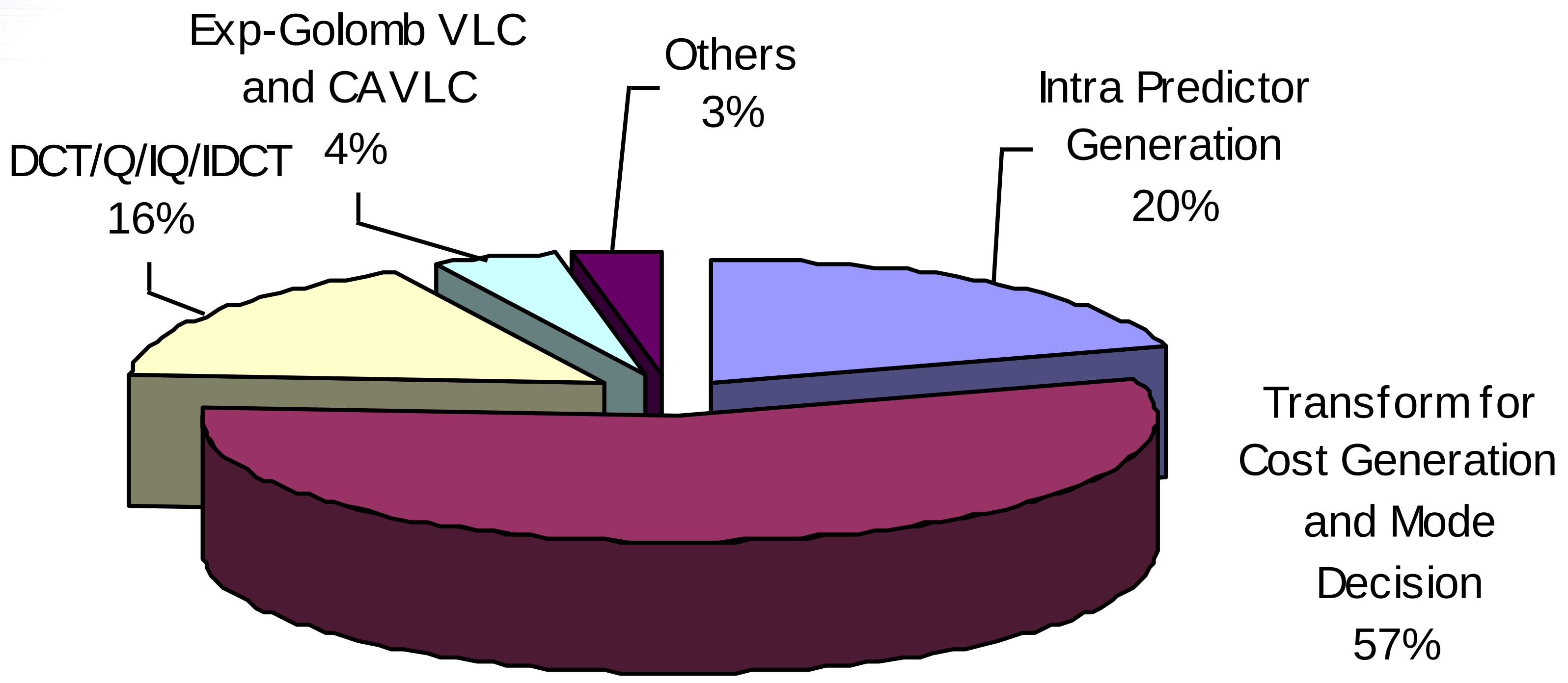

Quantization step size increased at a compounding rate of approximately 12.5%
4x4 Integer Transform
Coder Control
Input Video Signal
Split into Macroblocks 16x16 pixels
Transform/ Scal./Quant.
No mismatch
Decoder
Scaling & Inv. Transform
In the DPCM-loop Required at both encoder and decoder
Entropy Coding
Intra-Prediction Modes 9 4x4 & 9 8x8 & 4 16x16 modes
De-blocking Filter
Intra-frame Prediction
Rate-Distortion Optimized Mode Decision
Motion-Compensation
Intra/Inter
Output Video Signal
Exp-Golomb VLC & Context-Based Adaptive Variable Length Coding (CAVLC) OR Context-Based Adaptive Binary Arithmetic Coding (CABAC)
1/4 –pixel accuracy, variable block size, multiple reference frames, generalized B-picture with temporal or spatial direct mode
Motion Estimation
-

Instruction Profiling

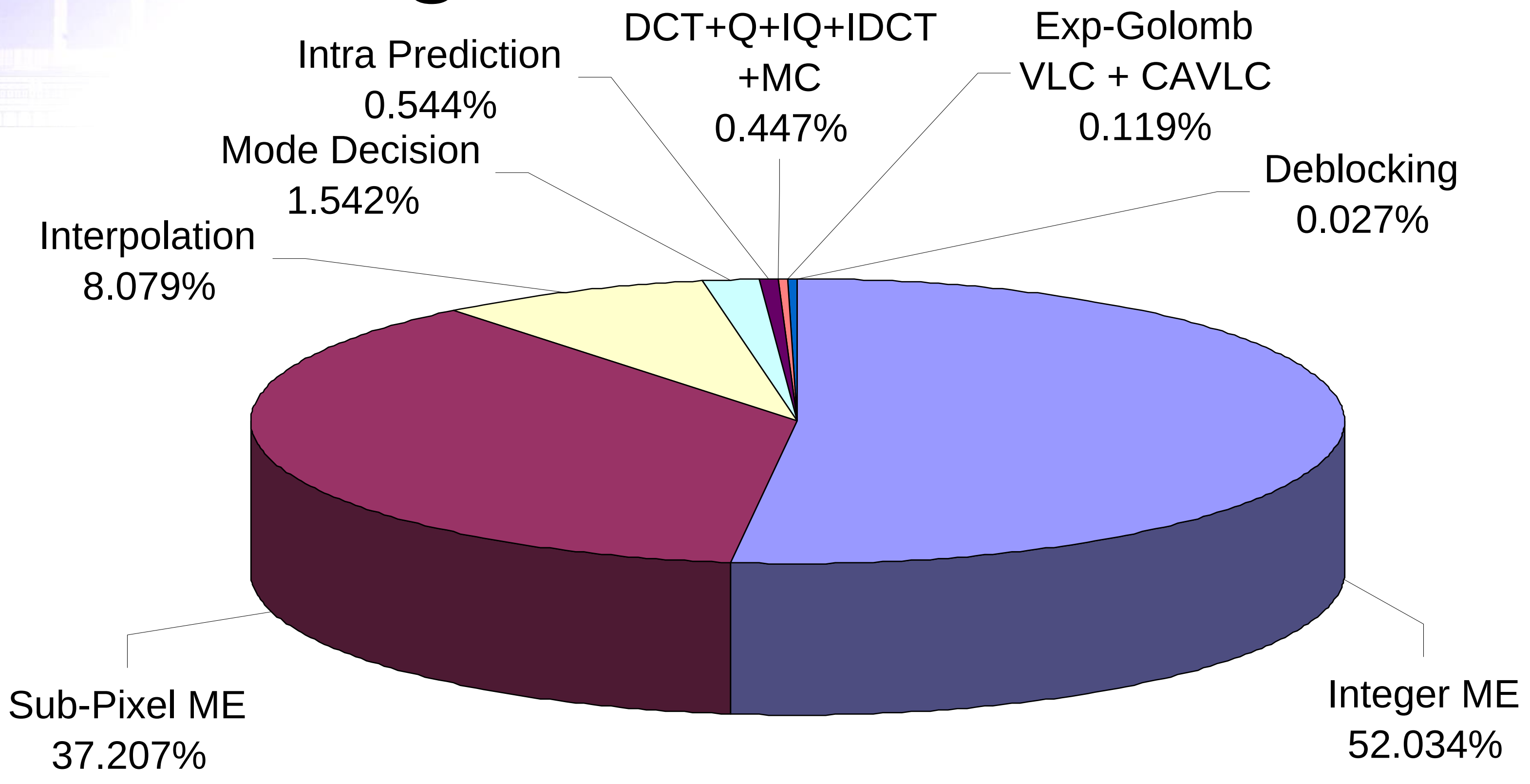
- Sun Blade 2000 with Ultra Sparc II 1GHz CPU running Solaris 8 operating system

Specifications	Encoding Parameters	Computing Power	Memory Access
CIF **352x288 30fps**	5 reference frames [-16,15]	**0.315TIPS**	**0.471TB/s**
D1 **720x480 30fps**	4 reference frames Ref0 H[-64,63] V[-32,31] Ref1-3 H[-32,31] V[-16,15]	**2.472TIPS**	**3.796TB/s**
HDTV720p **1280x720 30fps**	1 reference frame H[-64,63] V[-32,31]	**3.604TIPS**	**5.566TB/s**

I-Frame Run-Time Percentages



P-Frame Run-Time Percentages



CIF 30fps, Baseline Profile, 5 Ref. [-16.75 +16.75]

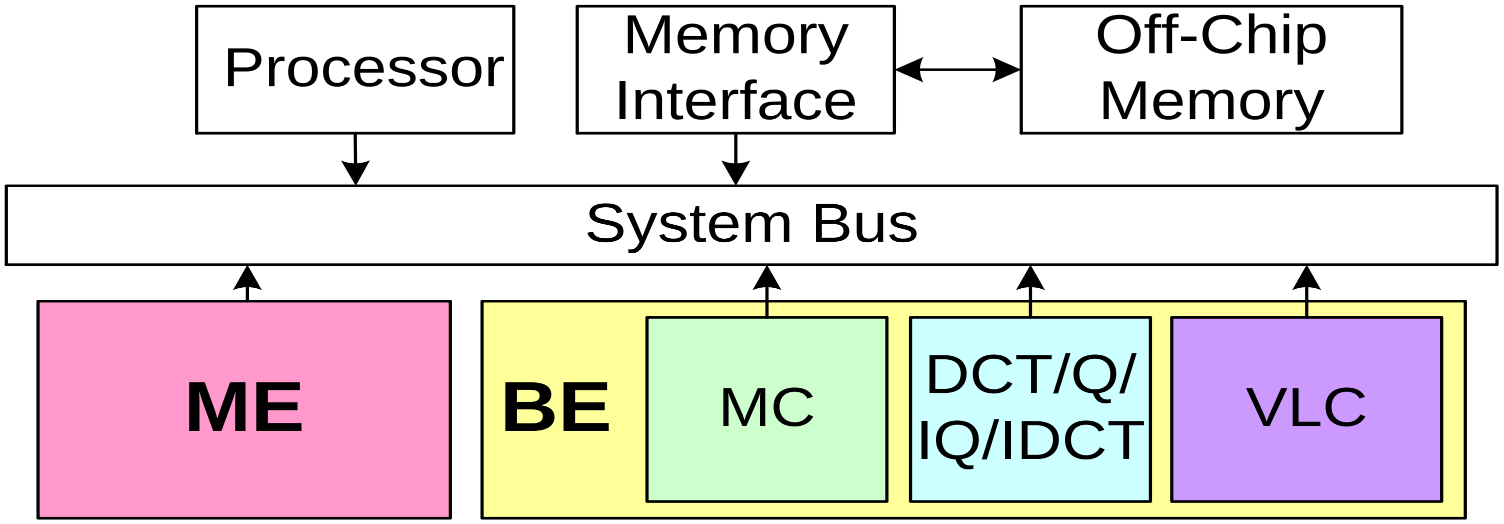
Design Challenge and Possible Solutions (1/2)

- Computational complexity and bandwidth requirement
  - Highly utilized parallel architectures
  - Efficient memory hierarchy combined with data sharing and data reuse schemes
- Sequential flow and data dependency
  - The enemy of parallel processing
  - MB-based pipelined structure
  - Apply modified hardware-oriented algorithms
  - Careful lifetime analysis

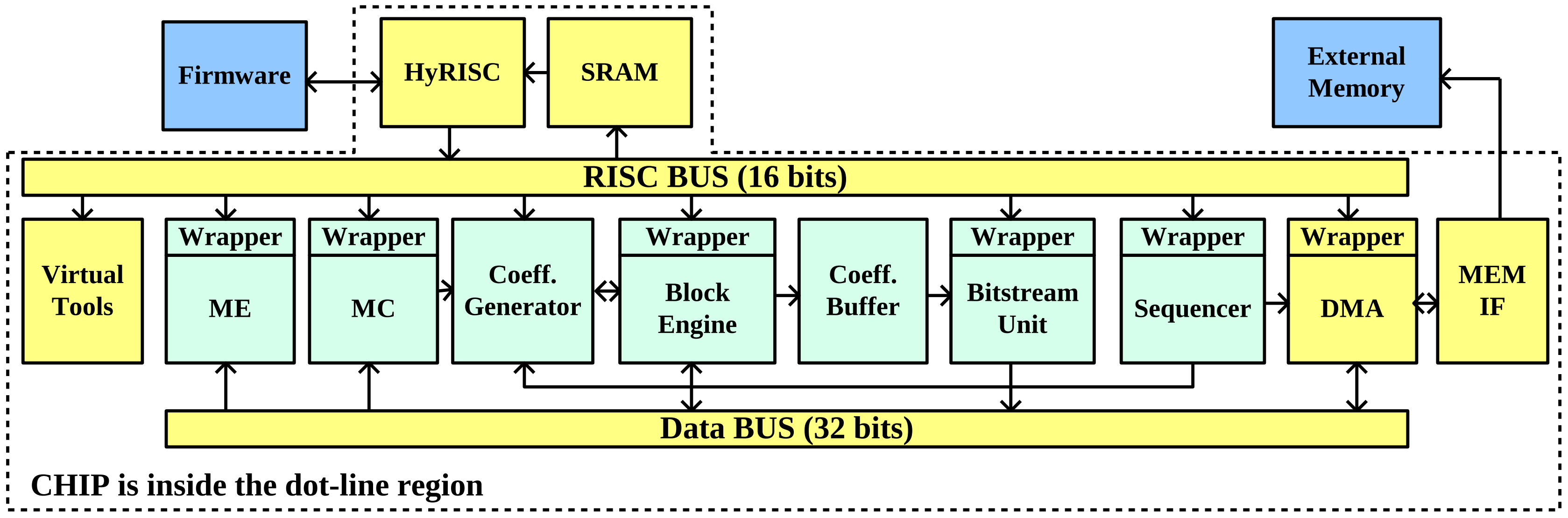
Design Challenge and Possible Solutions (2/2)

- Coding Loops
  - Not only frame-level reconstruction loops but also MB-level and block-level reconstruction loops
  - Lead to high latency and reduce the hardware utilization
  - Carefully scheduling and buffer design
- Abundant Modes
  - May lead to large hardware cost
  - Use unified architecture and reconfigurable architecture to reduce the hardware cost

Conventional System Architecture with Two-Stage Macroblock Pipelining

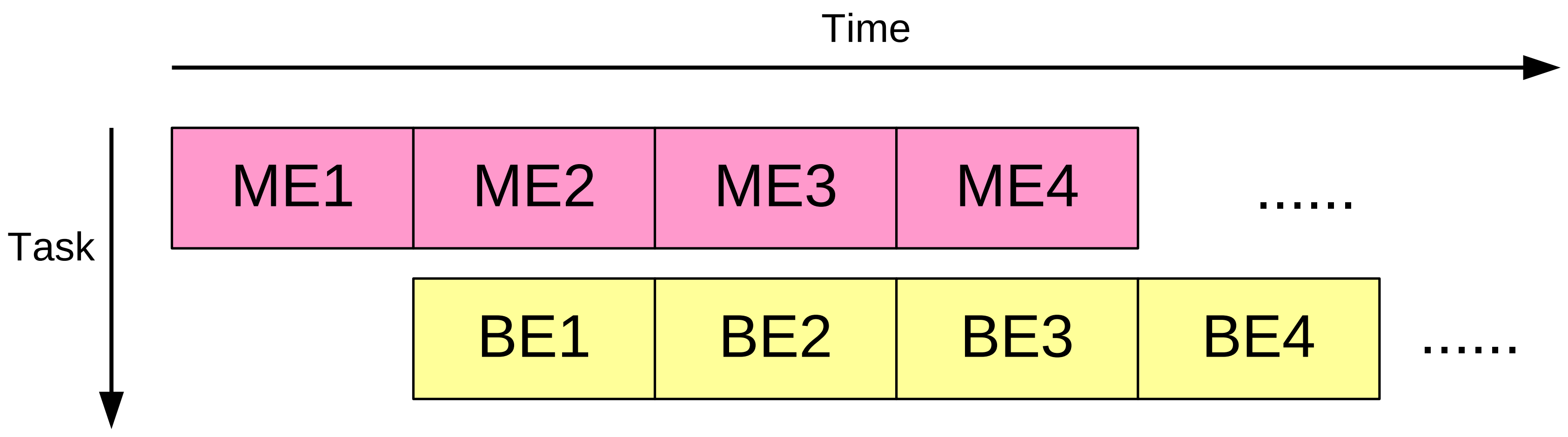


MPEG-4 Encoding System



Scheduling of the MPEG-4 Encoder

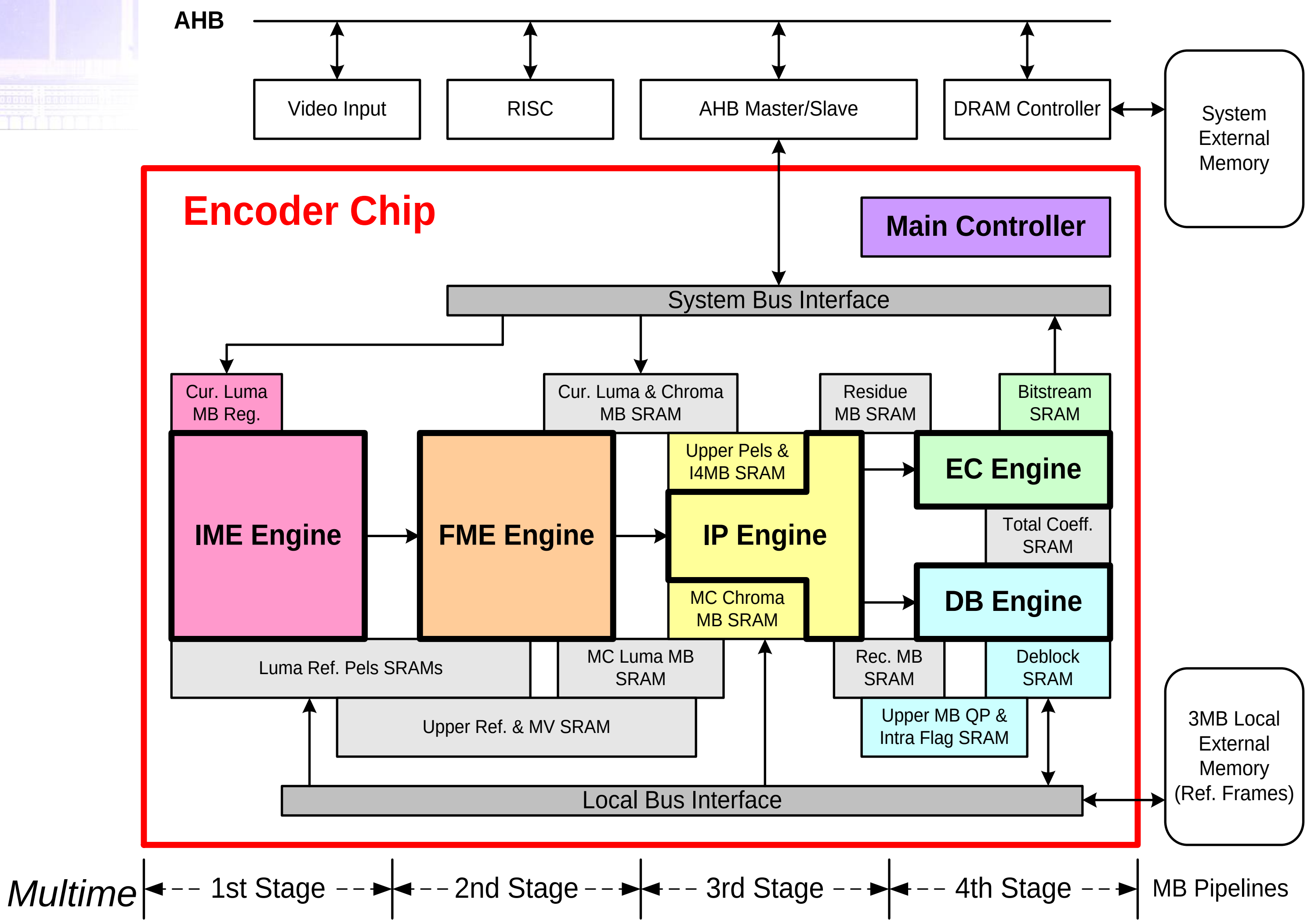
- Two macroblocks are processed simultaneously



Problems of Conventional Macroblock Pipelining for H.264/AVC

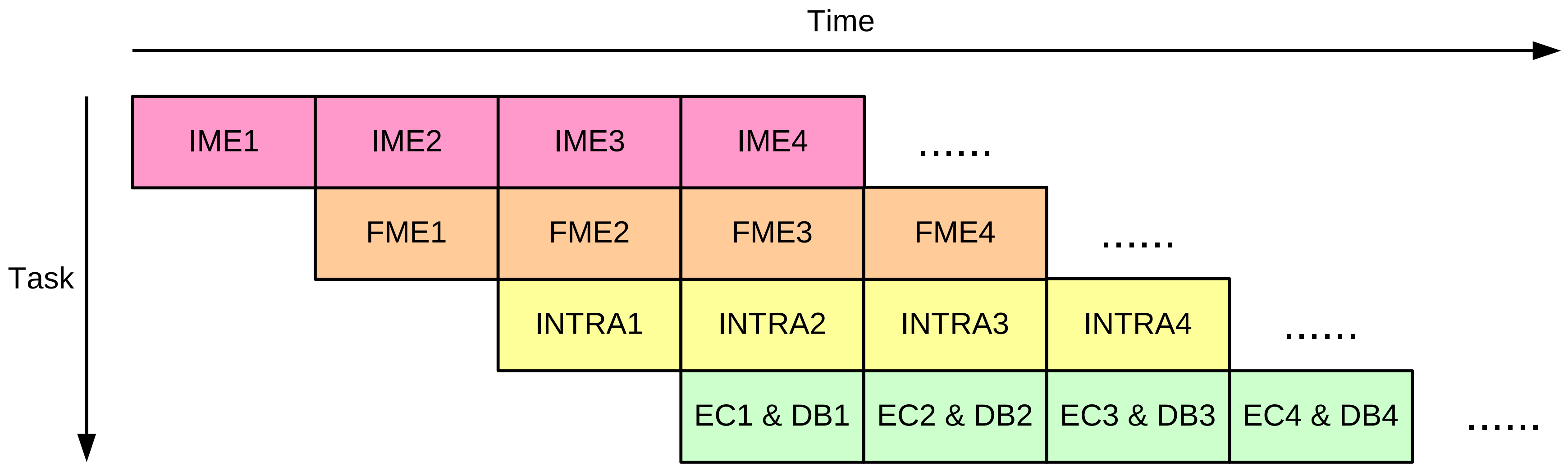
- Low throughput
  - Complex encoding algorithm
- Low utilization
  - Difficult resource sharing for integer motion estimation, fractional motion estimation, and intra prediction.
- High bandwidth
  - MC operations
  - Mode decision information
- Feasibility
  - Intra prediction and DPCM loop
- Require new MB pipelining and efficient modules

Proposed System Architecture with Four-Stage Macroblock Pipelining



Scheduling of the H.264 Encoder

- Four-stage pipelining



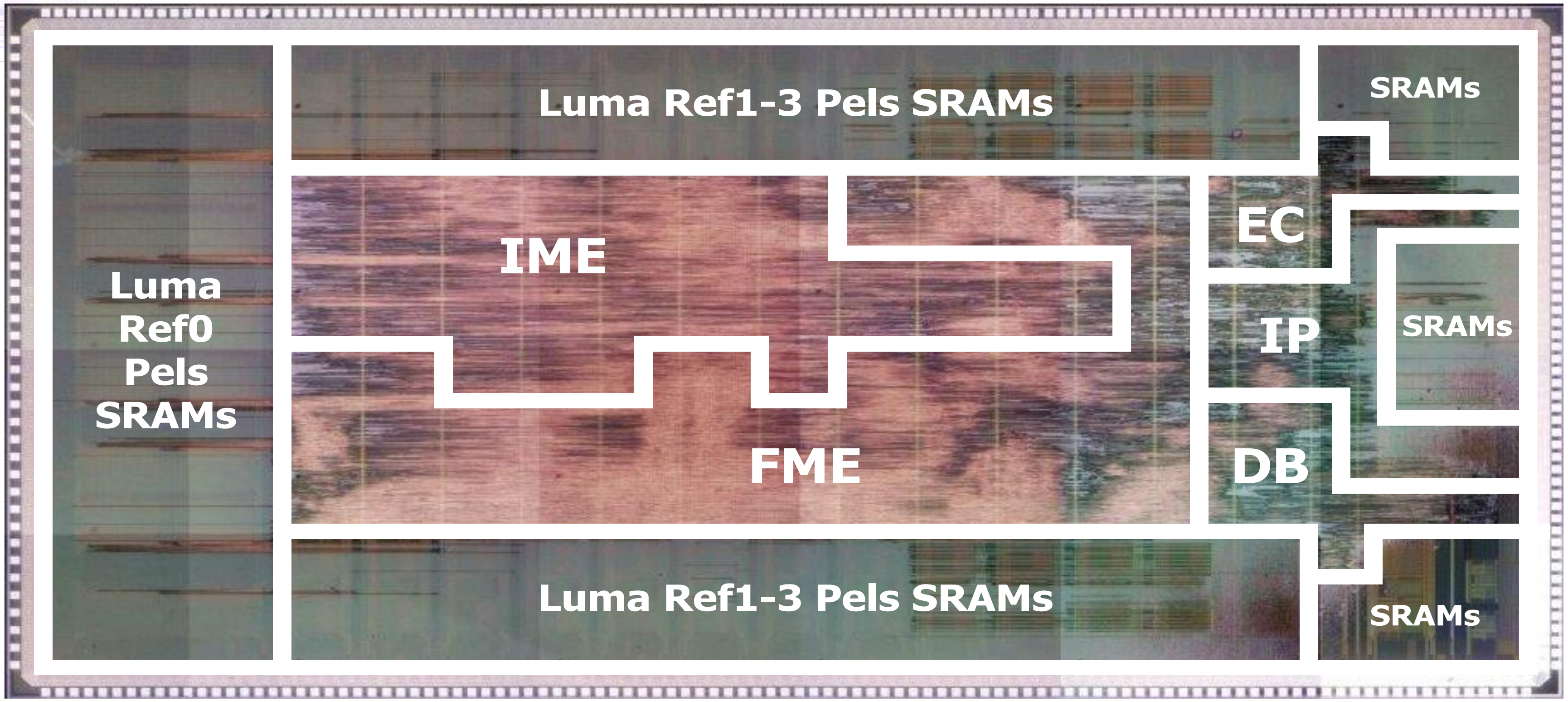
Features of the Proposed System Architecture (1/2)

- Prediction stage partitioned to IME, FME, and IP
- IME and FME share search area SRAMs.
- FME includes inter mode decision and luma MC.
- IP integrates DPCM loop, intra mode decision, intra/inter selection, chroma MC, generating residues and reconstructed pixels.
- EC and DB at the 4th stage
- MB data through IME, FME, IP, EC/DB
- Four MBs simultaneously processed

Features of the Proposed System Architecture (2/2)

- Video signals and parameters inputted, and bitstream outputted via system bus
- Reference frames in/out via local bus
- 40MB/s and 240MB/s for the system bus and local bus, respectively
- Balanced cycles for high utilization
- Local transfer to reduce bus traffic
- Double utilization and throughput compared with the conventional MB pipelining

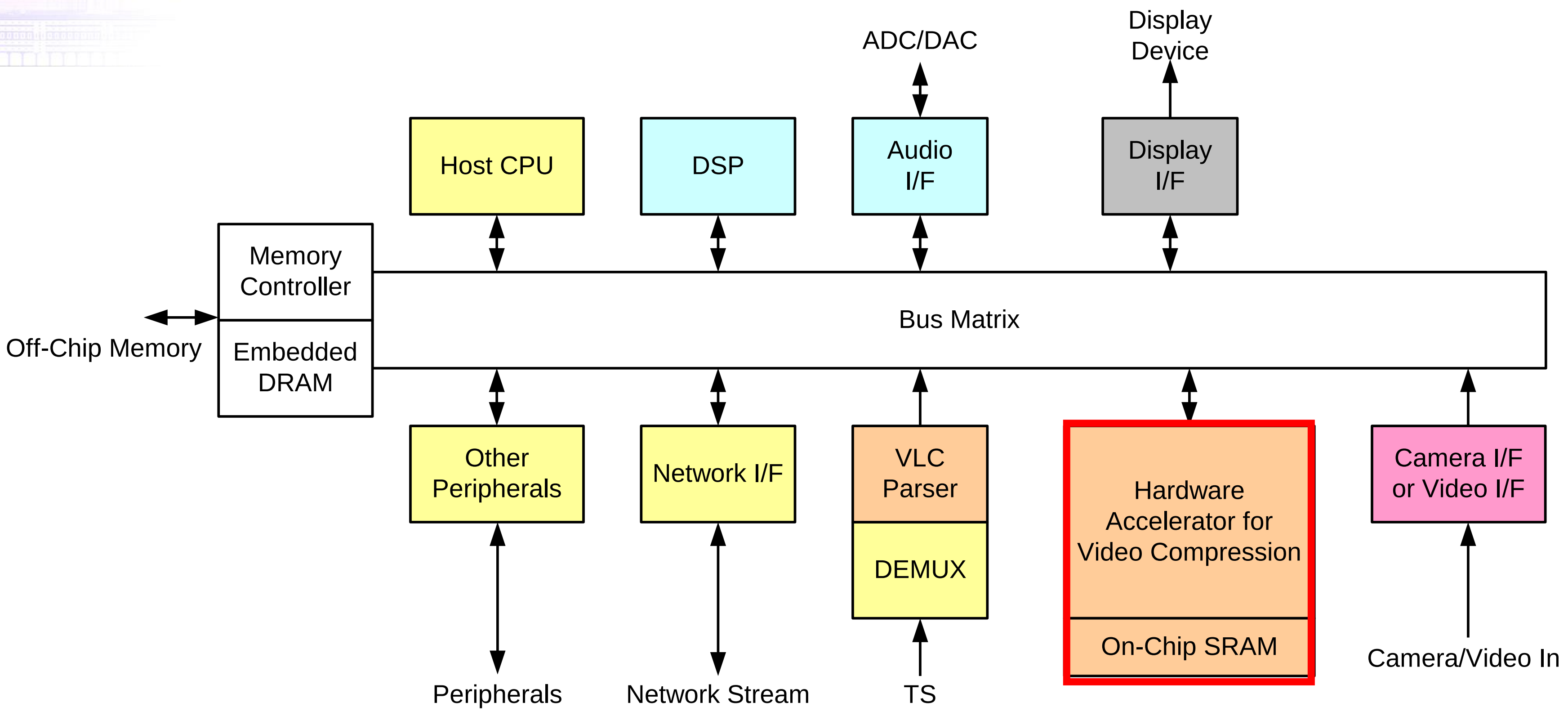
Chip Photo


Luma Ref1-3 Pels SRAMs
SRAMs
Luma Ref0 Pels SRAMs
IME
EC
IP
SRAMs
FME
DB
Luma Ref1-3 Pels SRAMs
SRAMs

Chip Features

Technology	UMC 0.18μm CMOS 1P6M
Supply Voltage	1.8V
Core Area	7.68×4.13mm^2
Logic Gates	922.8K (2-input NAND gate)
SRAMs	34.72KB
Encoding Features	All Baseline Profile Compression Tools
Max. Number of Ref. Frames	4
Max. Search Range (Ref. 0)	H[-64,+63] V[-32,+31]
Max. Search Range (Ref. 1-3)	H[-32,+31] V[-16,+15]
Operating Frequency	81MHz for D1
	(720×480, 30Frames/s, 4 Ref. Frames, Max. Search Range)
	108MHz for HDTV720p
	(1280×720, 30Frames/s, 1 Ref. Frame, Max. Search Range)
Power Consumption	581mW for D1
	785mW for HDTV720p

Multimedia Communication System



Hardware Architecture Exploration (1/2)

- Much more complex
  - The selection of CPU, DSP, and memory module
  - The performance of memory controller
  - The selection of I/O devices and communication channels
  - Bus matrix architecture
  - Hardware architecture for hardware accelerator for video compression

Hardware Architecture Exploration (2/2)

- From hardware accelerator point of view
  - Is the computational power enough?
  - Is the bandwidth enough?
  - May change the bus matrix architecture and the architecture of the hardware accelerator
- Highly depends on the target applications/specifications

Examples (1/2)

- IP camera with CIF 30fps
  - CIF 30 fps video with bitrate of 128 Kbps and single channel 16-bit 44.1 samples/s audio with bitrate of 32 Kbps,
  - Host CPU need to handle server tasks
    - Need a powerful CPU, maybe ARM9
    - All the hardware modules of video coding are required
  - Input: camera I/F
  - Output: Network I/F
  - Display I/F, VLC parser, and DEMUX can be removed
  - System bandwidth: 20 MBps
  - Single 16-bit bus at 20MHz should be enough

Examples (2/2)

- DVB
  - 1920x1080 30 fps video with bitrate of 10 Mbps and 5.1 channel 16-bit 48 samples/s audio with bitrate of 384 Kbps
  - Input: TS stream
  - Output: display I/F (maybe with 2D graphics ability)
  - Network I/F and camera I/F can be removed
  - Only video decoding related engines are kept in the hardware accelerator
  - Bus bandwidth: >300MBps
  - For 50MHz, multiple 32-bit buses are required