Challenges on Designing Parallel Processing for Realizing Real-Time Applications

July 12th, 2019

Architect, President & CEO

Yukoh Matsumoto, Ph.D.

TOPS Systems Corp.

Agenda

- **Development Services / TOPS Systems**
  - Efficient Parallel Processing Software and Hardware
- **Demand of Parallel Processing in Real-Time Systems**
  - ML, NW, 5G, Brake System, Free 3D View, Robotics Control
- **Challenges on Designing Parallel Processing**
  - Free Lunch is over in 2005, but …
  - Many Performance Black Boxes in SW and HW
- **Methodologies : Designing Time and Parallelism**
  - Models of Processing
  - White Box Performance based Parallel Processing Design
- **Architecture : Removing Inter-Core Overhead**
  - Communication and Synchronization
- **Solutions**

My principle of Life

When we do our best, Ideas come up.

When we do incomplete, Complains come up.

When we are lazy, Excuses come up.

Takeda Shingen (1521-1573)

In Japanese,

一生懸命だと、智慧がでる

中途半端だと、愚痴がでる

いい加減だと、言い訳がでる

武田 信玄

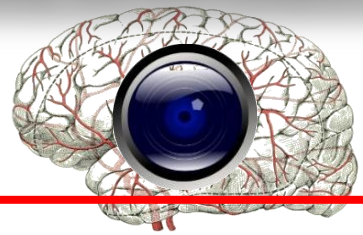
When we do our best, Ideas come up!

Vision & Technologies

Parallel Processing is a key at All Layers!

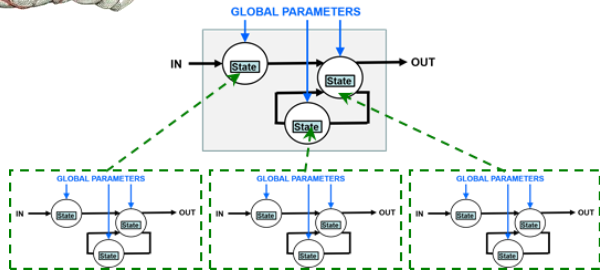
Next-Generation System
Application/Algorithm

AI Camera



Parallel Processing Software Design

DeepPN
CoolParallel SDE



Software Implementation

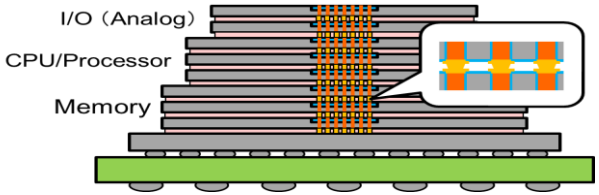
Tuning/Optimization Service

Parallel Processing CPU/Processor

SMYLEvideo, SMYLEdeep/ZOMP
TOPSTREAM™ RTRT
TOPSTREAM™ Bus

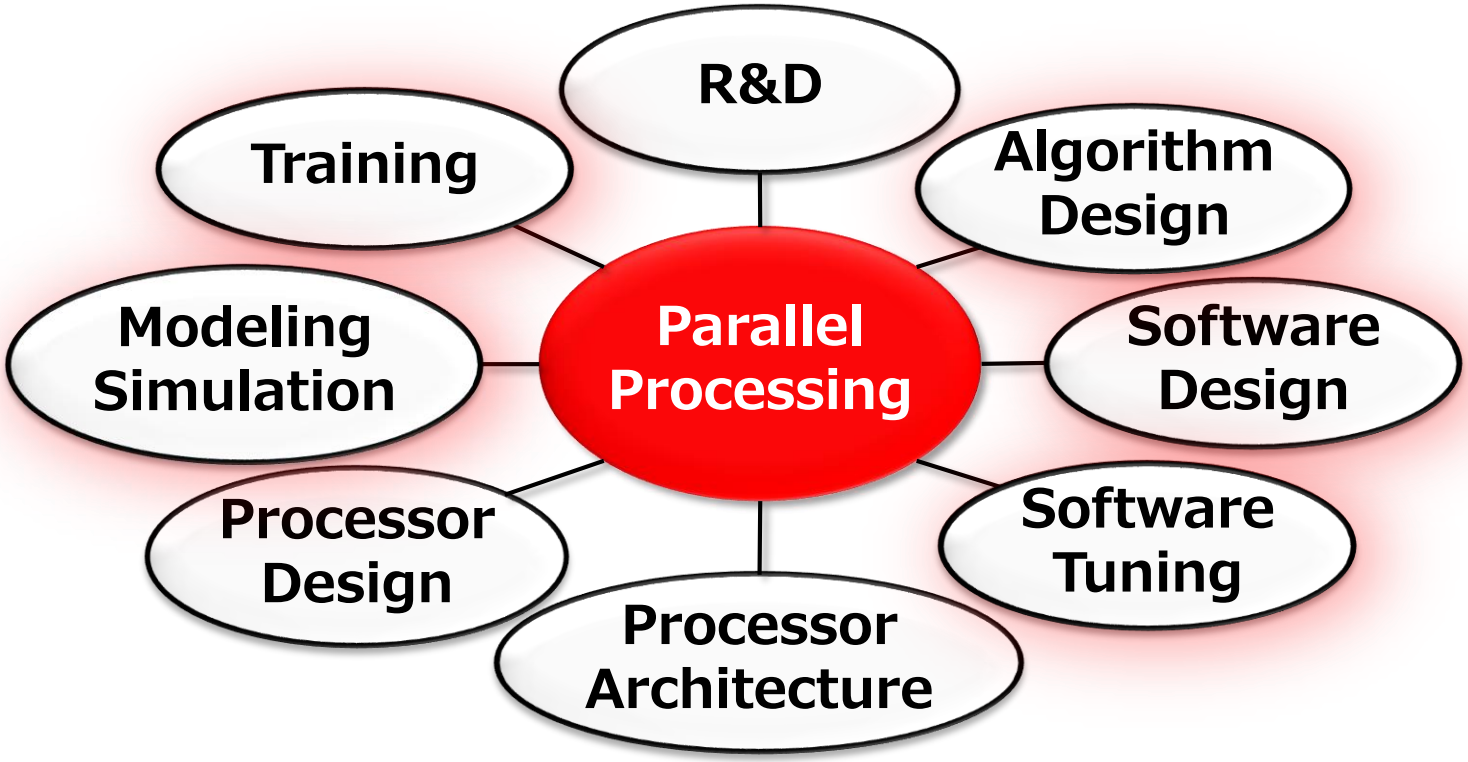
Scalable 3D LSI Stacking

Cool Interconnect



Today's Talk : Designing Parallel Processing

Development Services / TOPS Systems



CPU GPU

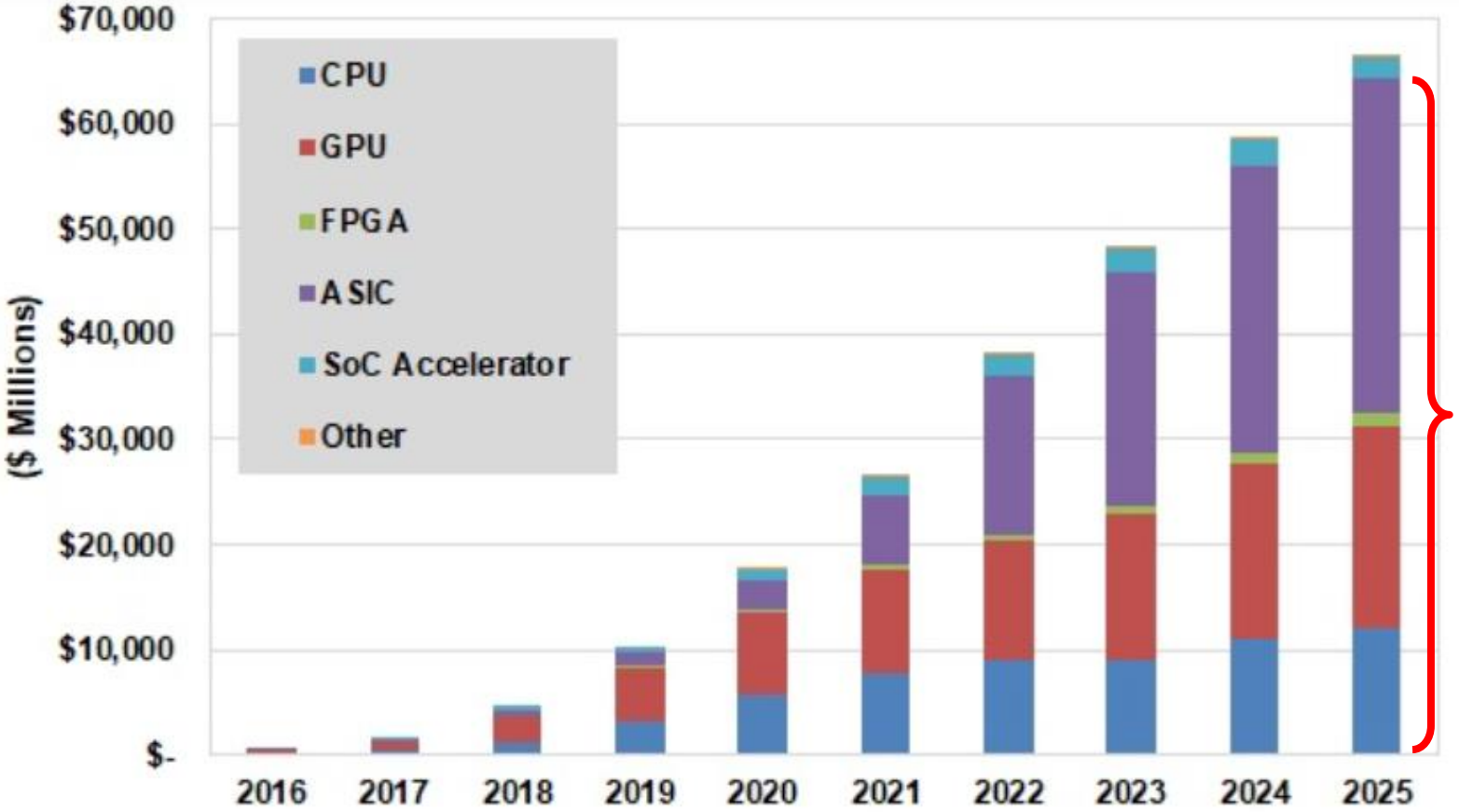
FPGA MPSoC

SMYLEdeep

Today's Talk : Ideas from experiences with customers

Demand of Parallel Processing in Real-Time Systems

Market Trend toward Parallel Processing



Deep learning chipsets by type. Source: Tractica

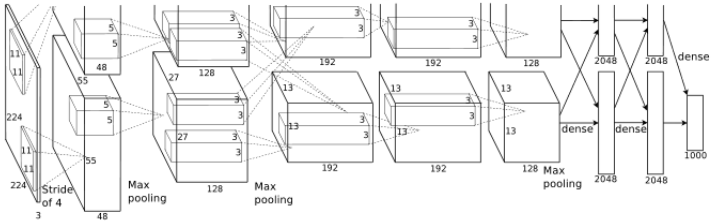
More than 90% of Hardware Platform is for Parallel Processing

Demand of Parallel Processing in Real-Time Systems

Examples : Next-Generation Automotive

> 60fps

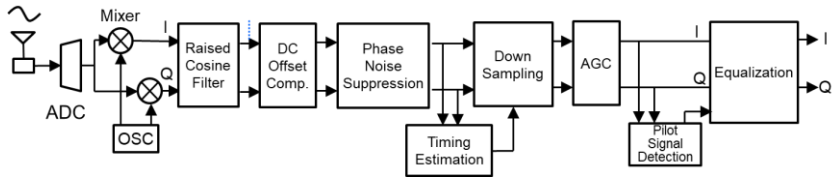
~4k



Visual Recognition/AI

~86GHz

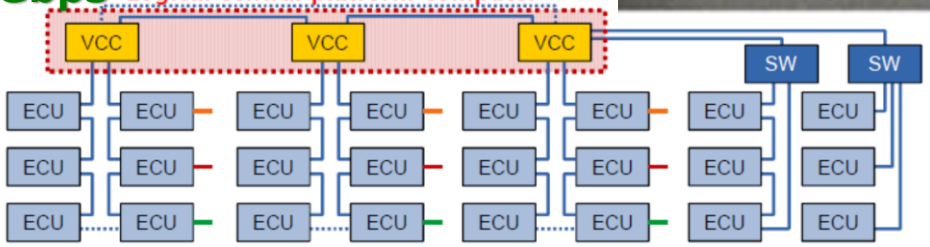
< 1ms



URLLC/5G

< 4μs/hop @ 1Gbps

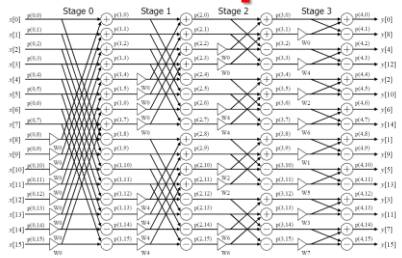
~10Gbps



TSN/In Vehicle Network

~ 96Gsps

~32k-points



FFT/Signal Processing

Efficient Parallel Processing is must for hight Performance and Energy-Efficiency

Demand of Parallel Processing in Real-Time Systems

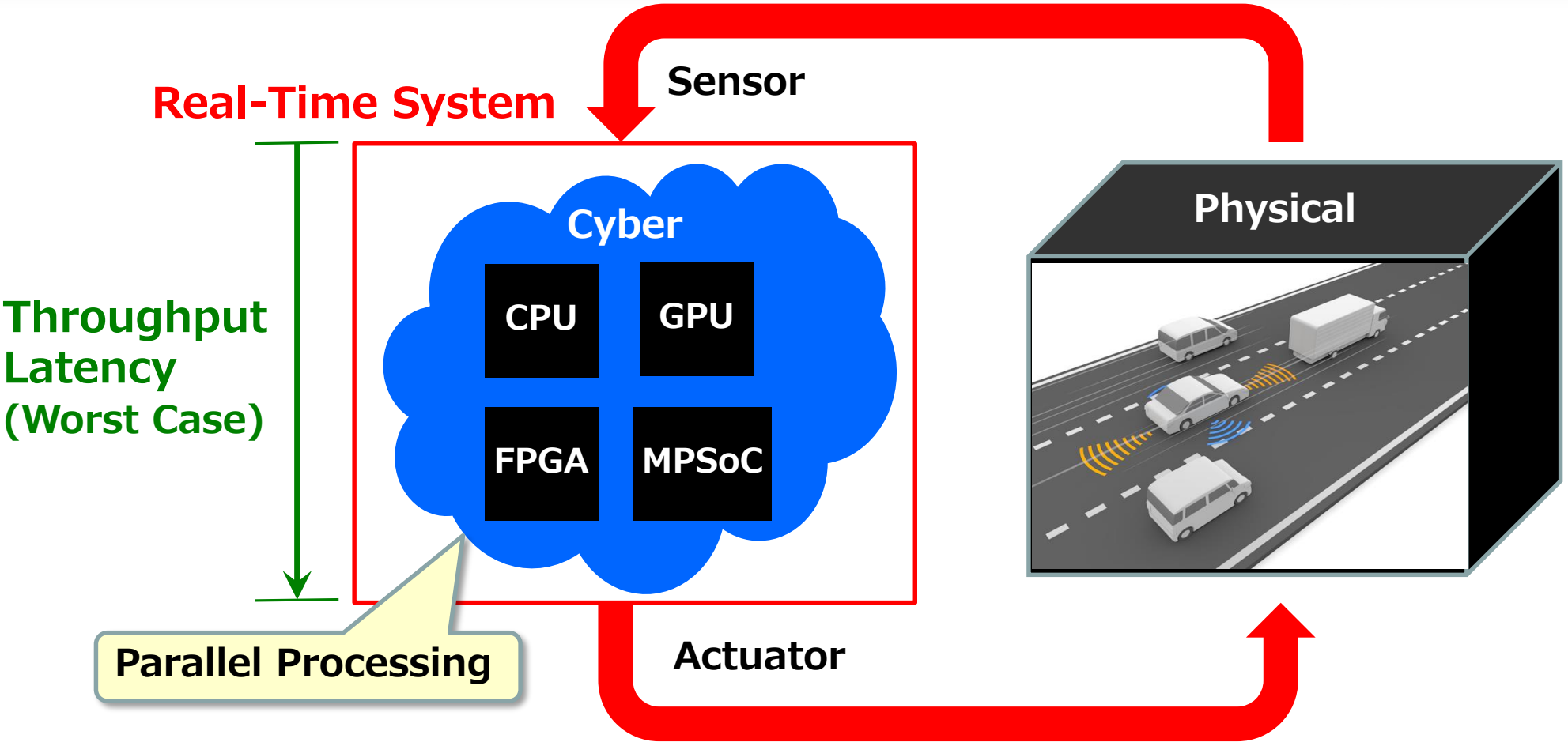
Examples of Parallelization Requirements from Constraints for Edge

		Metrics	Unit	FFT	DNN	10GbE	Security	ISP
Performance	Computing	Floating	TFLOPS	1	n/a	n/a	n/a	n/a
		Integer	TOPS	n/a	1~10	1~	0.01	1~
	Memory	Bandwidth	GB/Sec	32	1~2	30	1.2	4~
	Power Consumption		W	~10	1~1.5	~5	0.1	~1
	Chip Size	Area @ 28nm	mm^2	100	50	100	10	50
	Cost	Device	$	50	50	50	1	3
Efficiency	Hardware	Energy-Efficiency	TFLOPS/W	0.1	n/a	n/a	n/a	n/a
			TOPS/W	n/a	1	~0.2	0.01	1
		Area Efficiency	$TFLOPS/mm^2$	0.01	n/a	n/a	n/a	n/a
			$TOPS/mm^2$	n/a	0.02~0.2	0.01~	0.001	0.02
		Cost Efficiency	TFLOPS/$	0.02	n/a	n/a	n/a	n/a
			TOPS/$	n/a	0.02~0.2	0.02~0.2	0.01	0.33
Other	Scalability	Maximum Frequency	MHz	300	300	300	300	300
		Parallel Processing	Number of Parallels	3k	3k~30k	3k	16	3kpixel
	Note	Design Consideration		1MFFT/s ~4096pnt SingleFP 16Layer Prallel FFT	30fps Full-HD Object Rec GoogLeNet AlexNet Squeeze RezNet	100Gbps 10port Table Srch QoS TCP Mon. Security Data Comp	10Gbps AES n-bit Key 32x128 Mul	7Mpixel 120fps Focus White Ballance Color Conv.

3k-30k Parallel Processing is required to meet Performance and Efficiancy

Demand of Parallel Processing in Real-Time Systems

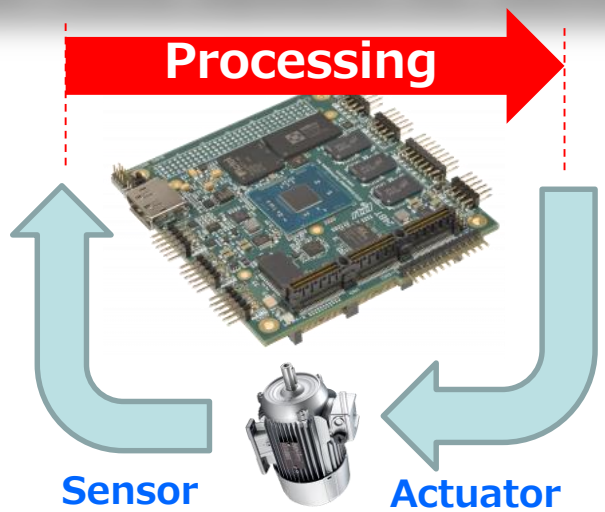
Edge Computing on Cyber Physical System (CPS)

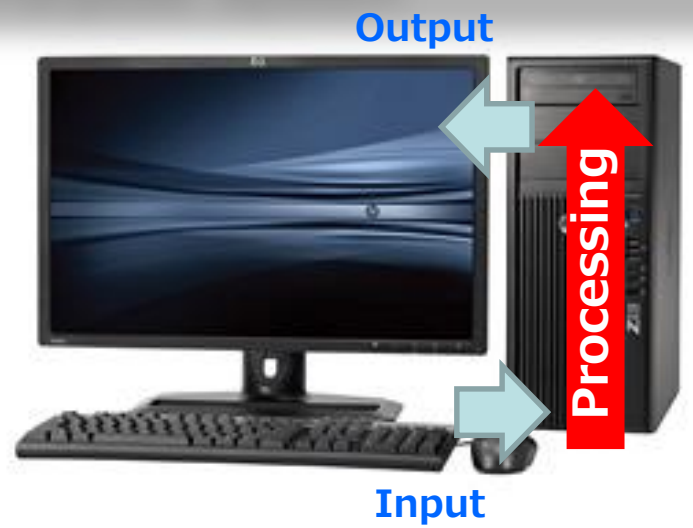


Efficient Parallel Processing on Edge Devices for High-Throuput and Low-Latency

Demand of Parallel Processing in Real-Time Systems

Real-Time System vs. General Purpose System





	Real-Time System	General Purpose System
Continuity	Yes	No
Deadline	Exist	None
Predictable Response Time	Must	No
Data	Used Immediately	Used by Processing Order
Type of Processing	Stream/Dataflow	Batch/Multi-Thread

Processing of Real-Time system is type of Stream Processing based on Dataflows

Challenges on Designing Parallel Processing

What currently facing in the industry

#1. Free Lunch is over, but still stick with sequential programming

#2. Almost No-Effort on Software Design

#3. Lack of Expression of Timing and Parallelism in Design tools

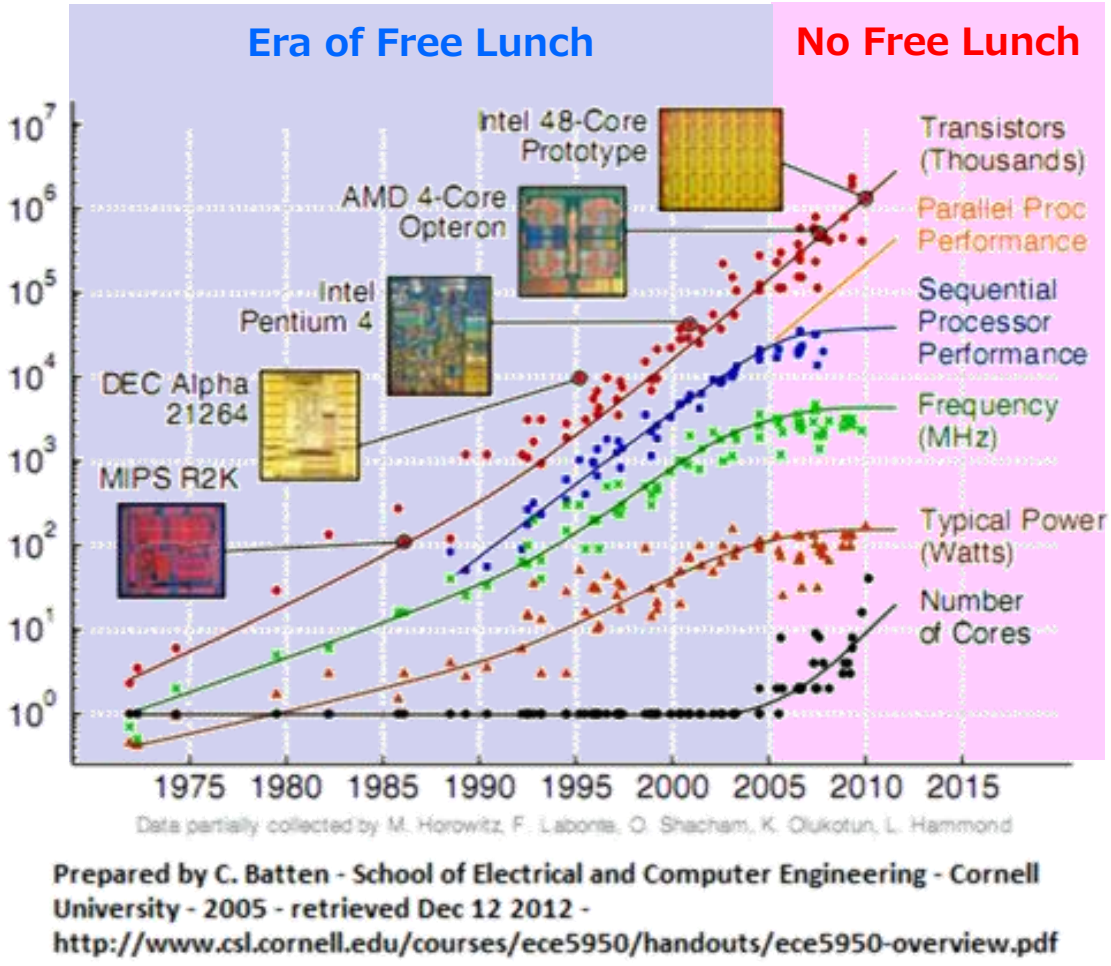
#4. Many Performance Black Boxes in Software and Hardware stack

#5. Limited Scalability in Hardware Platform

Very hard to utilize potential performance of CPU, GPU, FPGA

Challenges on Designing Parallel Processing

Free Lunch is over in 2005, but ...



Prepared by C. Batten - School of Electrical and Computer Engineering - Cornell University - 2005 - retrieved Dec 12 2012 - http://www.csl.cornell.edu/courses/ece5950/handouts/ece5950-overview.pdf

- **Change the meaning of programs**
  far more than optimizations
  - **Conventional**
    Programs for Functionality
    - Single Program (CPU)
    - Huge Memory

    Optimizations of Implementation
    - Mostly by Compiler
  - **After Free Lunch**
    Programs for Performance
    - Multiple Programs (Multiple CPU)
    - Multiple Memories (Distributed)

    Optimizations of Design
    - Consider Time and Space

- **Change the Characteristics of CPU**
  Conventional Software maybe slower

#1. Free Lunch is over, but still stick with sequential programming

Challenges on Designing Parallel Processing

What optimization required after "Free Lunch is Over"

Design :Optimizations Required for Parallel Processing

Specification

Optimize Algorithm
- Choose Algorithm optimum for **parallel resources on target hardware**, such as **ALU/MAC**, **mem**, and **comm./sync** capability.

Optimize Design
- **Parallel Processing Structure optimum for target hardware**
- Optimize Design for **parallel resources on target hardware**

Optimize Program
- Modify Source Code for **optimizabtion by compiler**

Program :Optimizations for Sequential Processing

Source Code

Optimize by Compiler
- optimization algorithms for **sequential code**

Optimize by Library
- Use optimum library that code has pre optimized for **sequential performance**, such as **Math**, **signal processing**, **DNN**.

Optimize for Target
- Fully utilize **special resources and features** on target hardware, such as **special instruction**, **register-files**, **DSP, e-Memory**

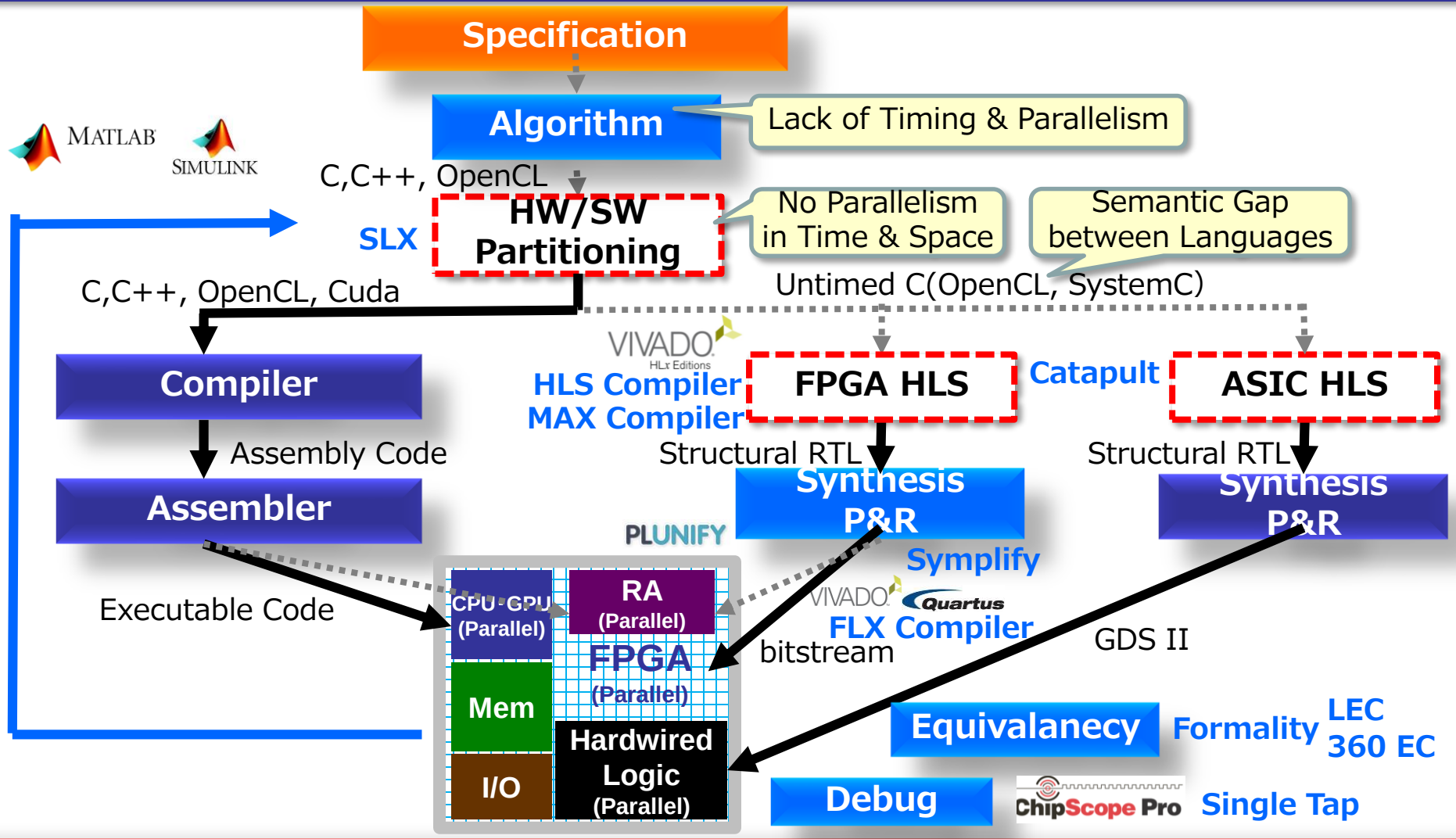
Executable

Optimiza for Dynamic
- Dynamically utilize resource on target hardware, such as **CPU utilization**, **SuperScalar**, **$ hit rate**, **Branch Prediction Rate**.

#2. Almost No-Effort on Software Design

Challenges on Designing Parallel Processing

Many Performance Black Boxes in Design Methodologies



#3. Lack of Expression of Timing and Parallelism in Design tools

Challenges on Designing Parallel Processing

Performance Black Boxes = Unpredictable/Uncontrollable Execution Time

$$Excution\ time = Number\ of\ Instruction \times CPI \times Cycle\ Time$$

- **Software**
  - Number of Instructions
    - Compiler Optimization
    - Interrupt
    - Scheduling/OS

- **Hardware**
  - CPI
    - Instruction Cache Miss
    - Data Cache Miss
    - Branch Prediction Miss
    - Super Scalar Scheduling
    - Inter-Core Synchronization
  - Cycle Time
    - Dynamic Frequency Scaling

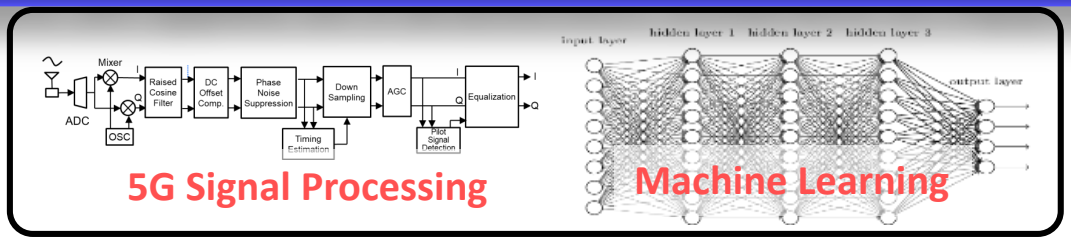
#4. Many Performance Black Boxes in Hardware stack

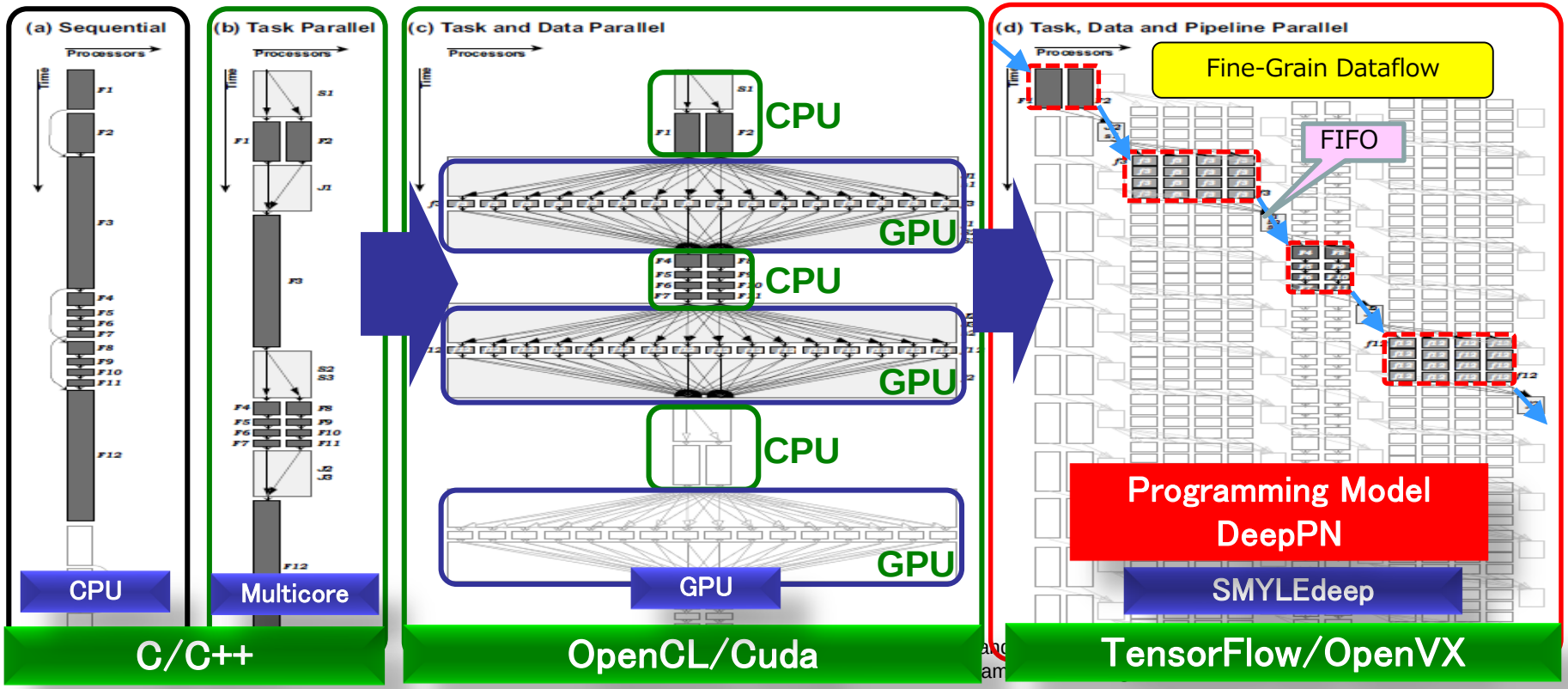
How to approach the challenges

Methodologies

Utilizes all types of Parallelism inherent in Applications

Real-Time Applications





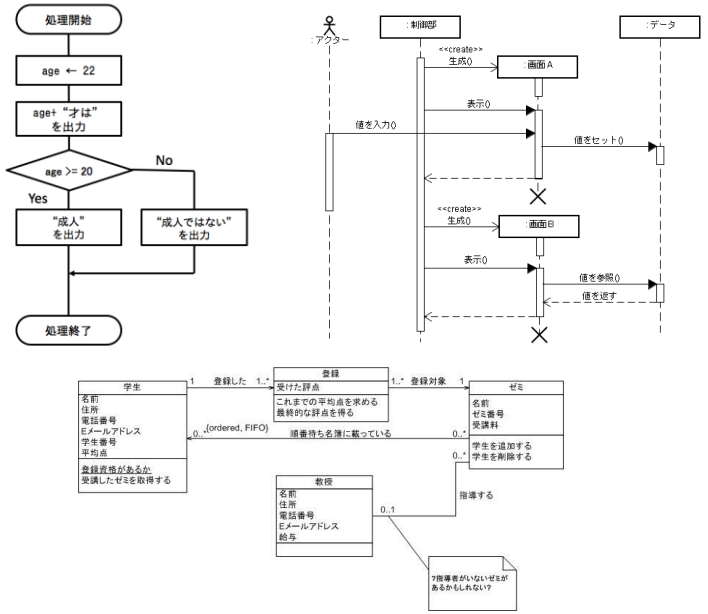
Dataflow allows easy extraction and optimization of spatial parallel processing

Methodologies

Basic Model of Processing

Conventional Expression

- Flow Chart, Sequence Diagram

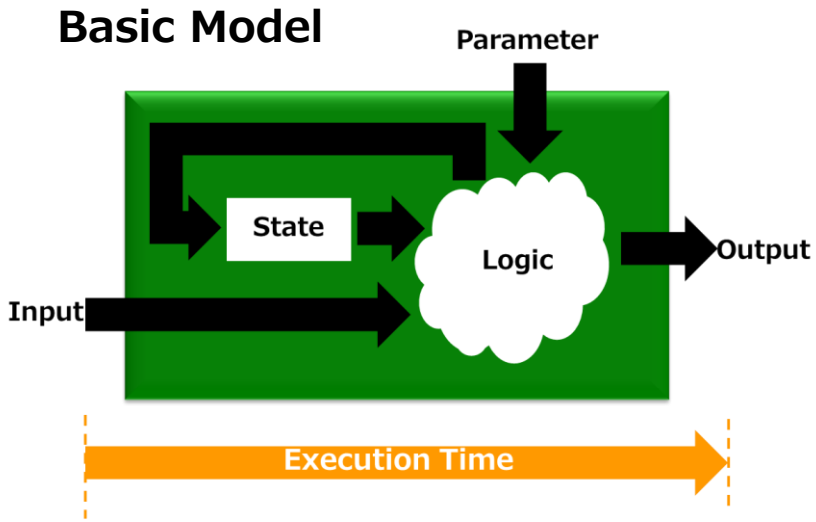


Issue : **Implicit Performance information**

- **No Data Size**
- **No Data Dependency**
- **No Timing**

Quantitative Expression

- Input: Byte
- Output: Byte
- Parameter: Byte
- State: Byte
- Logic: Number of Inst.
- Execution Time: Cycles

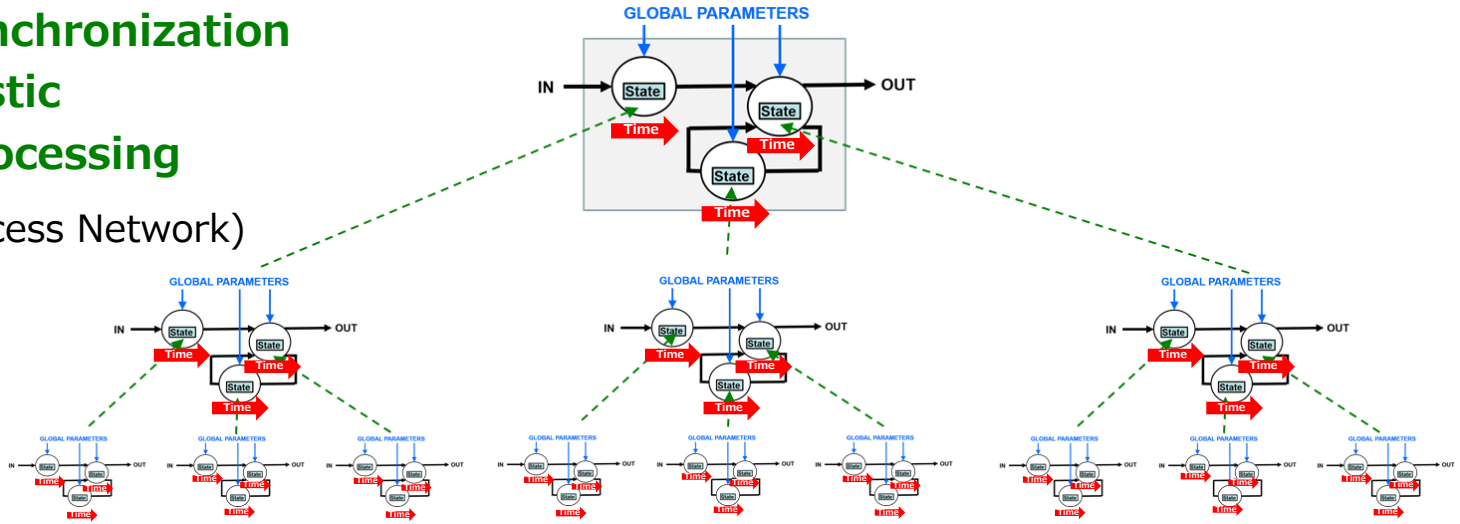


Execution Time = function(input, parameter, state)

Methodologies

Model of Parallel Processing : DeepPN

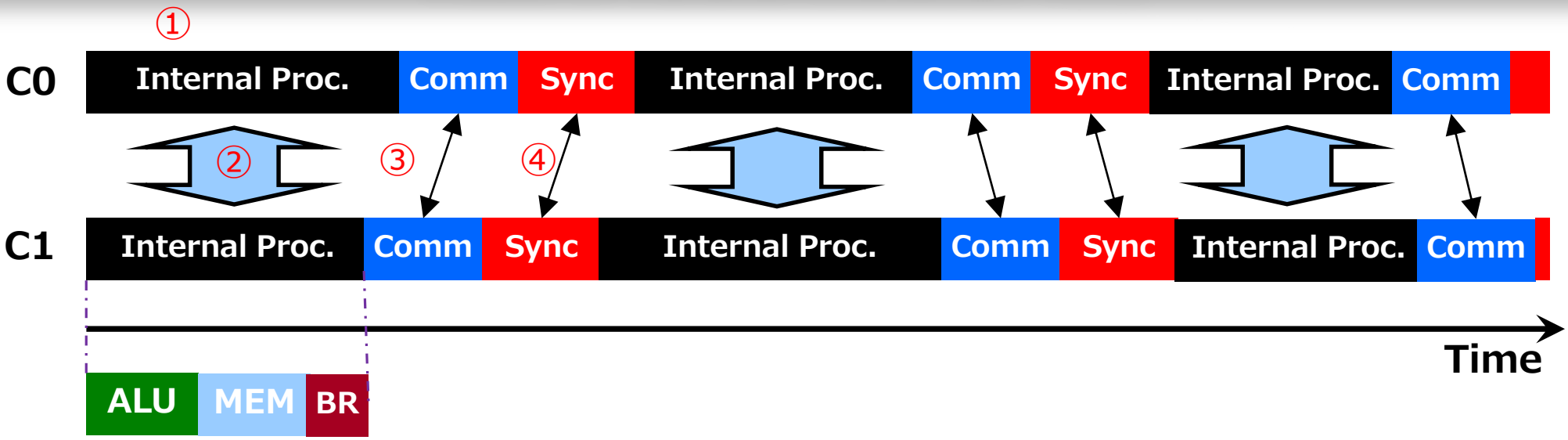
- Issues : **Current Programming limits Parallel Processing**
- Objectives: **Define how to express parallel processing exlicitly**
  - **Easy to Express parallelism inherit in Real-Time Applications**
  - **Easy to speed-up Real-Time applications by optimization of**
    - **Efficient Parallel Processing**
    - **Less Memory Access**
    - **Simple Communication**
    - **Simple Synchronization**
    - **Deterministic**
    - **Stream Processing**
- **DeepPN** (Deep Process Network)



DeepPN : Dataflow Graph, Hiearchy, Gobal Variable

Methodologies

Model of Parallel Execution Time



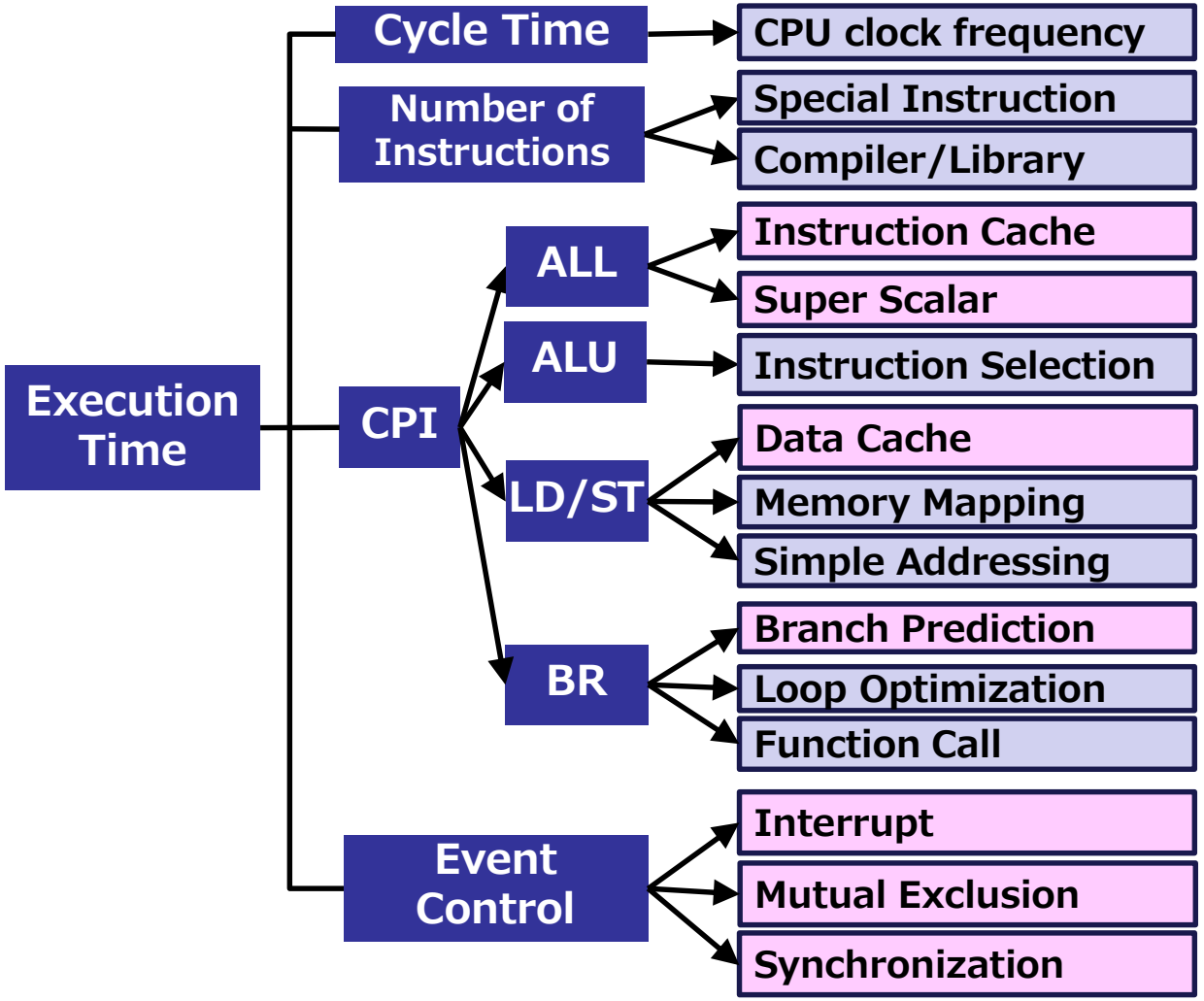
■ **Parallel Execution Time**

① Internal Processing	; ALU, Memory(LD, ST), Branch
② Inter-Core Conflict	; Shared Memory Access
③ Inter-Core Communication	; Data Transmit and Receive
④ Inter-Core Synchronization	; Wait for Lock, Barrier

For faster processing, minimize each by design optimization

Methodologies

For Predictable/Controllable Processing Execution Time



Static Dynamic

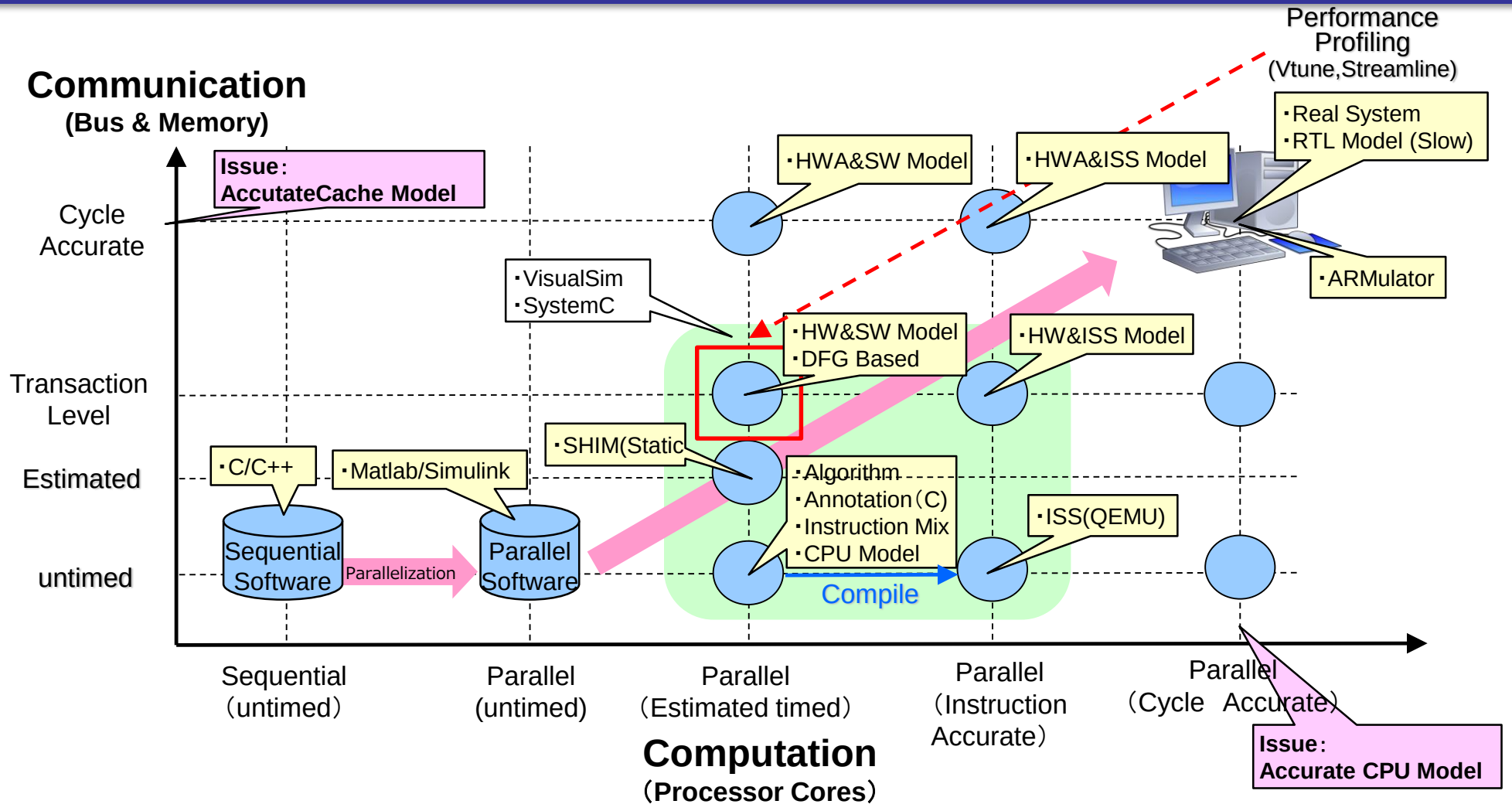
Control CPI

	Practical Goal	Ideal Goal
Instruction Cache Hit	>98%	100% w/o Cache
SuperScalar	1.5	Fixed Static
Data Cache Hit	>96%	100% w/o Cache
Branch Prediction Hit	>98%	100% w/o BP

Control and Predict Dynamic Behavior

Methodologies

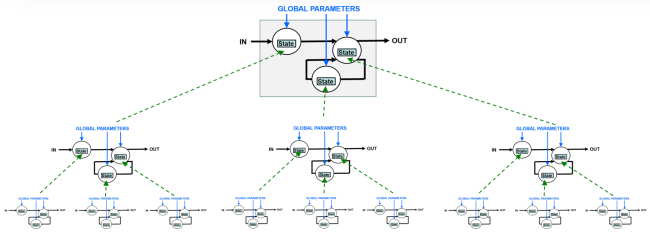
Modeling & Simulation of Parallel Processing Performance



Design Phase : Quick Modeling with ±20% of Accuracy

Methodologies

Guidelines and Tools



DeepPN Programming Model

MATLAB SIMULINK TensorFlow OpenVX Vision Acceleration

DFG

Application (Samples)

Guidelines for Parallel Processing

Tools & Modules

Parallel Processing Design Environment

Modules & Libs DeepPN

Design

DFG

DeepPN API

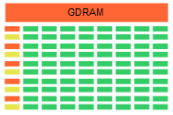
SHIM*

Parallel Processing Implementation Environment

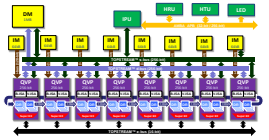
Implementation

DFG

DFG



GPU

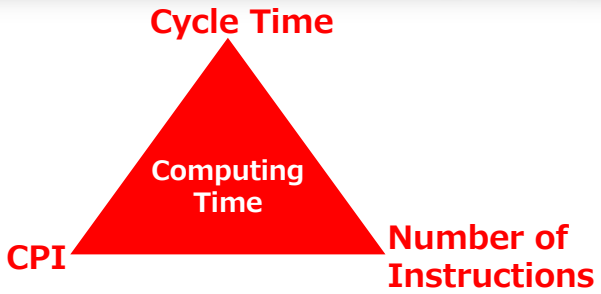


SMYLE$_{RT}$

TOPS Systems plan to provide Guideline and Tools

Architecture

Decrese Computing Time and Increase Energy-Efficiency



Increase Energy-Efficiency

$$Performance\uparrow = OPC\uparrow \times f\downarrow$$

$$Power\downarrow = \frac{1}{2}\,\alpha\, C\, V^2 f\downarrow$$

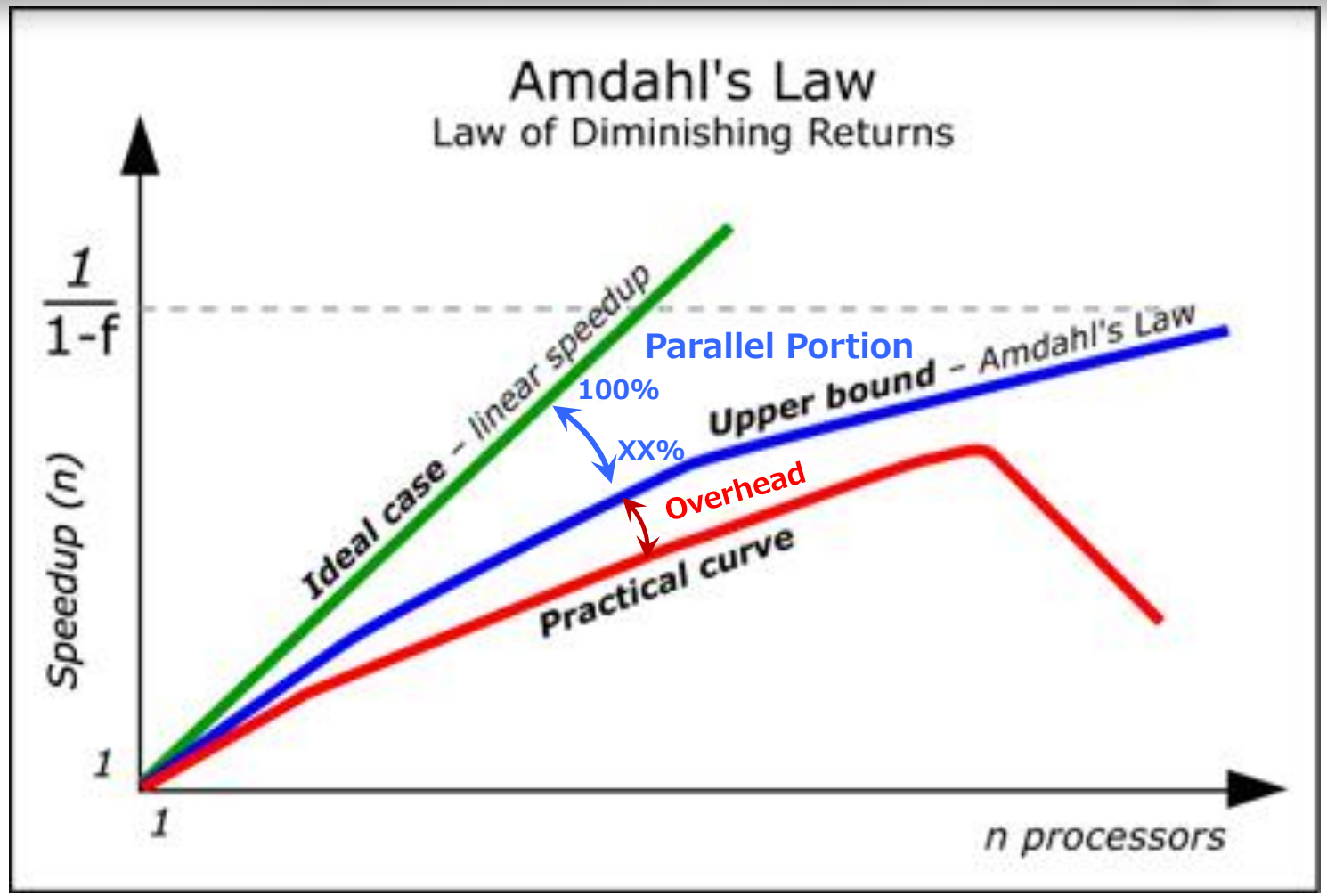
Theory for Decrease Computing Time	Break Down	Traditional Approaches	SMYLEdeep's Approaches
Less Cycle Time	Clock Frequency	High Clock Rate(GHz)	Low Clock Rate(100MHz)
Less Number of Instructions	ALU	SIMD(8-32packed)	Special(-768operations)
	Load	Single, Block	Less, Stream(No Latency)
	Store	Single, Block	Less, Stream(No Latency)
	Branch	Bcc, Jmp, Call/Ret	Bcc, Jmp, Call/Ret
	Communication	Store & Load	None
	Synchronization	Lock	None(Prefix)
Less CPI (Clock per Instruction)	Cache	Instruction/Data	Instruction/Large RegFile
	Super Scalar	-4way	None
	Branch Prediction	Yes(due to deep pipe)	No(shallow pipeline)

ZOMP

Tightly Coupled Multi-ISA cores optimum for Efficient Parallel Processing

Architecture

Common Issue on Parallel Processing

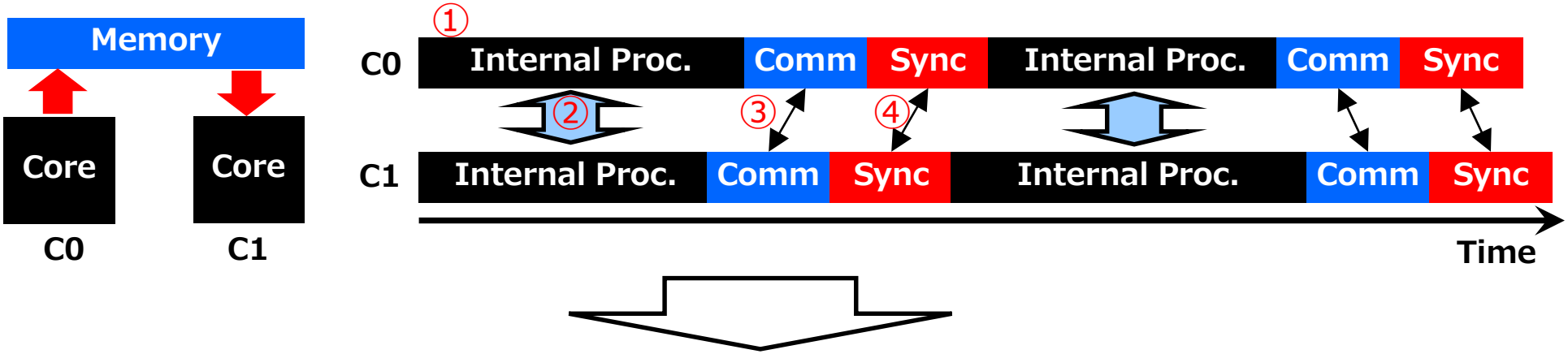


Goal : Increase Parallel Protion and Minimize Overhead

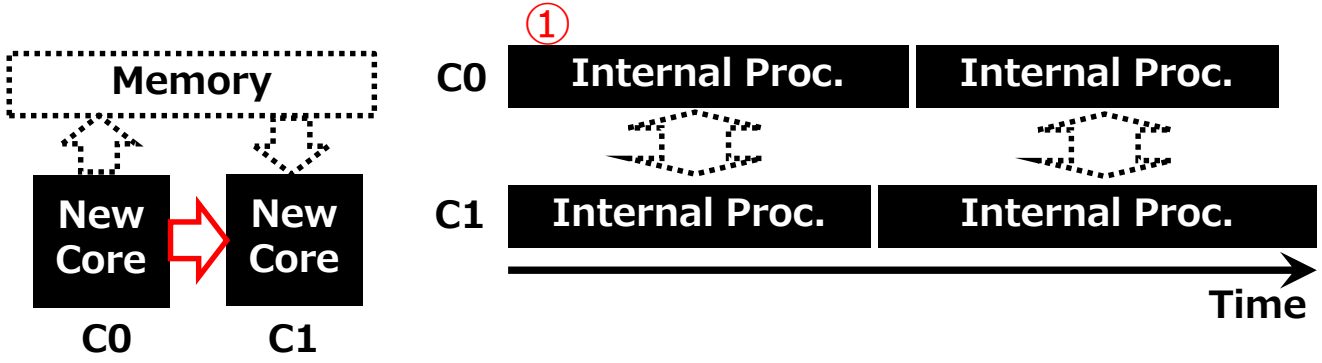
Architecture

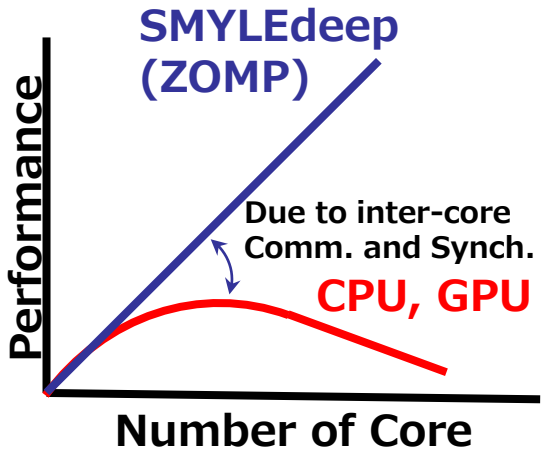
Zero-Overhead Message Passing (ZOMP)

■ Traditional Parallel Processing



■ SMYLEdeep: Closely Coupled Special Multi-core

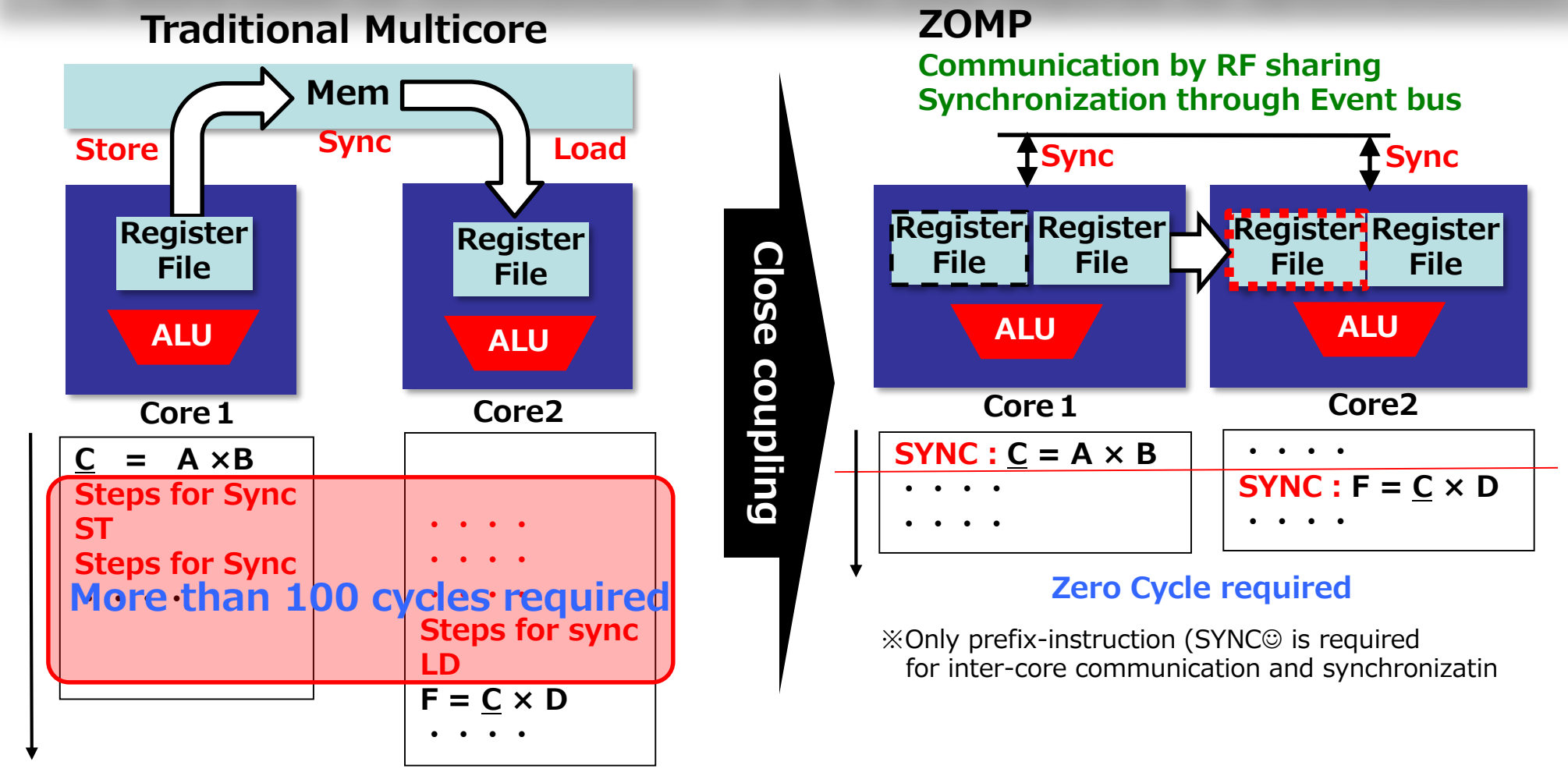




Remove Overhead due to Inter-Core Communication and Synchronization

Architecture

No Store-Load for communication and No Lock required for synchronization

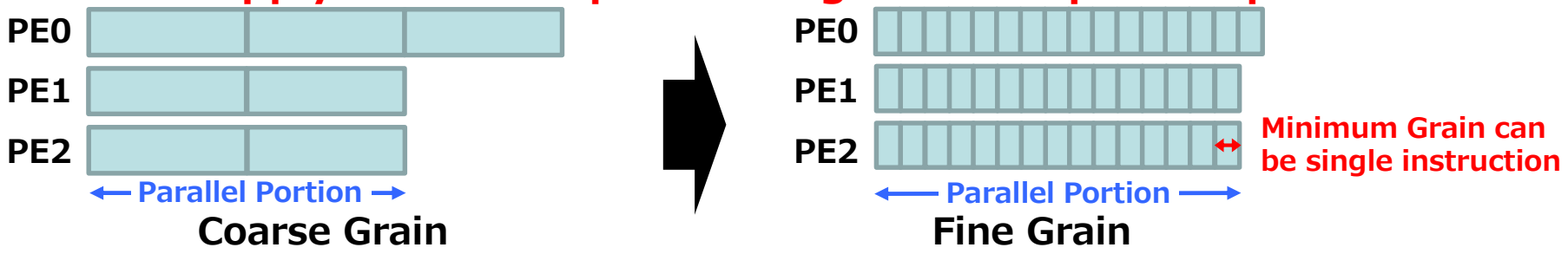


Two Programs can communicate and synchronize with Zero Cycle!

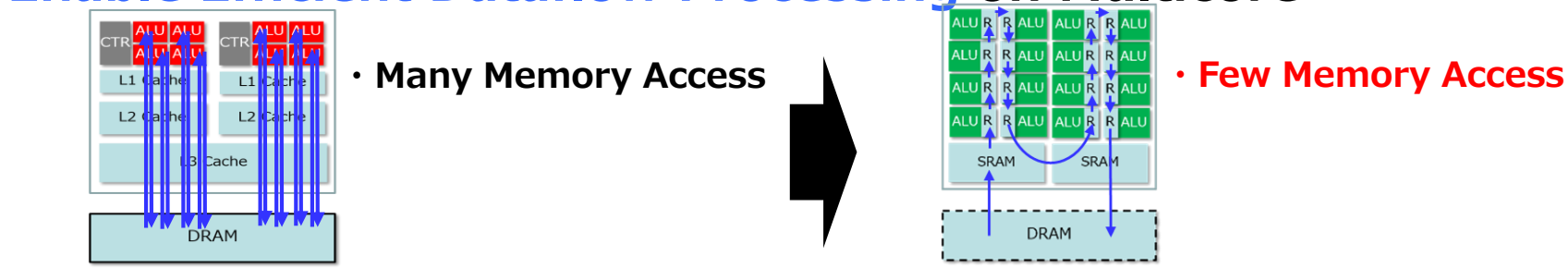
Architecture

You can scaler performance of Multicore linearly with number of cores

- **Zero Overhead cycles for Inter-Core Communication**
  - You can reduce number of ST/LD instructions
- **Zero Overhead cycles for Inter-Core Synchronization**
  - You can remove LOCK instructions for mutual exclusion
- **Increase Parallel Portion of Multicore Software**
  - You can apply Fine Grain partitioning for more parallel portion



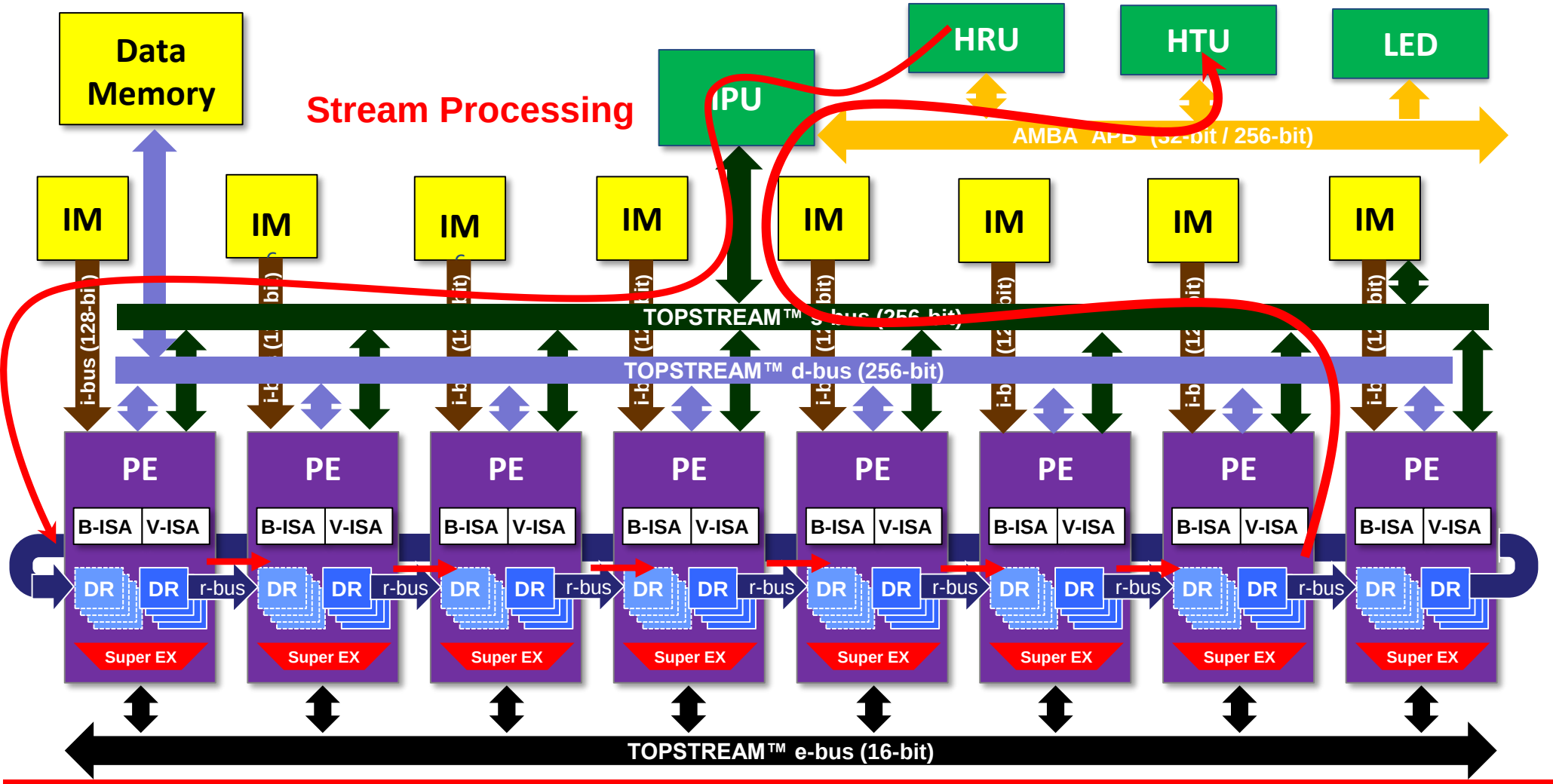
- **Enable Efficient Dataflow Processing on Multicore**



ZOMP enables Scalability and Efficient Parallel Processing

Architecture : SMYLE$_{Deep}$

Tightly Coupled Cores for efficient parallel processing



#5 Scalable and Deterministic Hardware Platform

Architecture

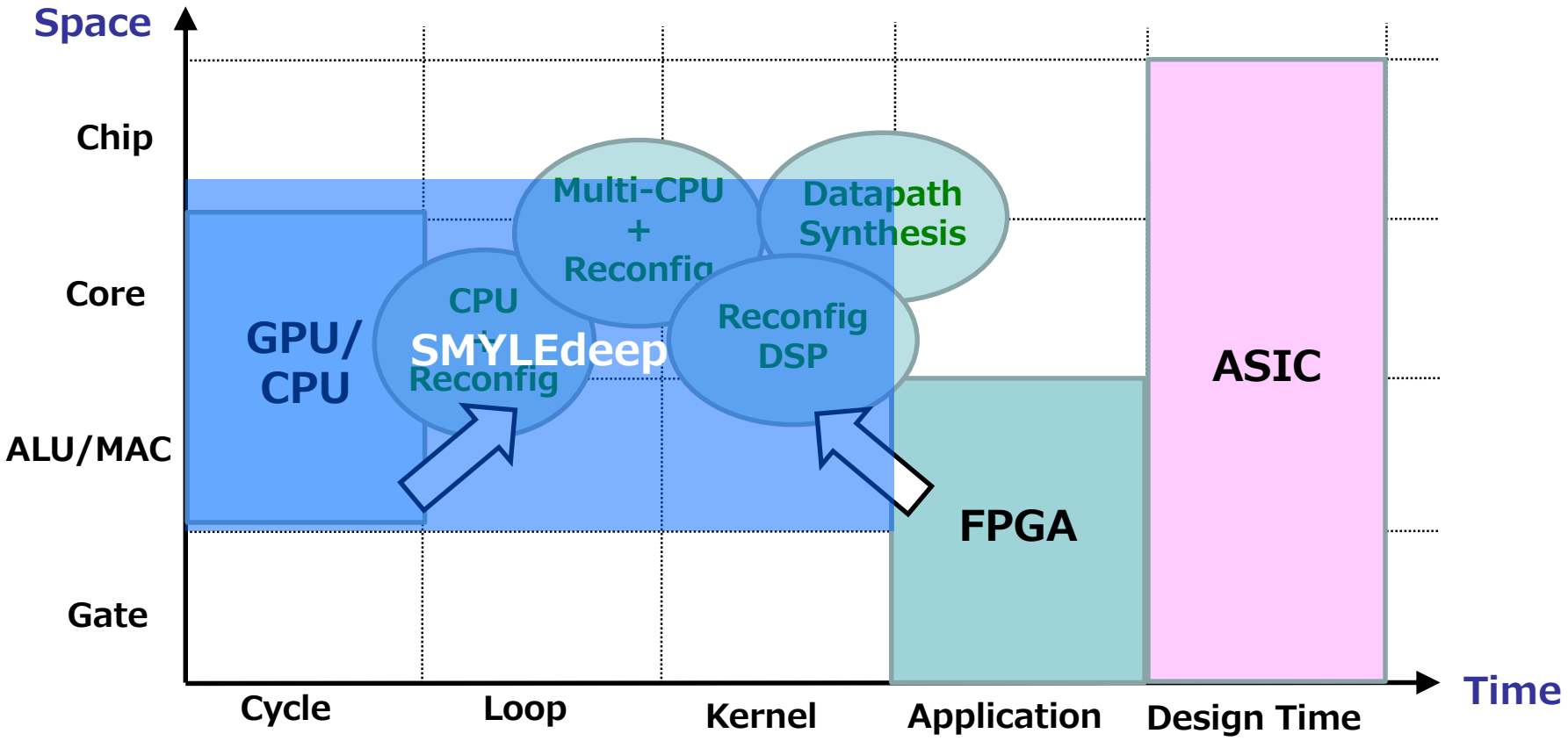
Parallel Processing on SMYLEdeep

- **Task Parallel**
  - Each core executes different tasks independently with MIMD
- **Data Parallel**
  - Each core execute same code as SIMD
- **Pipeline Parallel**
  - Each core executes different tasks independently with MIMD and pass the output to next core through R-bus with zero cycle
- **Mixture of Task, Data, Pipeline processing**

Applicable to type of Real-Time Applications

Types of Hardware Architecture

Utilize Advantage of configurability in Time and Space for Parallel Processing



SMYLEdeep: Tightly Coupled Parallel Processing Architecture

Solutions

#1 Top down design with Time and Parallelism

- Avoid troublesome exploration of parallelism at implementation

#2 Intensive Software Design prior to programming

- Migrate to explicit Processing Model
- Provide Parallel Software Design Guideline and Software Tools

#3 Structural coding of Parallelism

- Shifting to modules according to Dataflow

#4 Design Optimization based on White Box Performance

- Quantitative Performance Analysis and Estimation
- Remove Unpredictability / Uncontrollability

#5 Scalable and Deterministic Hardware Platform

- Provide SMYLE$_{RT}$ for optimum and efficient real-time processing

We provides services and products for “Highly Efficient-Parallel Processing”

Toward Efficient Parallel Processing

Changes

from Programming to Software Design
from Shared Memry to Direct Inter-Core Comm.

Looking for Academic and Industrial Partners to provide solutions for WW MPSoC users!

Go Next Generation

yukoh@topscom.co.jp