Development of Full-HD Multi-standard Video CODEC IP Based on Heterogeneous Multiprocessor Architecture

H.Nakata[1], K.Hosogi[1], M.Ehama[1], T.Yuasa[1], T.Fujihira[1]
K.Iwata[2], M.Kimura[2], F.Izuhara[2], S.Mochizuki[2], M.Nobori[2]

[1]Embedded System Platform Laboratory
Central Research Laboratory
Hitachi, Ltd.

[2]System Design Div.
System Solution Business Group
Renesas Technology Corp.

Agenda

Video codec trends

Video codec standards are increasing…

MPEG-1, MPEG-2, MPEG-4
H.263, H.264 (MPEG-4/AVC), VC-1, etc.

Video resolution becomes high…

Many consumer devices are supporting full-HD.

Digital TV

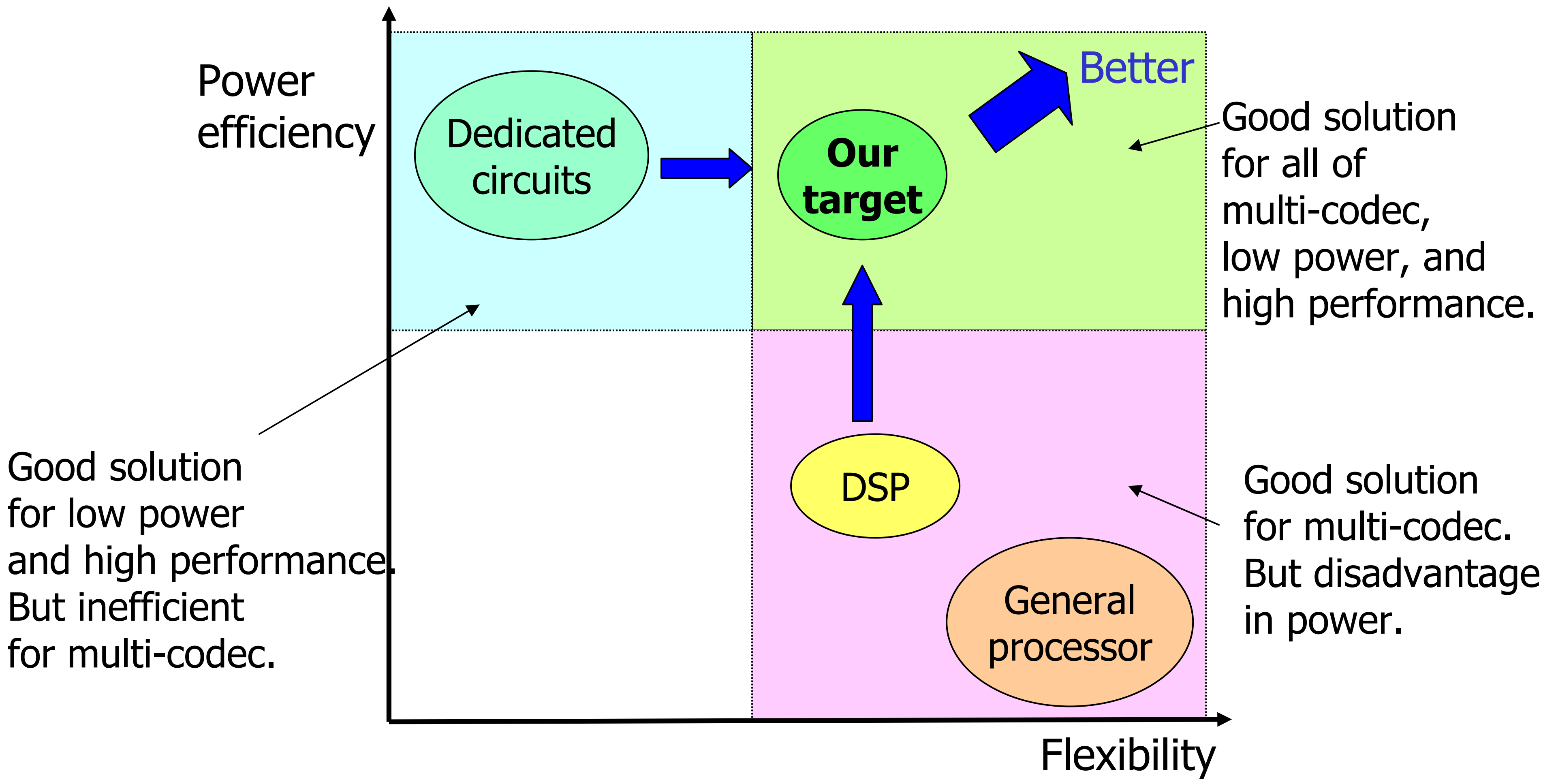
Digital Video Camera

Blu-ray Recorder

Digital Still Camera

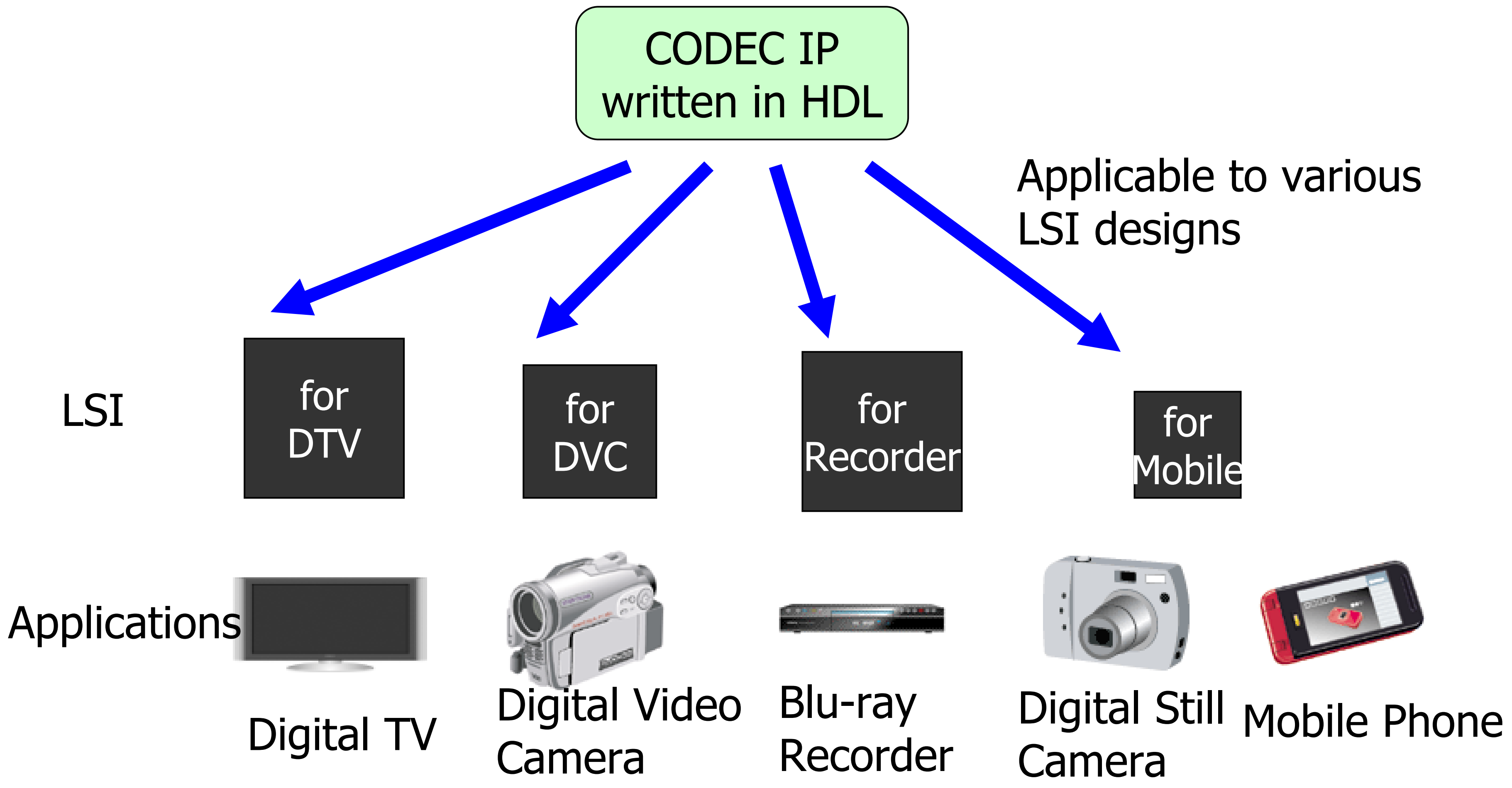
Mobile Phone

Our target for video CODEC


Power efficiency
Dedicated circuits
Our target
Better
Good solution for all of multi-codec, low power, and high performance.
Good solution for low power and high performance. But inefficient for multi-codec.
DSP
General processor
Good solution for multi-codec. But disadvantage in power.
Flexibility

We tried to apply a heterogeneous multiprocessor architecture to a video CODEC for our target.

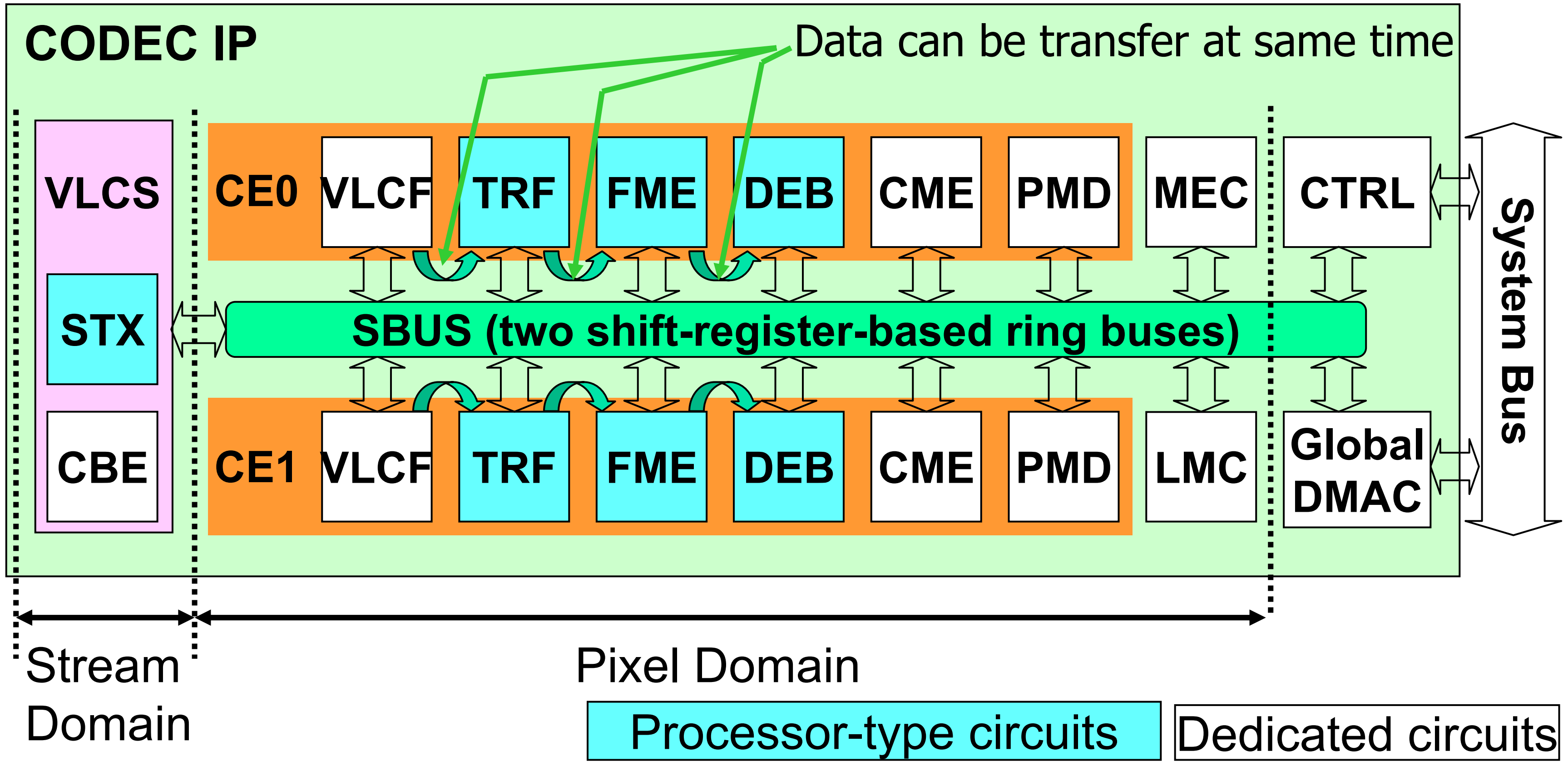
CODEC IP applicable to many purpose



HDL: Hardware Description Language

Top level architecture

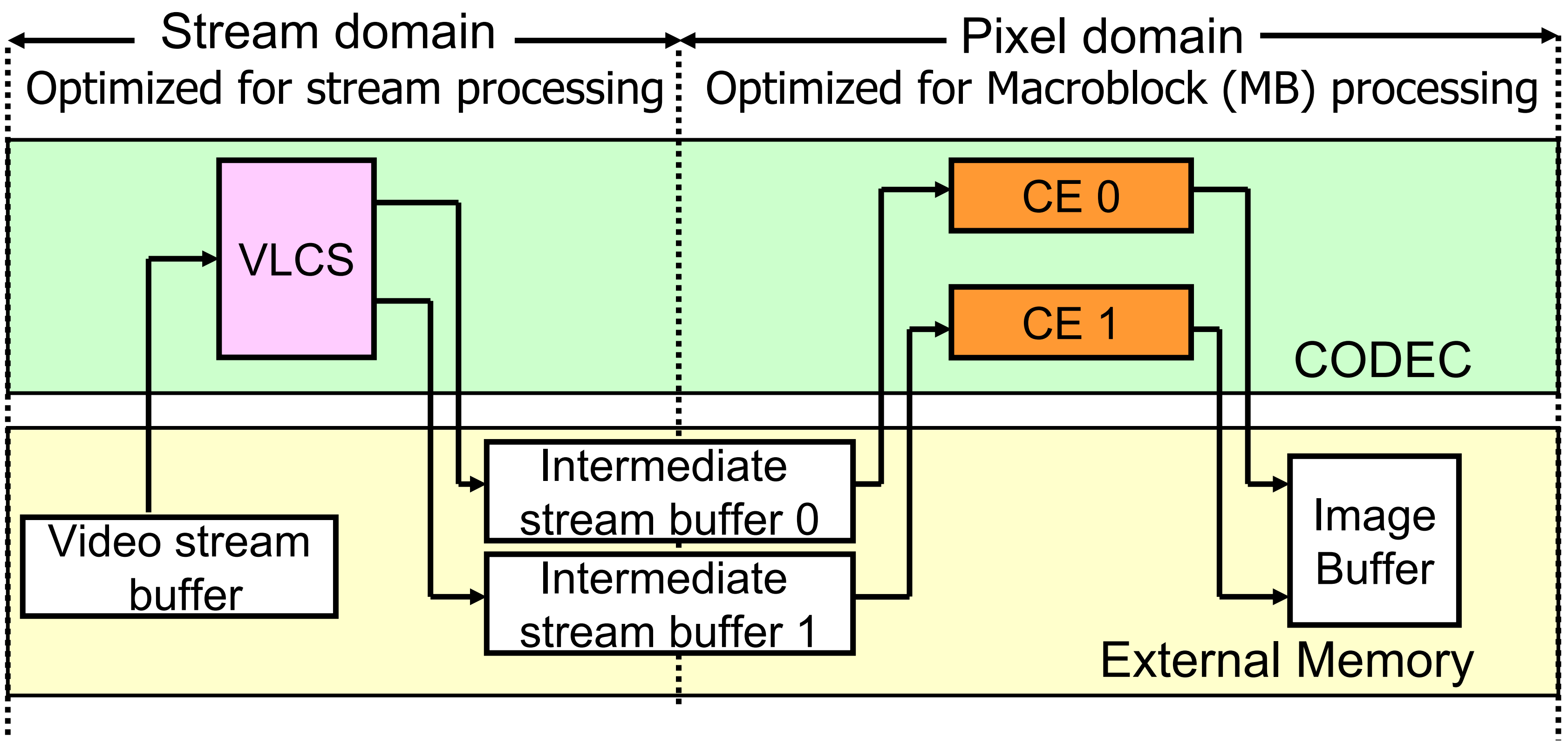
- All modules are connected to SBUS
- SBUS is structured with 2 unidirectional **shift-register-based** 64bit buses
- The directions of the 2 buses are opposite to each other
- Some of modules use **original programmable processors**



Separate stream domain and pixel domain

- Separate both domains by intermediate stream buffers

  ➡ Optimize performance for each domain

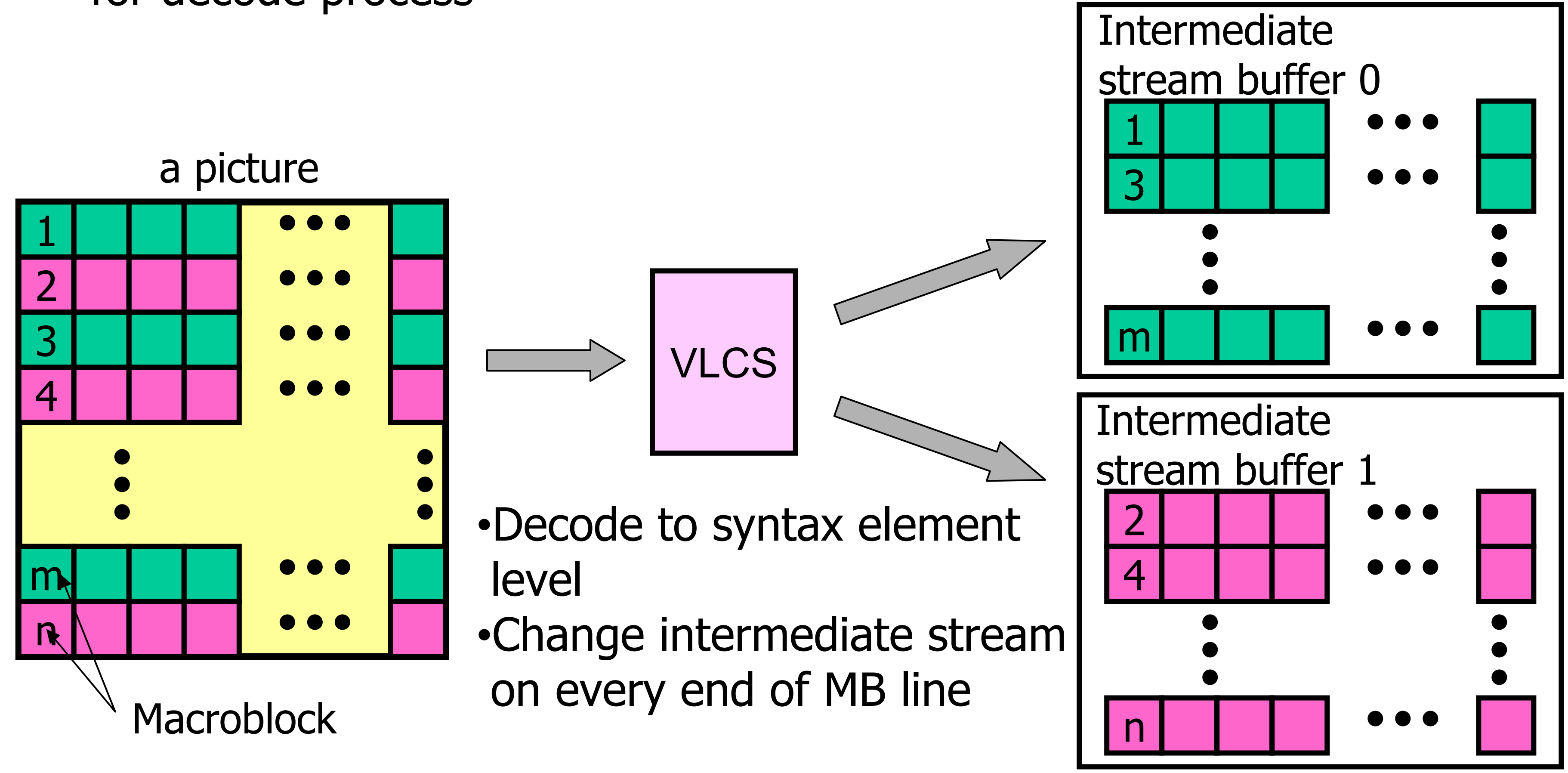

Stream domain
Optimized for stream processing
Pixel domain
Optimized for Macroblock (MB) processing
VLCS
CE 0
CE 1
CODEC
Intermediate stream buffer 0
Intermediate stream buffer 1
Video stream buffer
Image Buffer
External Memory

Note) This figure shows decode process.
Data transfer directions are opposite for encode process.

Distribute to plural intermediate streams

Pixel domain has 2 CEs which work in parallel

VLCS has to distribute an intermediate stream to both CEs for decode process

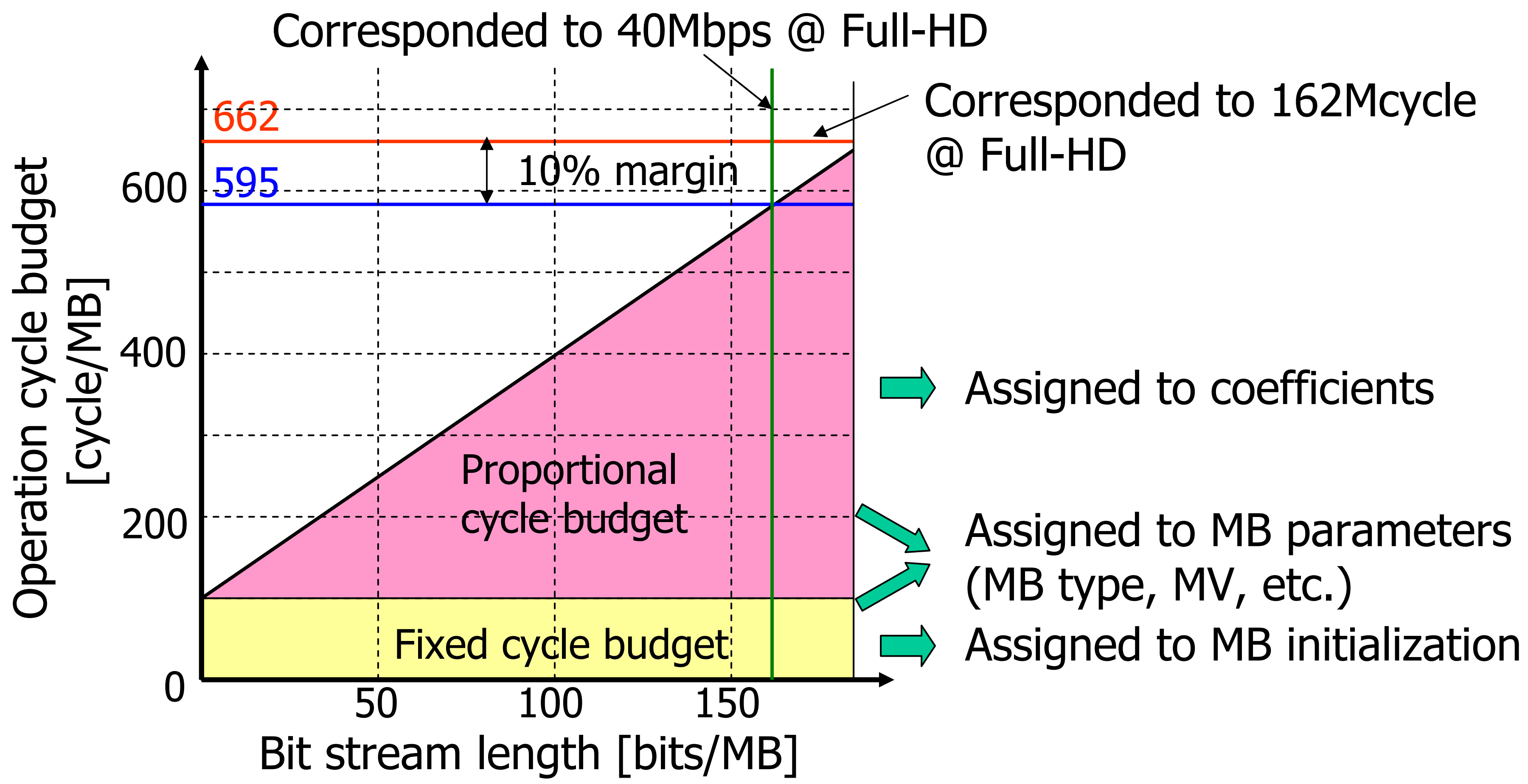


- Decode to syntax element level
- Change intermediate stream on every end of MB line

Note) This figure shows decode process. The data flow is opposite for encode process.

Stream domain operation cycle budgeting

- Performance target: 40Mbps for full-HD @ 162MHz operation



Reserve 100 fixed operation cycles per MB and assign 3 cycles/bit for bits in streams (This meets 40Mbps performance included 10% margin)

Intermediate stream compaction

- Intermediate stream is compacted by simple coding method
  - Coded by
    1. fixed length code (FLC)
    2. FLC – exp. golomb combined code (EGFLC)
  - EGFLC is used for coefficients and MVs.

- Intermediate stream can be encoded and decoded fast by simple logic
- Reduce size of intermediate buffer and bandwidth for intermediate data transfer
- EGFLC is about 20% smaller than normal exp. golomb code in our case.

Example of EGFLC

number	EGFLC		
	prefix		suffix
0		1	
1	0	1	0
2	0	1	1
3	00	1	00
4	00	1	01
5	00	1	10
6	00	1	11
7	00	0	000111
8	00	0	001000
…	00	0	xxxxxx

Similar to exp-golomb code

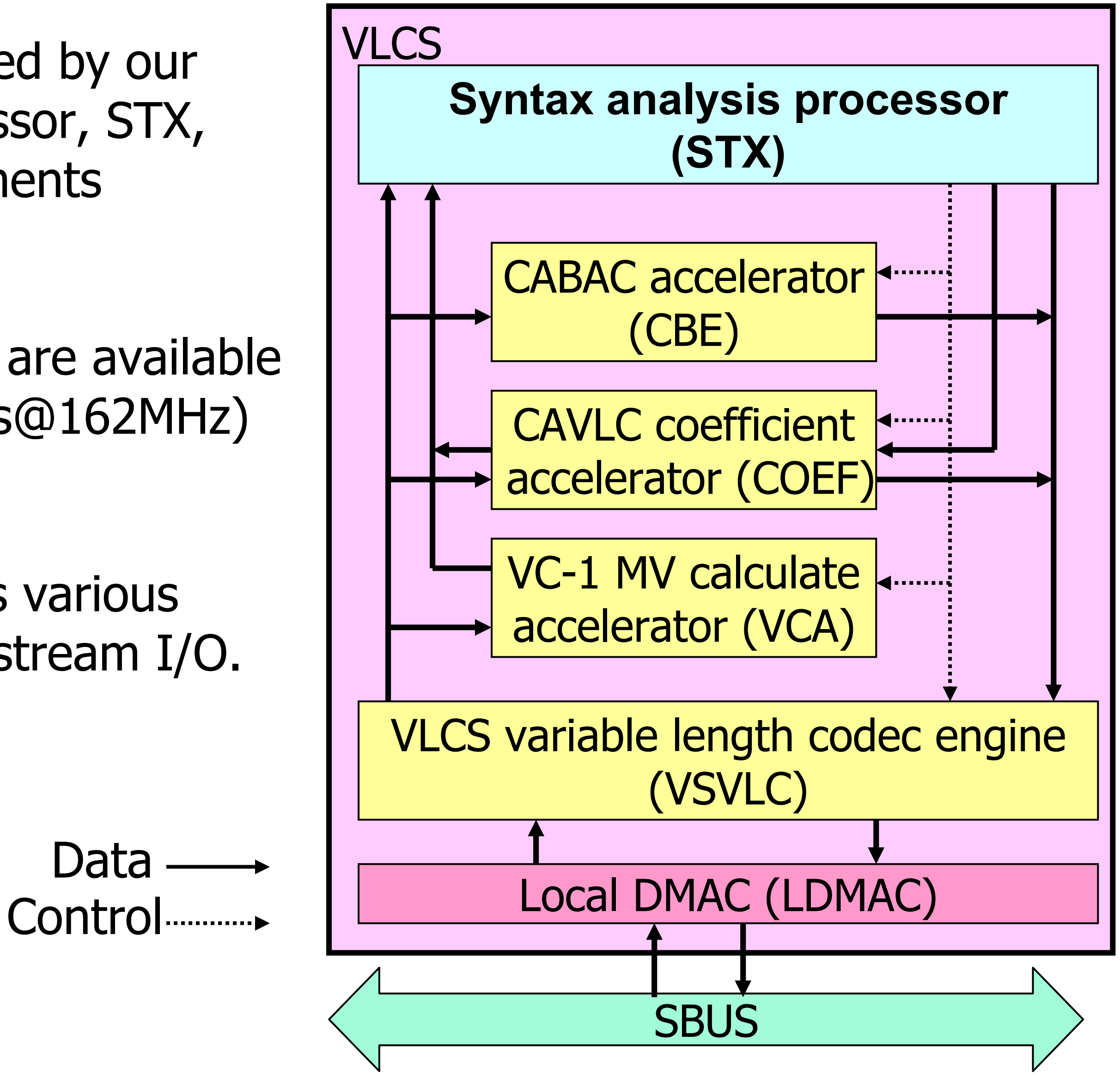
FLC is used as suffix

VLCS structure

• Stream syntax is analyzed by our original 2way LIW processor, STX, except some syntax elements

• Some dedicated circuits are available for performance (40Mbps@162MHz)

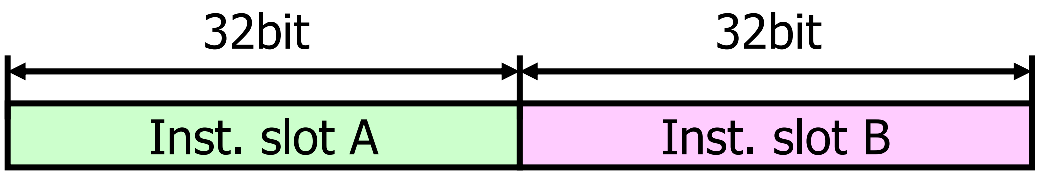
• VSVLC decodes/encodes various variable length code for stream I/O.


Data
Control
VLCS
Syntax analysis processor (STX)
CABAC accelerator (CBE)
CAVLC coefficient accelerator (COEF)
VC-1 MV calculate accelerator (VCA)
VLCS variable length codec engine (VSVLC)
Local DMAC (LDMAC)
SBUS

Syntax analysis processor (STX)

- Two 32bit instruction slots available

STX instruction slot assignments

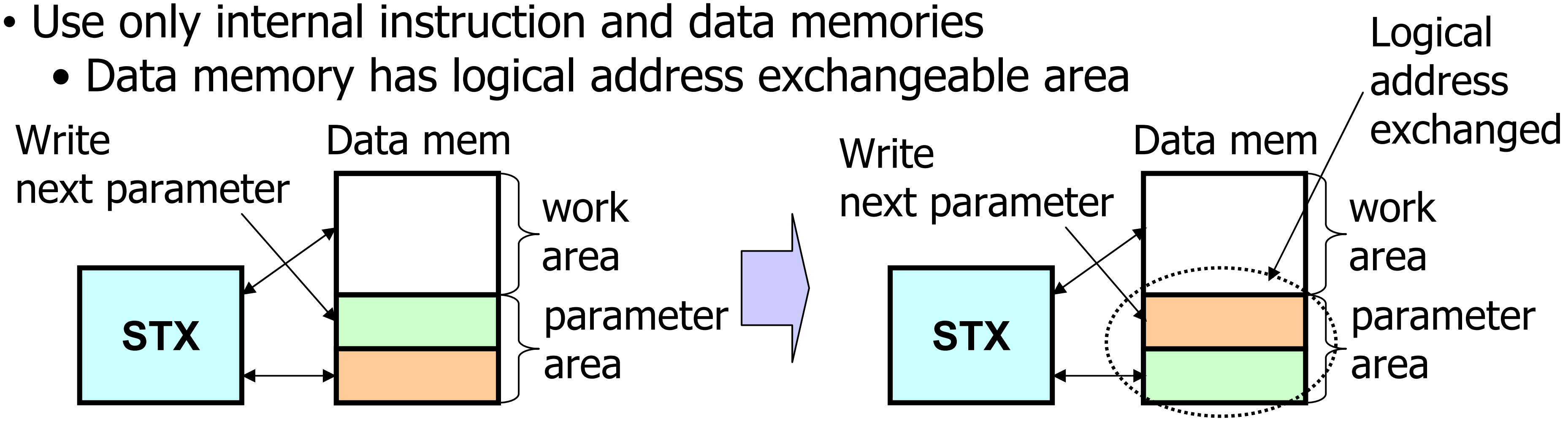


Inst. slot A	Inst. slot B
• register data transfer	• register data transfer
• load/store	• arithmetic operation
• stream I/O	• branch
• accelerator control	

2 instruction slots used rate

Stream Type	Rate
H.264 CAVLC	32%
H.264 CABAC	38%
MPEG-2	48%
MPEG-4	45%
VC-1	46%

- Use only internal instruction and data memories
  - Data memory has logical address exchangeable area



Pixel-domain operation cycle budgeting

Required operation amount for MB is not so different

➡ Assign operation cycle budget for a macroblock

Full-HD (1920×1080 30fps) video MB rate : 244,800 MB/s
Target operation frequency : 162MHz

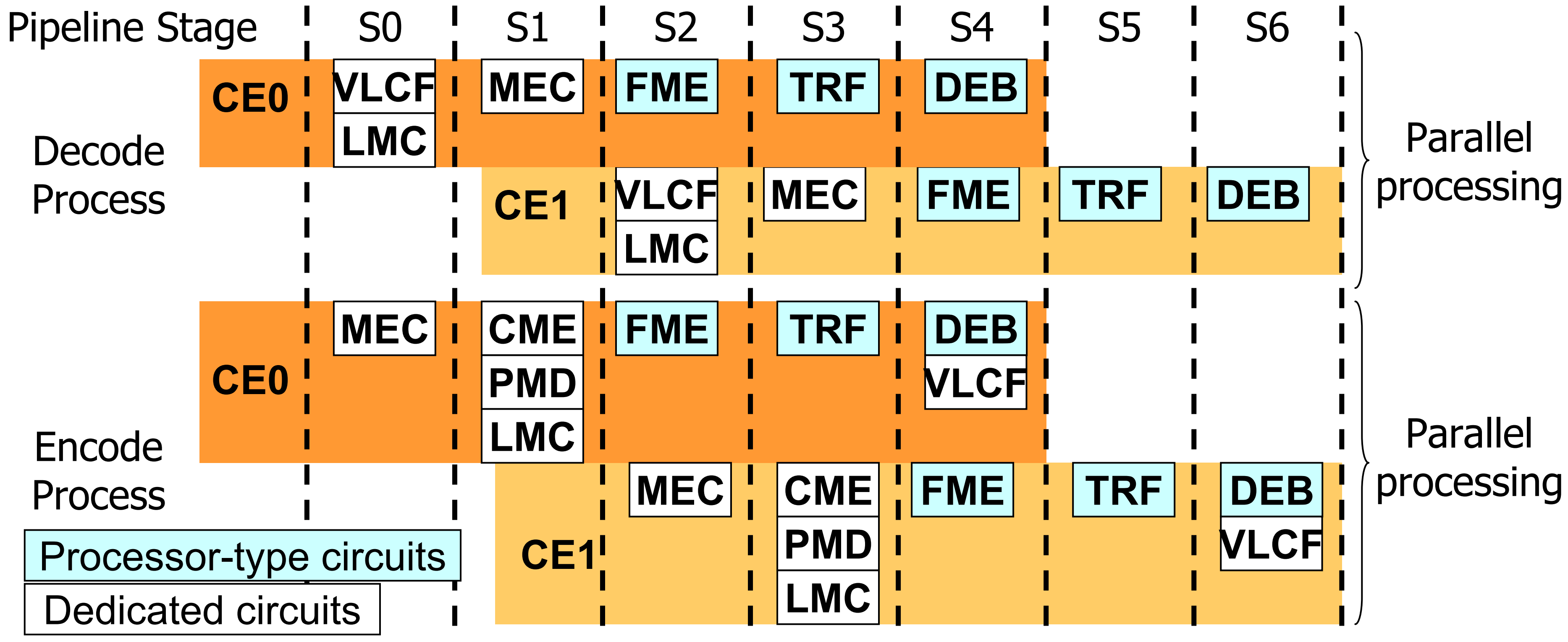
➡ Only 661 cycle is available for a MB processing pipeline stage

⬇ Too strict for processor based operation
(A MB has 384 pixels for luma & chroma)

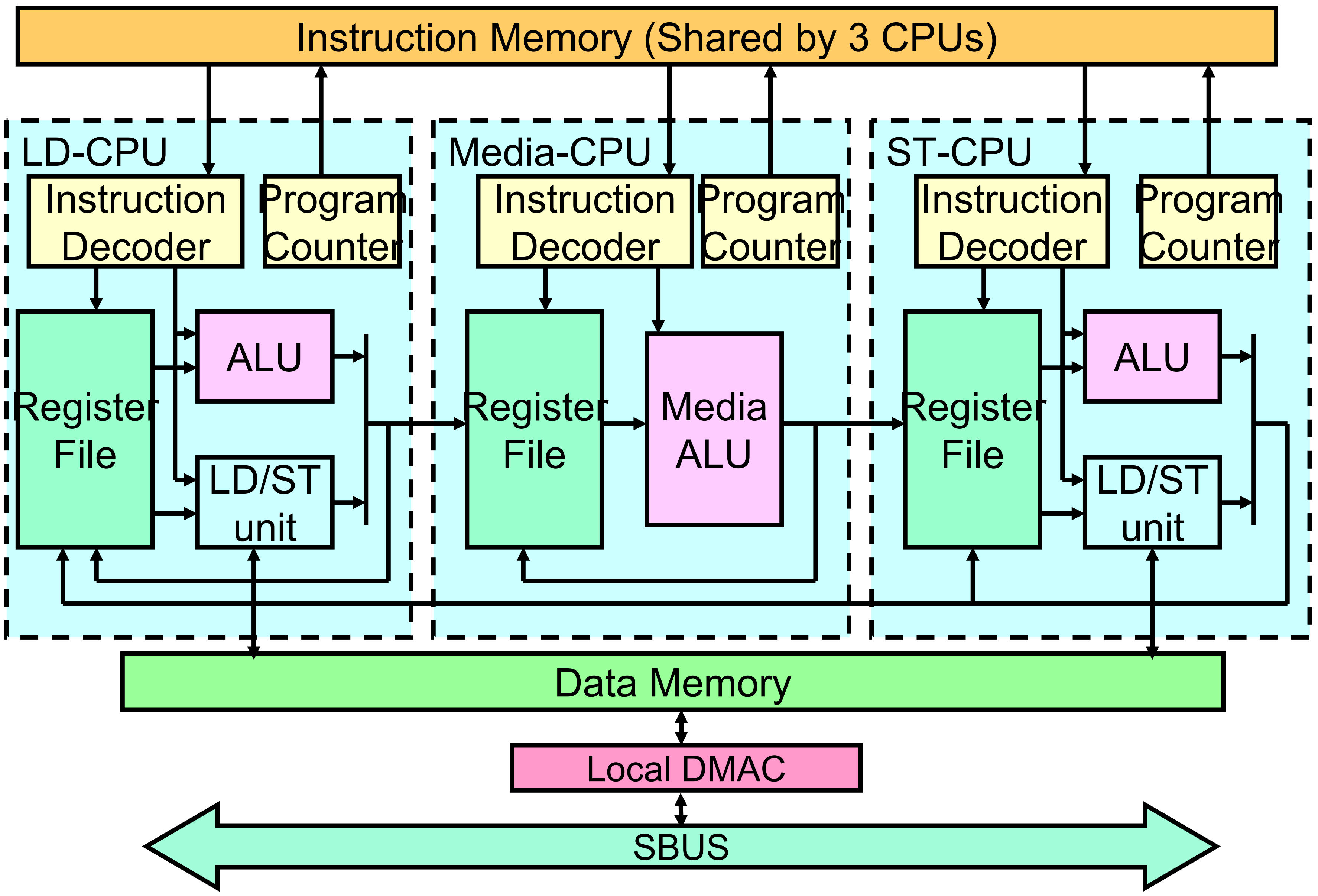
Assign 661×2 = 1,332 cycles by **2 parallel processing**
(**1,200** cycle for actual operation, 132 cycle for margin)

Hierarchical parallel processing

- Pixel domain uses hierarchical parallel processing technique
  1. 2 MBs processed 2 codec elements (CEs) in parallel
  2. Each MB is processed by “pipeline” technique: each module is assigned as an pipeline stage.
  3. Parallel processing is executed in each module: processor type modules have some tiny processor elements.

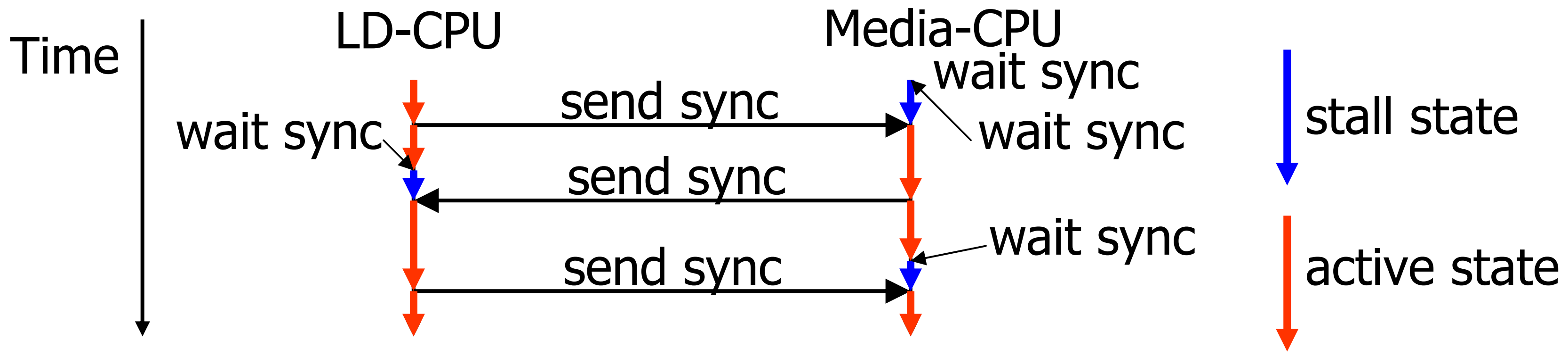


Pixel domain processor (Programmable Image Processing Element: PIPE)

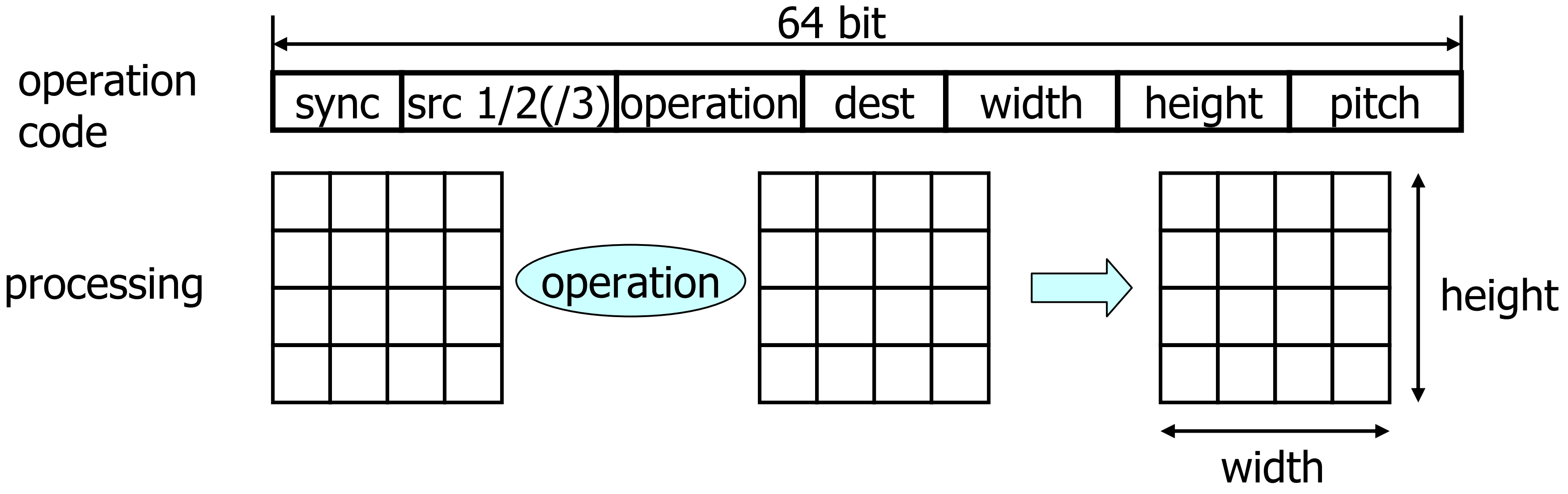


PIPE based on MIAD architecture (MIAD: Multiple Instruction Arrayed Data)

- LD-CPU, Media-CPU, and ST-CPU have own program counter
  Those CPUs synchronize each other by sync flags in operation code



- Those CPUs take 2 dimensional arrayed data operands

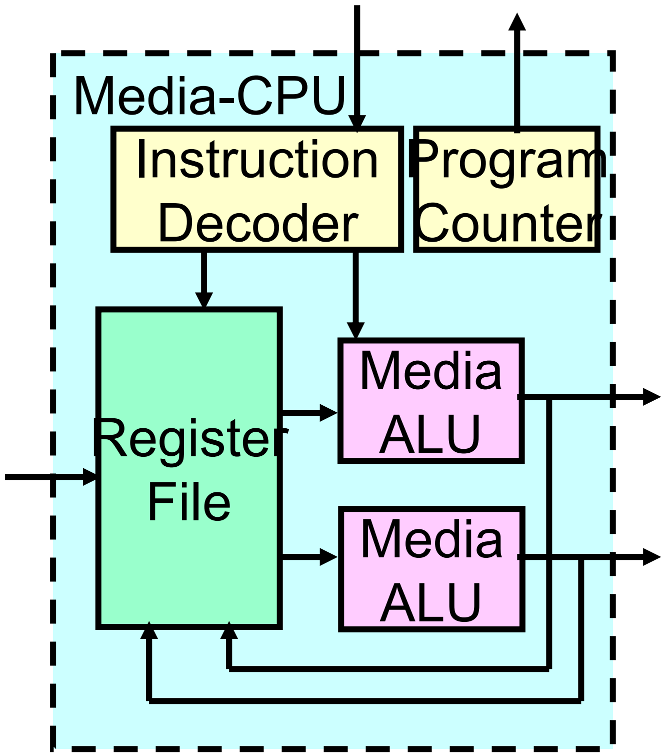


PIPE extension

PIPE instruction set is extended for each module

- Major extensions are added to Media-CPU
- Some data setup operation extensions are added to LD/ST-CPU

Module name (Main function)	Major extensions
FME (Fine Motion Estimation/ Compensation)	•2way LIW mode •Fine motion estimation/compensation specific instructions
TRF (Transform and Quantization)	•2way LIW mode •Transform and quantization specific instructions
DEB (De-blocking filter)	•De-blocking specific instructions



Media-CPU with 2way LIW extension

Hybrid architecture

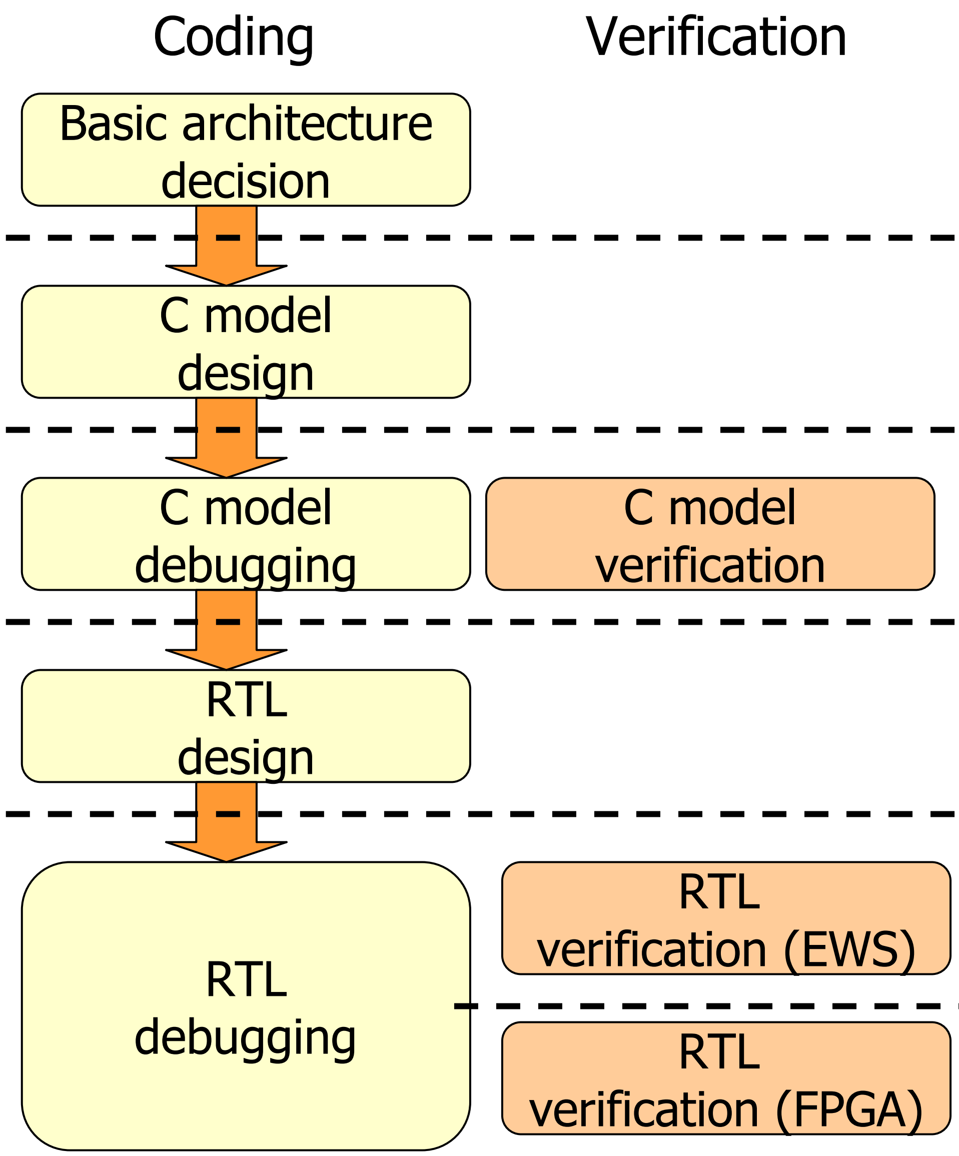
CE works by combination of PIPEs and dedicated circuits

- PIPE architecture is optimized for 2D arrayed pixel processing
- Dedicated circuits used for the functions PIPE is inefficient for

Modules implemented by dedicated circuits in pixel-domain

Module name	Main functions	Reasons to use dedicated circuits
VLCF	• decode/encode intermediate stream • MV calculation	• PIPE is inefficient
PMD	• intra prediction mode selection (used by H.264 encode process only)	• logic size
LMC	• internal line buffer control	• PIPE is inefficient
CME	• coarse motion estimation and compensation	• performance
MEC	• frame buffer access control for CME operations	• PIPE is inefficient

Design flow



- Decide modules in top level (functions & interfaces)

- Design C-language-based model corresponded to the modules
- Develop firmware for processors

- Compare with reference code results
- Check performance roughly

- Design RTL corresponded to the modules (refer C model for detail)

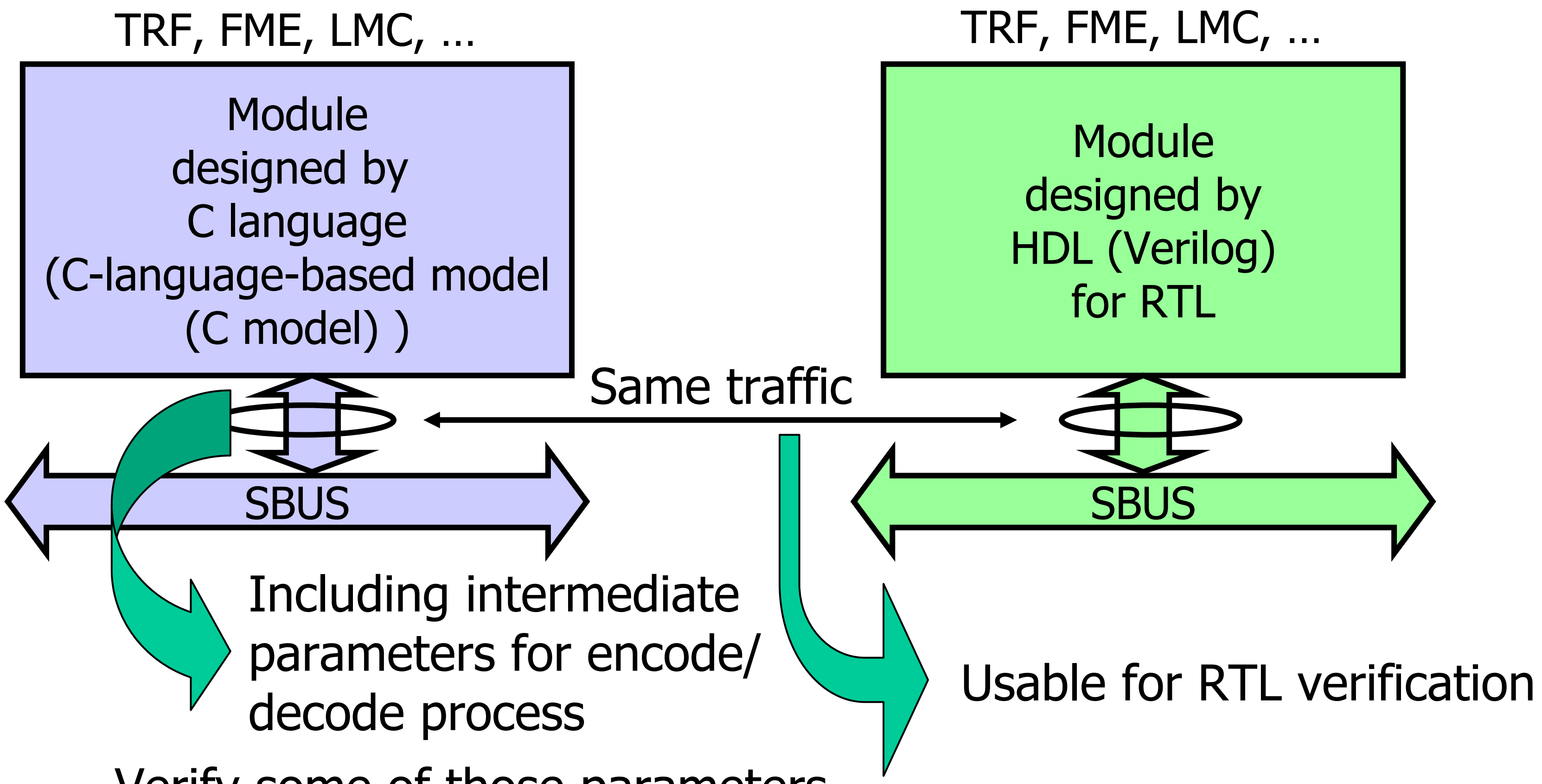
- Check function using C model
- Check performance/coverage /assertions

- Detail verification using many long streams

C-language-based model design

All modules are connected to SBUS

➡ The SBUS traffic of C model is designed to be the same as RTL


TRF, FME, LMC, …
Module designed by C language (C-language-based model (C model))
TRF, FME, LMC, …
Module designed by HDL (Verilog) for RTL
Same traffic
SBUS
SBUS

Including intermediate parameters for encode/ decode process

Usable for RTL verification

Verify some of those parameters using codec reference code

Firmware development

- Processors (STX and PIPE) designed in C-language-based model
  - Processor models in C model can take binary codes
  - Cycle accurate processor models

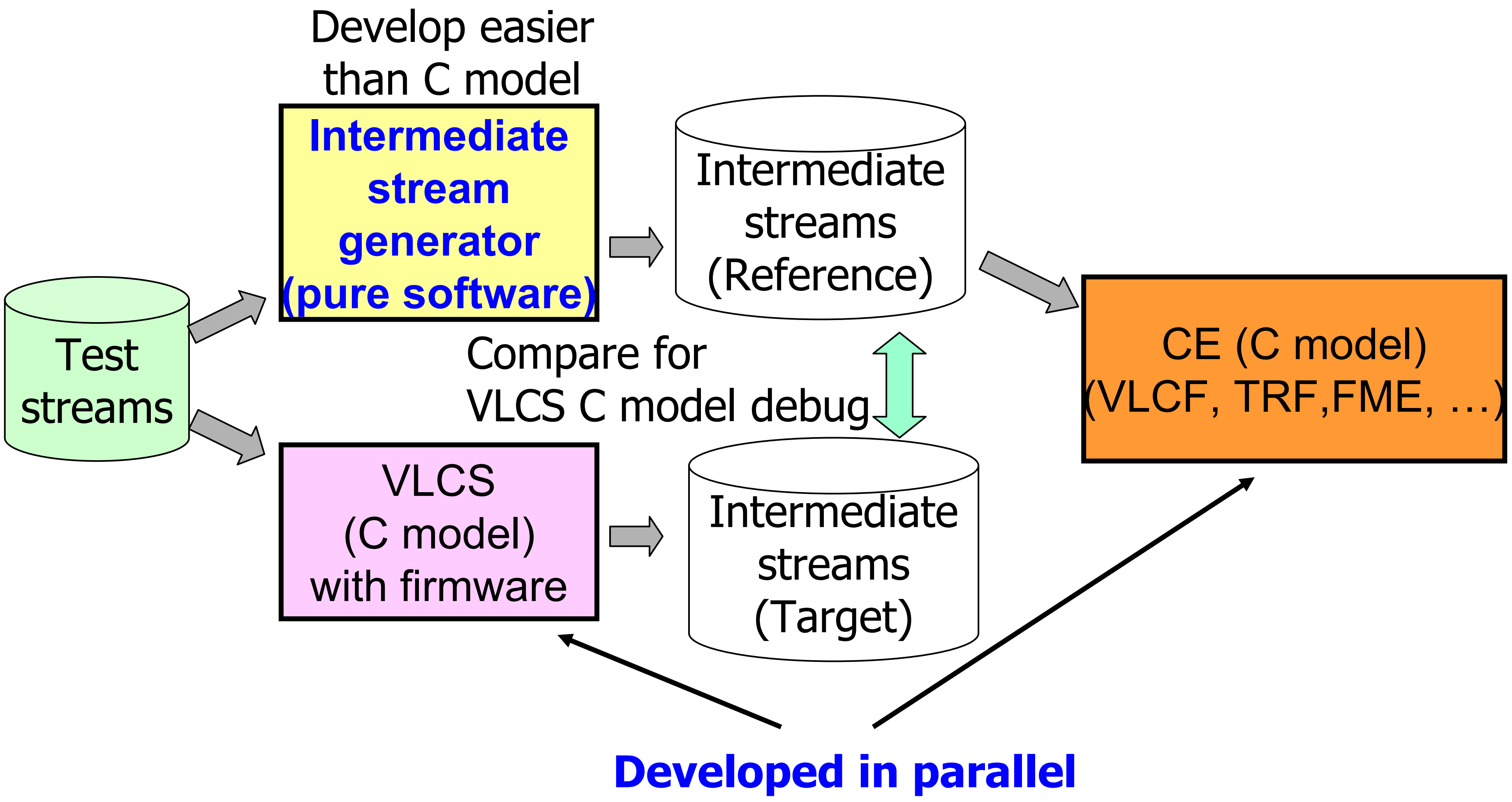
  - Firmware developed as a part of C model
  - Rough performance evaluated in C model design
  - Revise architecture if any problems found

- Firmware developed using assembler

  Because…
  - Small firmware code size
  - Save time to develop high level language tools

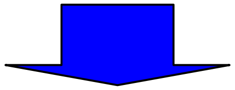
Concurrent C model development

- Intermediate stream generator was developed for concurrent design

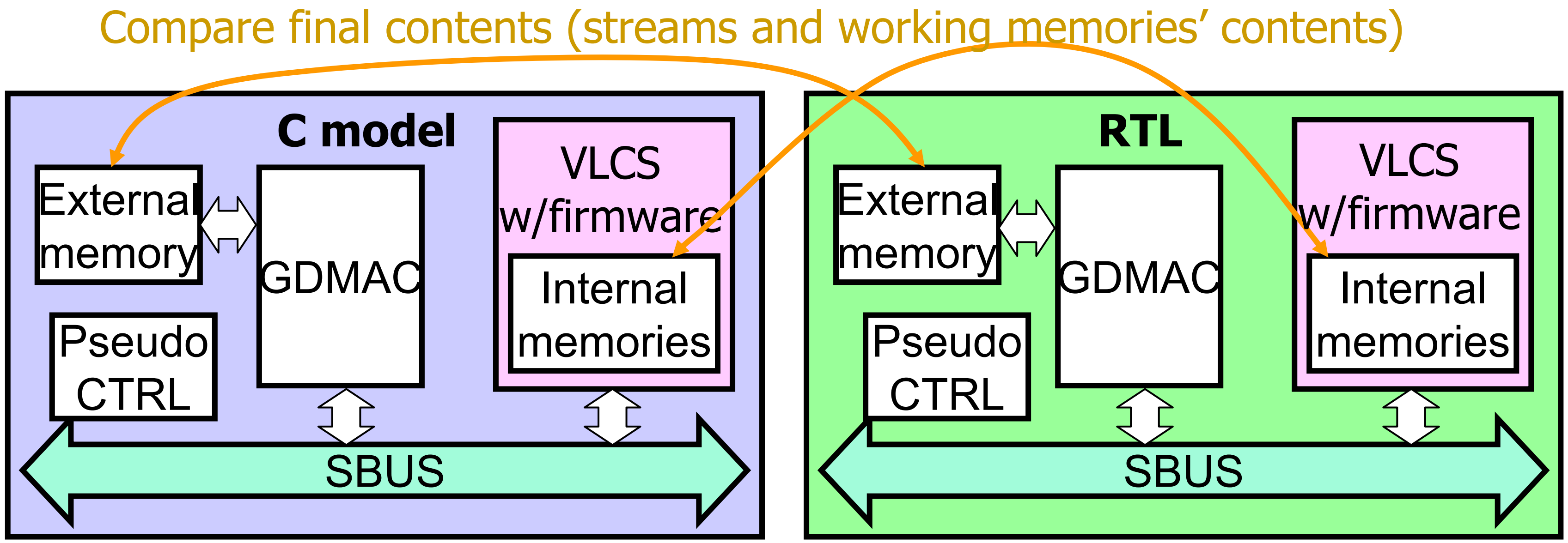

Develop easier than C model
Intermediate stream generator (pure software)
Intermediate streams (Reference)
Test streams
Compare for VLCS C model debug
CE (C model) (VLCF, TRF,FME, …)
VLCS (C model) with firmware
Intermediate streams (Target)
Developed in parallel

VLCS RTL verification

- Difficult to make the same traffic between C and RTL for VLCS
  - Plural streams transferred by local DMAC (LDMAC)
    (Impossible to predict the stream data transfer order)
  - VLCS works tightly with global DMAC (GDMAC) for stream handling
    (GDMAC model required as test environment)



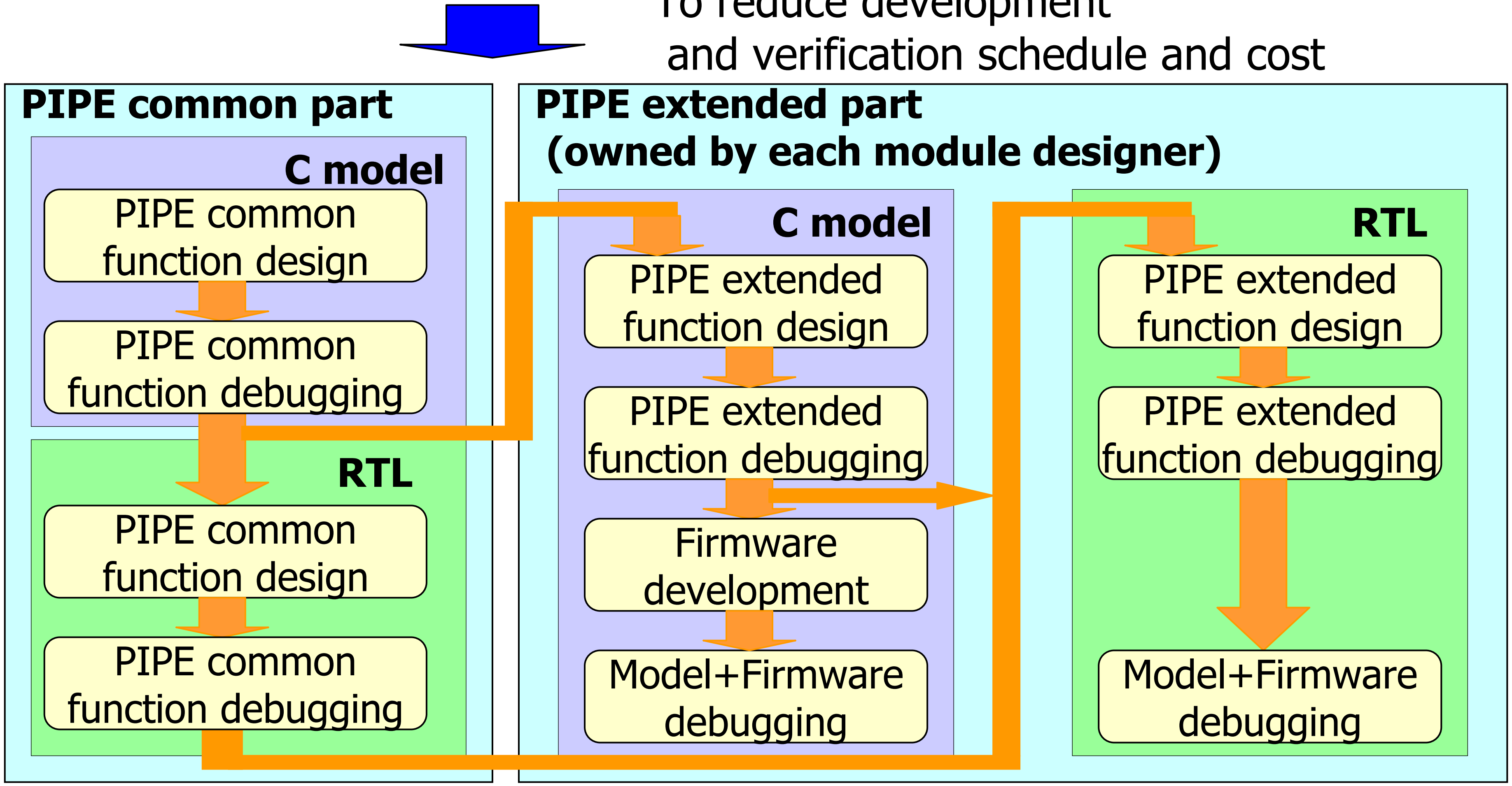
- Verify final result values in internal and external memories
- Use real GDMAC model for test environment



PIPE based module RTL design & verification

•PIPE is a common processor
•PIPE is extended for each module

To reduce development
 and verification schedule and cost



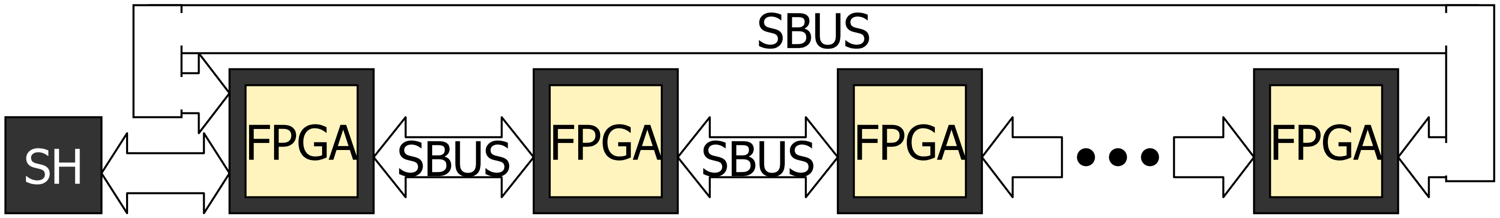
PIPE common part
C model
PIPE common function design
PIPE common function debugging
RTL
PIPE common function design
PIPE common function debugging
PIPE extended part
(owned by each module designer)
C model
PIPE extended function design
PIPE extended function debugging
Firmware development
Model+Firmware debugging
RTL
PIPE extended function design
PIPE extended function debugging
Model+Firmware debugging

Verification using FPGA

- FPGA used for a detailed verification

  How implement large IP on FPGA

  - Allocate to 9 FPGAs (Xilinx VERTEX-4 XC4VLX200)
  - Connect FPGAs using SBUS
  - Verify encoder mode and decoder mode separately
    (Remove unnecessary logic for each mode)



  What bugs found by FPGA verification

  - Stall control
  - Interrupt control
  - Synchronization between processors
  - Error stream handling
  - Corner cases (Need to verify with many video streams)

Adding codec standard support

- Codec standards added step by step
  - IP basic architecture expects for adding codec standards support

    But supporting one codec standards requires much works…

3 phases for IP development

- first phase
  - Designed basic architecture for multi codec support
  - Designed detail logic for H.264/MPEG-4 AVC wo/MBAFF (decode/encode)
- second phase
  - Supported for MPEG-2 and MPEG-4 (decode/encode)
  - Optimized PIPE micro architecture for logic size compaction
- third phase
  - Supported for H.264/MPEG-4 AVC MBAFF
  - Supported for VC-1 (decode only)

For codec support extension, firmware and additional RTL are developed

Developed CODEC IP

- IP developed dividing to 3 phases
- The 3rd phase IP development has been completed

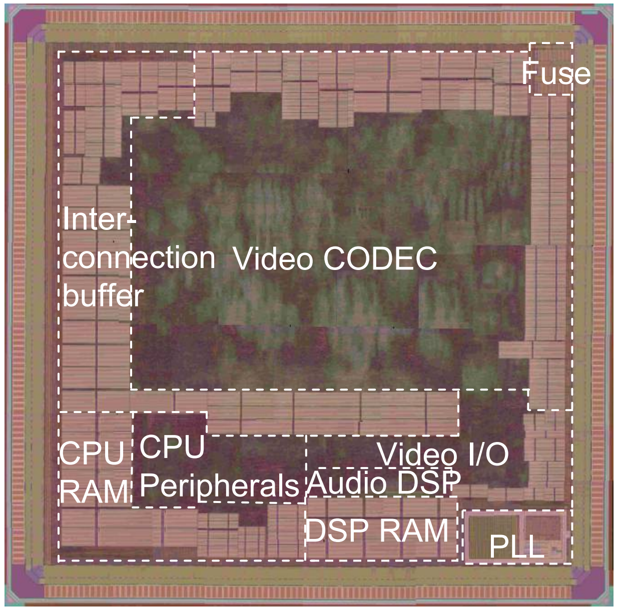
Development Phase	Phase 1	Phase 2	Phase 3
VLCS Logic [Relative logic size]	240kG [1.00]	289kG [1.20]	337kG [1.40]
PIPE-Based Logic (Sum of all PIPE based modules in the CODEC IP) [Relative logic size]	2694kG [1.00]	2475kG (*1) [0.92]	2712kG [1.01]
Supported Codec Standard	H.264/ MPEG-4 AVC (w/o MBAFF)	H.264/ MPEG-4 AVC (w/o MBAFF) MPEG-2 MPEG-4	H.264/ MPEG-4 AVC (w/ MBAFF) MPEG-2 MPEG-4 VC-1(decode only)

(*1) Smaller than phase 1 because of PIPE micro architecture optimization

Sample implementation results on a chip

- The 1st phase IP has been implemented in the test chip

Technology	65 nm, 7-layer, Cu, CMOS
Supply Voltage	1.2 V (Internal) 1.8 V (I/O)
Clock Frequency	162 MHz (Internal) 324 MHz (DDR-SDRAM I/O)
Supported Codec Standard	H.264/MPEG-4 AVC (w/o MBAFF) High profile level 4.1
Performance	1920x1080 30 fps 40 Mbps (CABAC)
CODEC Logic	3745 kG
CODEC Internal Memory	228 kB
Measured Power Consumption (excluding I/O)	Encoding: 256 mW Decoding: 172 mW (both for full-HD case)



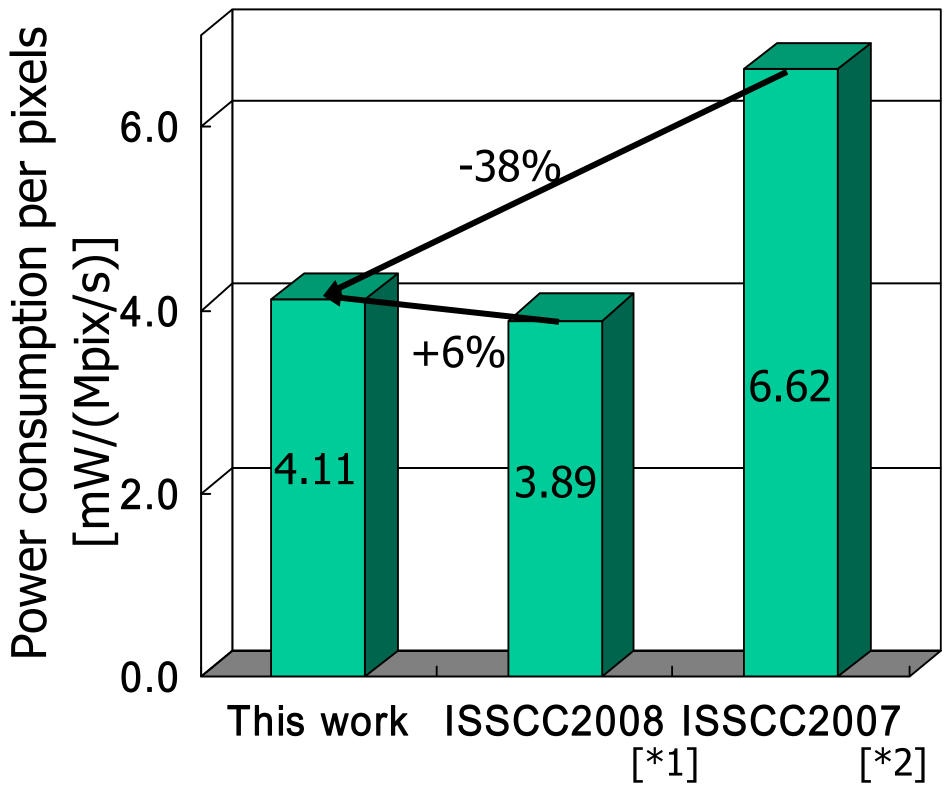
Micrograph of the test chip(*1)
© 2008 IEEE

(*1) K.Iwata, et al. "A 256mW Full-HD H.264 High-Profile CODEC Featuring Dual Macroblock-Pipeline Architecture in 65nm CMOS," 2008 Symposium on VLSI Circuits Digest of Technical Papers, pp.102-103

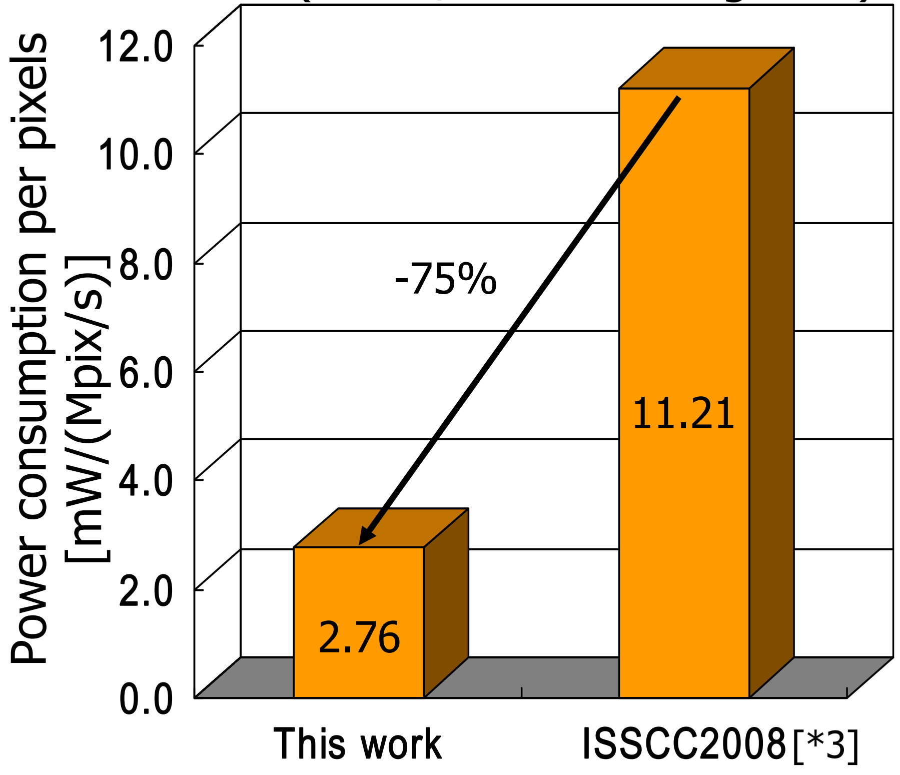
Design comparison

Comparison with other state-of-art designs

Compared with H.264/AVC specific dedicated-circuits-based design (H.264/AVC encoding case)



Compared with processor-based design (H.264/AVC decoding case)



[*1] Y.K. Lin, et al., "A 242mW 10mm2 1080p H.264/AVC High-Profile Encoder Chip," session 16.5, ISSCC 2008

[*2] H.C Chang, et al., "A 7mW-to-183mW Dynamic Quality-Scalable H.264 Video Encoder Chip," session 15.6, ISSCC 2007

[*3] S. Nomura, et al., "A 9.7mW AAC-Decoding, 620mW H.264 720p 60fps Decoding, 8-Core Media Processor with Embedded Forward-Body-Biasing and Power-Gating Circuit in 65nm CMOS Technology," session 13.4, ISSCC 2008

Summary and Conclusions

1. A multi-standard video CODEC IP has been developed.

2. The IP can handle full-HD (1920×1080 30fps) video at 162MHz for MPEG-2/4, H.264 for decode/encode. VC-1 is supported for decode.

3. The IP takes heterogeneous multiprocessor architecture; uses 2 kinds of processors, STX and PIPE, and PIPE was extended for each module.

4. A test chip developed with 1st phase IP; The CODEC works only 256mW for full-HD H.264 encode and 172mW for decode. This power consumption is very low though we used processors for flexibility.

Acknowledgement

- Thank you for all persons who gave me this presentation opportunities.
- I want to say to all of you

Thank you