TOSHIBA
Leading Innovation >>>

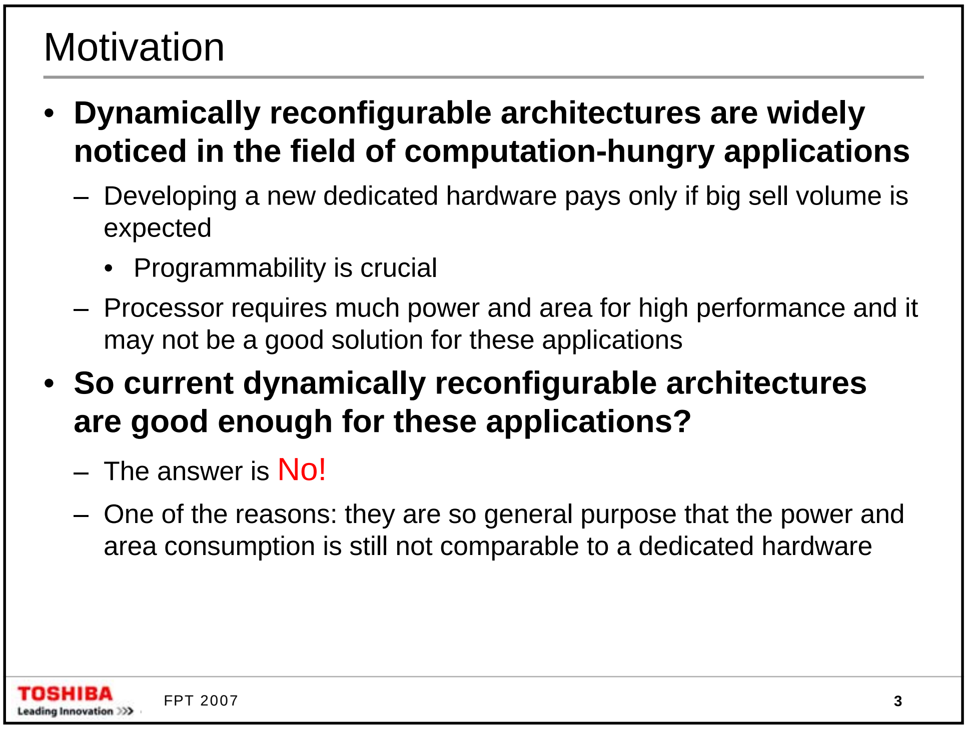
A Dynamically Reconfigurable Architecture for Stream Processing

Takashi Yoshikawa
Corporate Research & Development Center, Toshiba Corporation

FPT 2007.

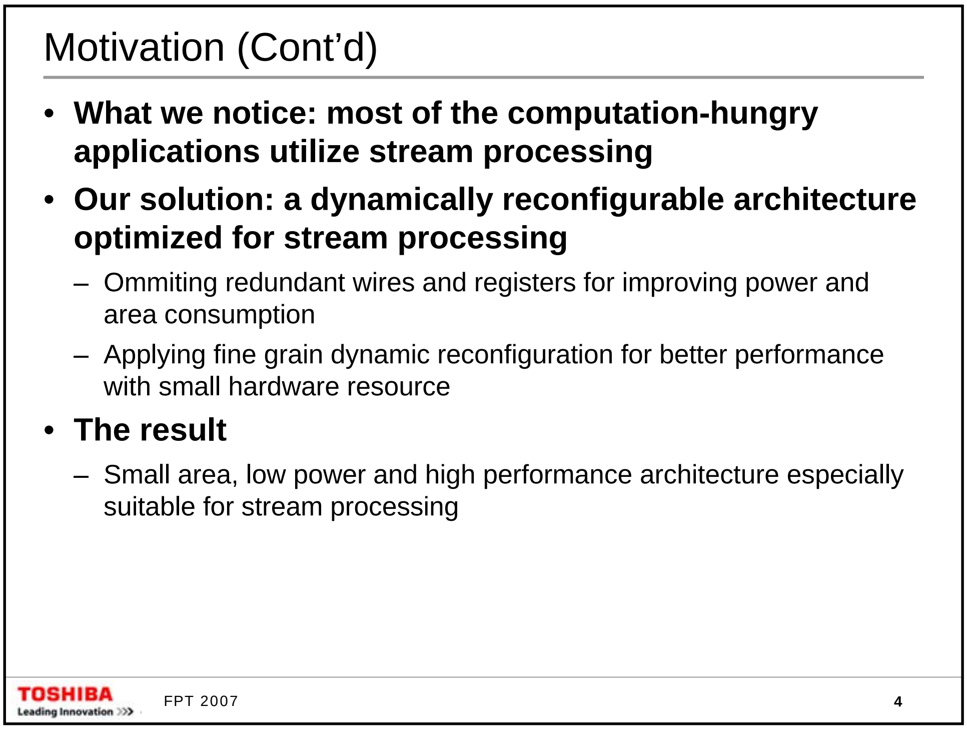
Agenda

1. **Motivation**
2. **The Architecture**
3. **Code Development Environment**
4. **Evaluation**
5. **Conclusion**



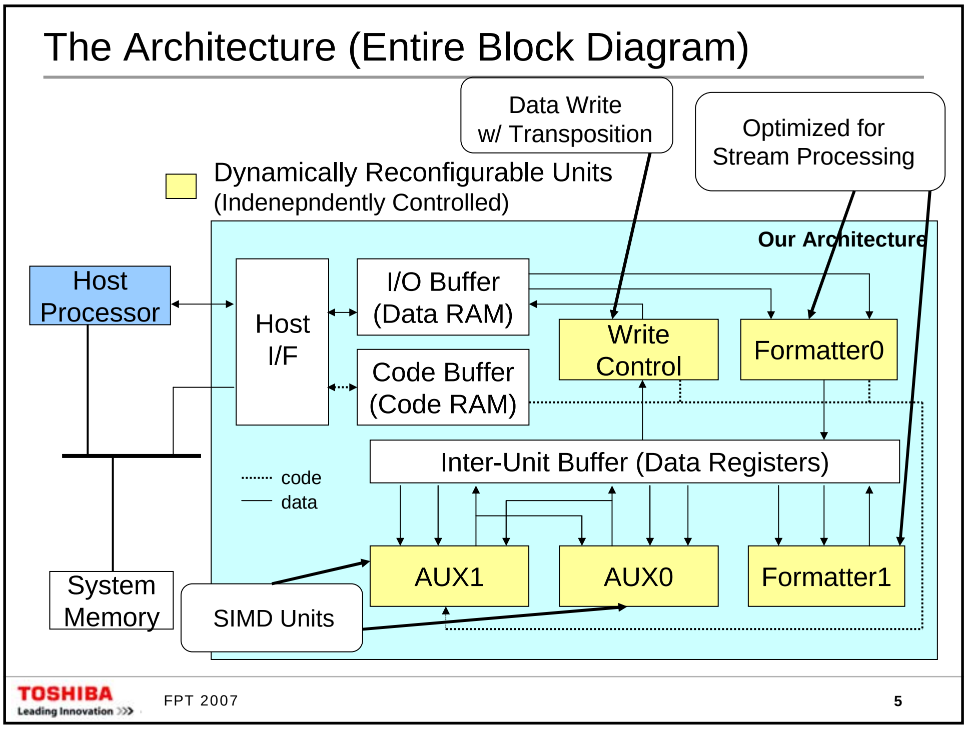
Motivation

- **Dynamically reconfigurable architectures are widely noticed in the field of computation-hungry applications**
  - Developing a new dedicated hardware pays only if big sell volume is expected
    - Programmability is crucial
  - Processor requires much power and area for high performance and it may not be a good solution for these applications
- **So current dynamically reconfigurable architectures are good enough for these applications?**
  - The answer is No!
  - One of the reasons: they are so general purpose that the power and area consumption is still not comparable to a dedicated hardware

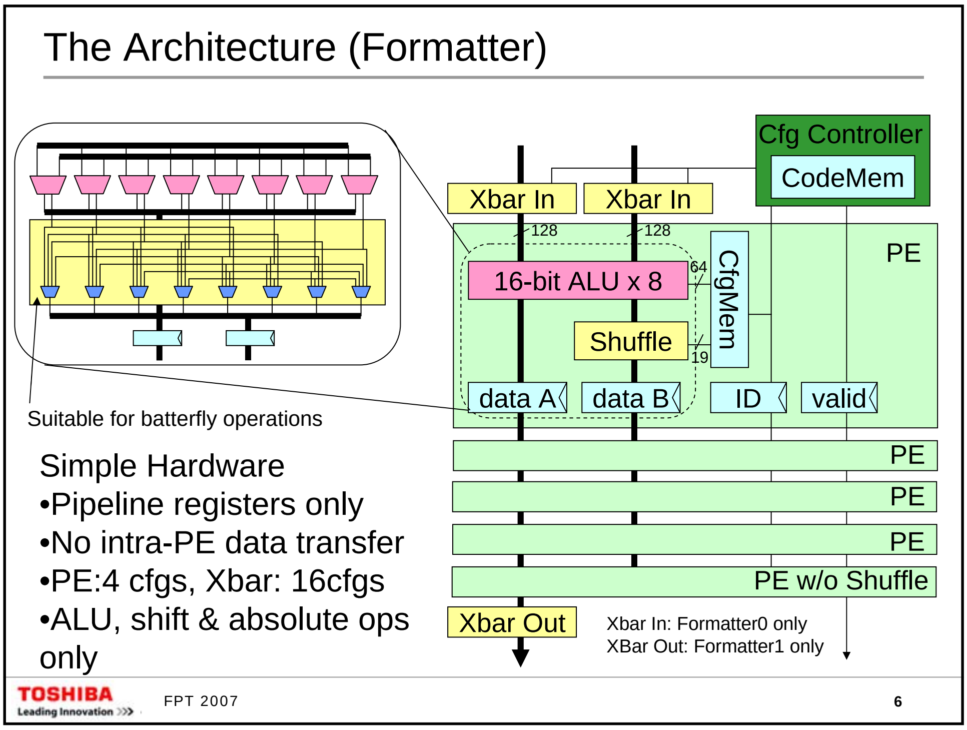


Motivation (Cont'd)

- **What we notice: most of the computation-hungry applications utilize stream processing**
- **Our solution: a dynamically reconfigurable architecture optimized for stream processing**
  - Ommiting redundant wires and registers for improving power and area consumption
  - Applying fine grain dynamic reconfiguration for better performance with small hardware resource
- **The result**
  - Small area, low power and high performance architecture especially suitable for stream processing



The Architecture (Entire Block Diagram)



The Architecture (Formatter)

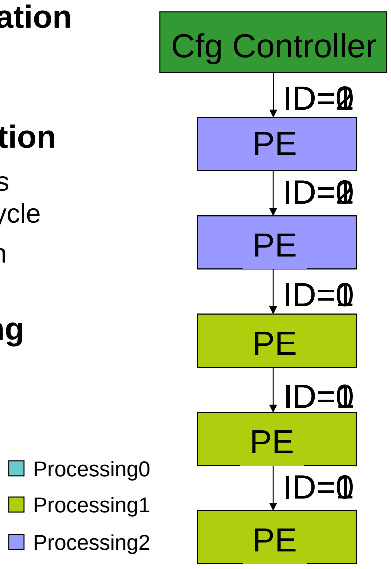
Suitable for batterfly operations

Xbar In: Formatter0 only
XBar Out: Formatter1 only

Simple Hardware
•Pipeline registers only
•No intra-PE data transfer
•PE:4 cfgs, Xbar: 16cfgs
•ALU, shift & absolute ops only

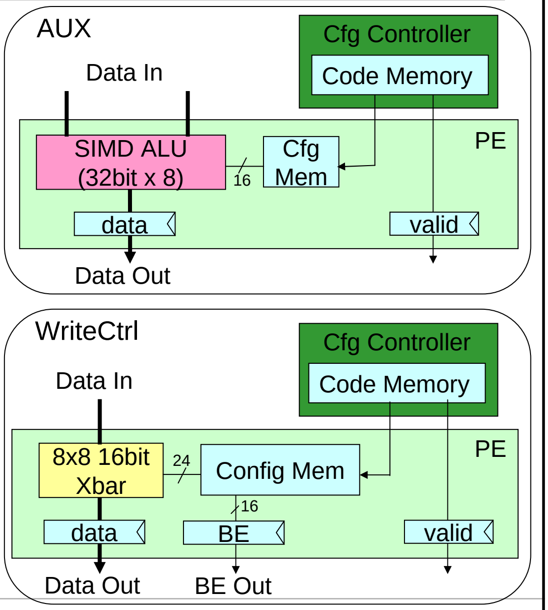
The Architecture (Formatter – Cont'd)

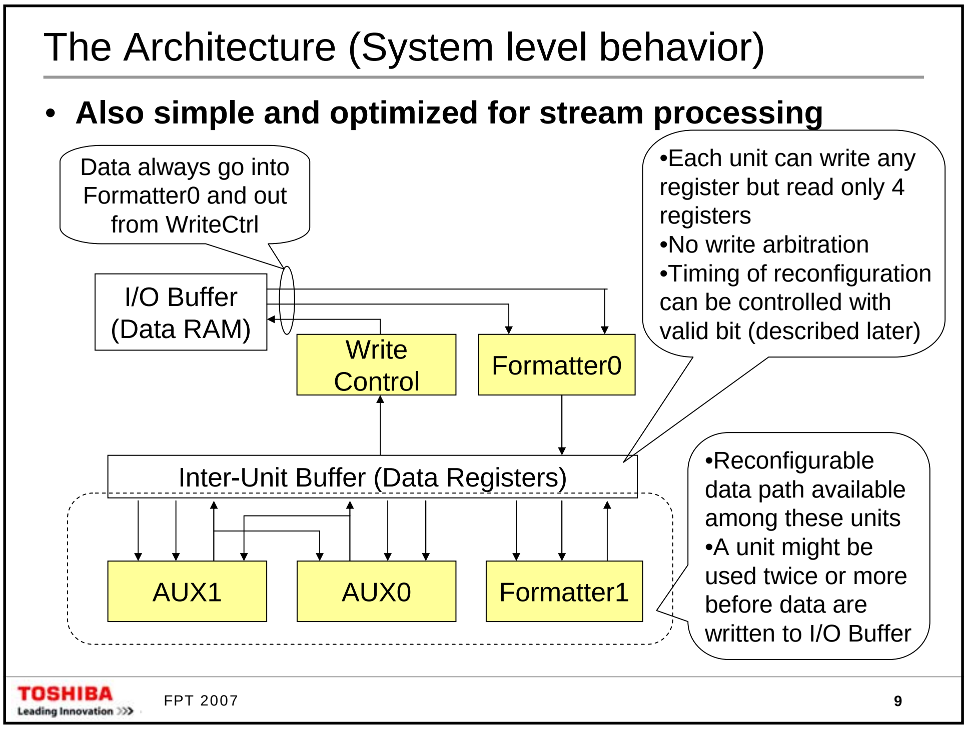
- **No conditional reconfiguration**
  - Reconfiguration is statically scheduled
- **Pipeline-style reconfiguration**
  - Cfg ID from Cfg Controller goes through PE pipeline cycle by cycle
  - Each PE alters its configuration when it receives a new ID
- **Ideal for stream processing**
  - Enables 2 or more stream processing at the same time



The Architecture (AUX and WriteCtrl)

- **AUX**
  - SIMD-style reconfiguration
  - 1 PE only
  - Bypass data path available between AUXs
  - 32bit ops, multiply, clip, compare and select ops
  - Up to 32 cfgs
- **WriteCtrl**
  - Write data to I/O Buffer
  - Support data transposition
  - Support byte enable
  - Xbar: 16 cfgs, BE: 32cfgs





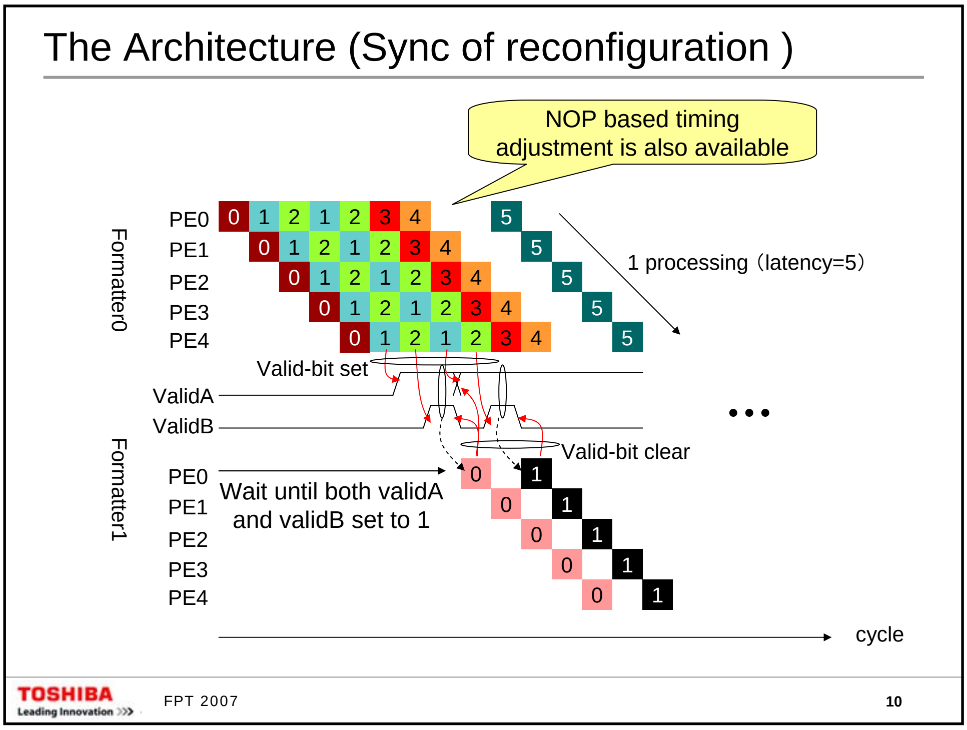
The Architecture (System level behavior)

- **Also simple and optimized for stream processing**

Data always go into Formatter0 and out from WriteCtrl

- Each unit can write any register but read only 4 registers
- No write arbitration
- Timing of reconfiguration can be controlled with valid bit (described later)

- Reconfigurable data path available among these units
- A unit might be used twice or more before data are written to I/O Buffer



The Architecture (Sync of reconfiguration)

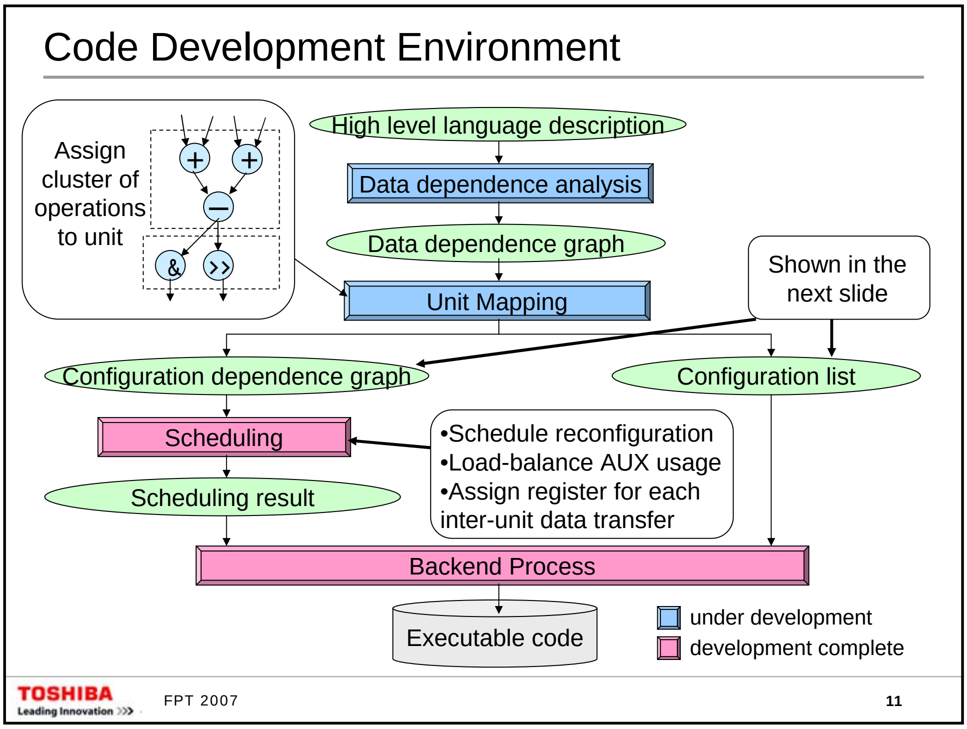
NOP based timing adjustment is also available

1 processing（latency=5）

Valid-bit set

Valid-bit clear

Wait until both validA and validB set to 1



Code Development Environment

Assign cluster of operations to unit

High level language description

Data dependence analysis

Data dependence graph

Unit Mapping

Shown in the next slide

Configuration dependence graph

Configuration list

Scheduling

- •Schedule reconfiguration
- •Load-balance AUX usage
- •Assign register for each inter-unit data transfer

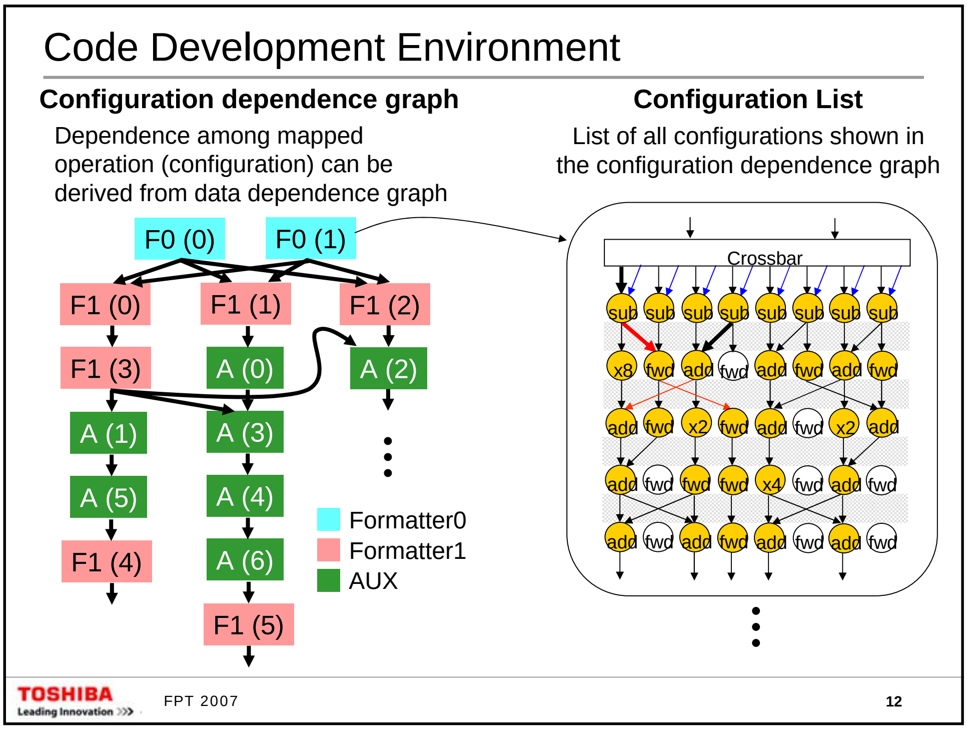
Scheduling result

Backend Process

Executable code

■ under development

■ development complete



Code Development Environment

Configuration dependence graph

Dependence among mapped operation (configuration) can be derived from data dependence graph

Configuration List

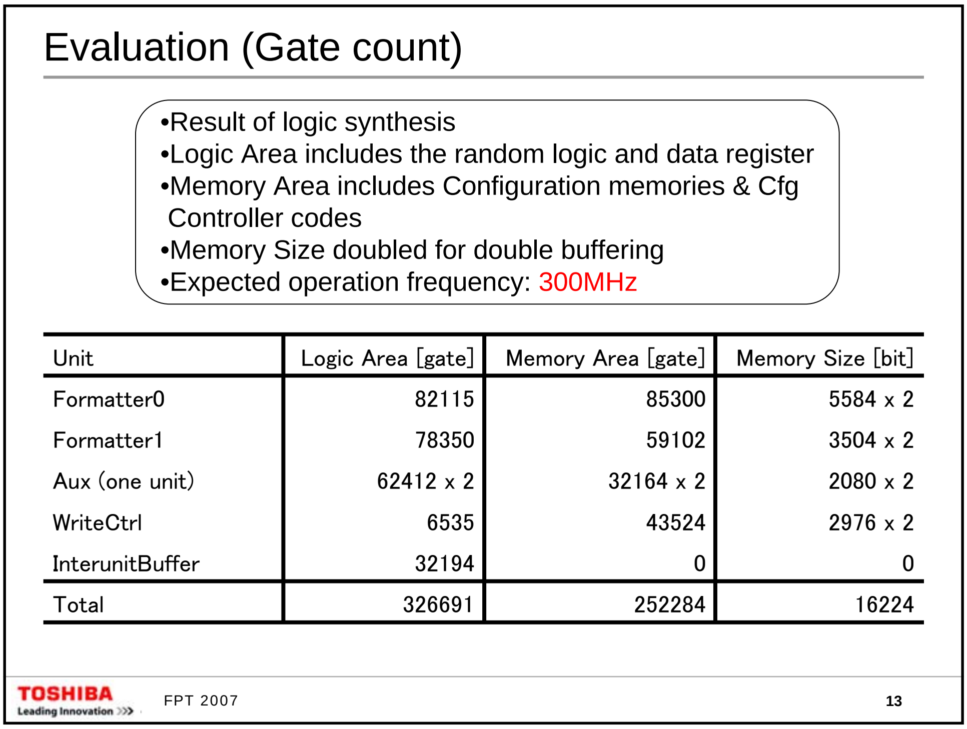
List of all configurations shown in the configuration dependence graph

F0 (0)
F0 (1)
F1 (0)
F1 (1)
F1 (2)
F1 (3)
A (0)
A (2)
A (1)
A (3)
A (5)
A (4)
F1 (4)
A (6)
F1 (5)
Crossbar
sub sub sub sub sub sub sub sub
x8 fwd add fwd add fwd add fwd
add fwd x2 fwd add fwd x2 add
add fwd fwd fwd x4 fwd add fwd
add fwd add fwd add fwd add fwd

■ Formatter0

■ Formatter1

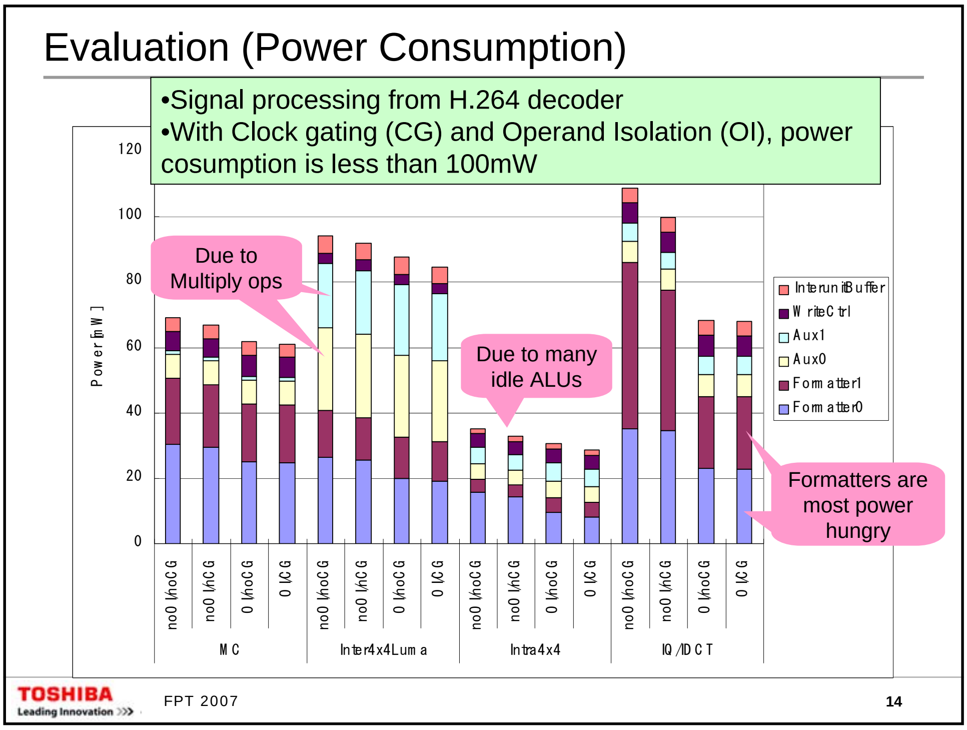
■ AUX



Evaluation (Gate count)

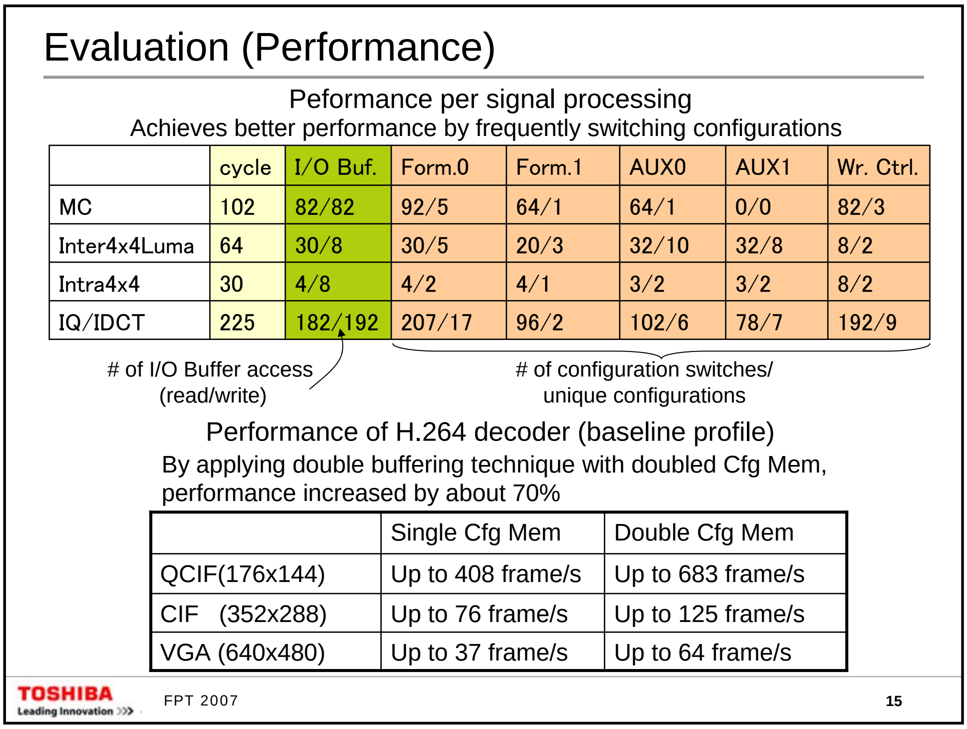
- •Result of logic synthesis
- •Logic Area includes the random logic and data register
- •Memory Area includes Configuration memories & Cfg Controller codes
- •Memory Size doubled for double buffering
- •Expected operation frequency: 300MHz

Unit	Logic Area [gate]	Memory Area [gate]	Memory Size [bit]
Formatter0	82115	85300	5584 x 2
Formatter1	78350	59102	3504 x 2
Aux (one unit)	62412 x 2	32164 x 2	2080 x 2
WriteCtrl	6535	43524	2976 x 2
InterunitBuffer	32194	0	0
Total	326691	252284	16224



Evaluation (Power Consumption)

- •Signal processing from H.264 decoder
- •With Clock gating (CG) and Operand Isolation (OI), power cosumption is less than 100mW



Evaluation (Performance)

Peformance per signal processing

Achieves better performance by frequently switching configurations

	cycle	I/O Buf.	Form.0	Form.1	AUX0	AUX1	Wr. Ctrl.
MC	102	82/82	92/5	64/1	64/1	0/0	82/3
Inter4x4Luma	64	30/8	30/5	20/3	32/10	32/8	8/2
Intra4x4	30	4/8	4/2	4/1	3/2	3/2	8/2
IQ/IDCT	225	182/192	207/17	96/2	102/6	78/7	192/9

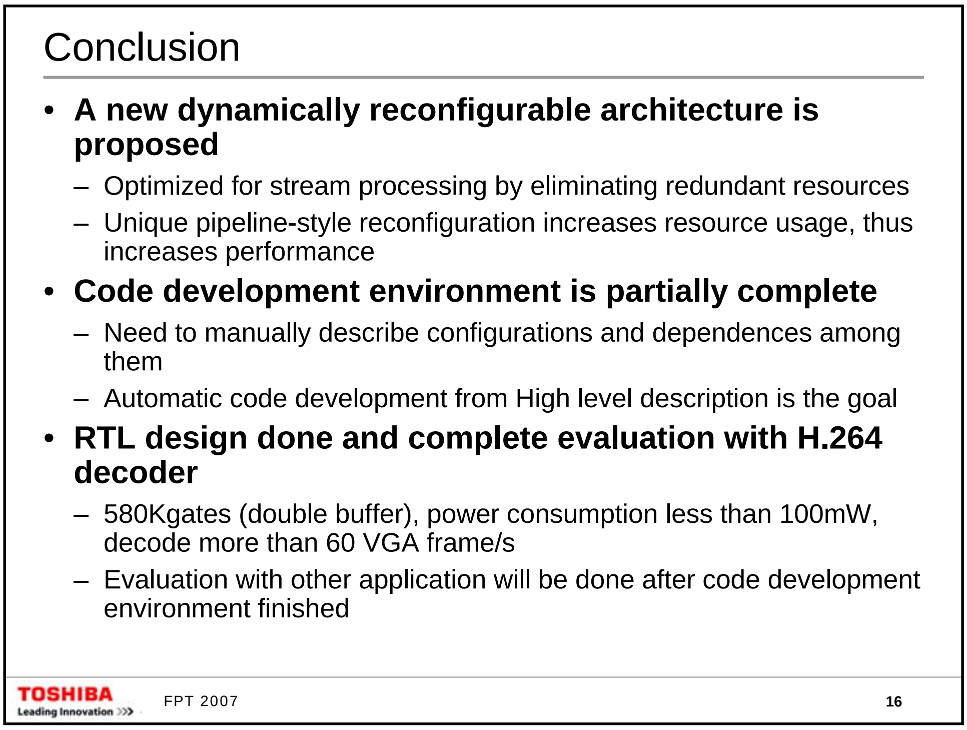
# of I/O Buffer access (read/write)

# of configuration switches/ unique configurations

Performance of H.264 decoder (baseline profile)

By applying double buffering technique with doubled Cfg Mem, performance increased by about 70%

	Single Cfg Mem	Double Cfg Mem
QCIF(176x144)	Up to 408 frame/s	Up to 683 frame/s
CIF (352x288)	Up to 76 frame/s	Up to 125 frame/s
VGA (640x480)	Up to 37 frame/s	Up to 64 frame/s



Conclusion

- **A new dynamically reconfigurable architecture is proposed**
  - Optimized for stream processing by eliminating redundant resources
  - Unique pipeline-style reconfiguration increases resource usage, thus increases performance
- **Code development environment is partially complete**
  - Need to manually describe configurations and dependences among them
  - Automatic code development from High level description is the goal
- **RTL design done and complete evaluation with H.264 decoder**
  - 580Kgates (double buffer), power consumption less than 100mW, decode more than 60 VGA frame/s
  - Evaluation with other application will be done after code development environment finished