Debugging of application software based on KPN on heterogeneous multi / many-core processors

Yukoh Matsumoto, Ph.D.
President & CEO, Architect

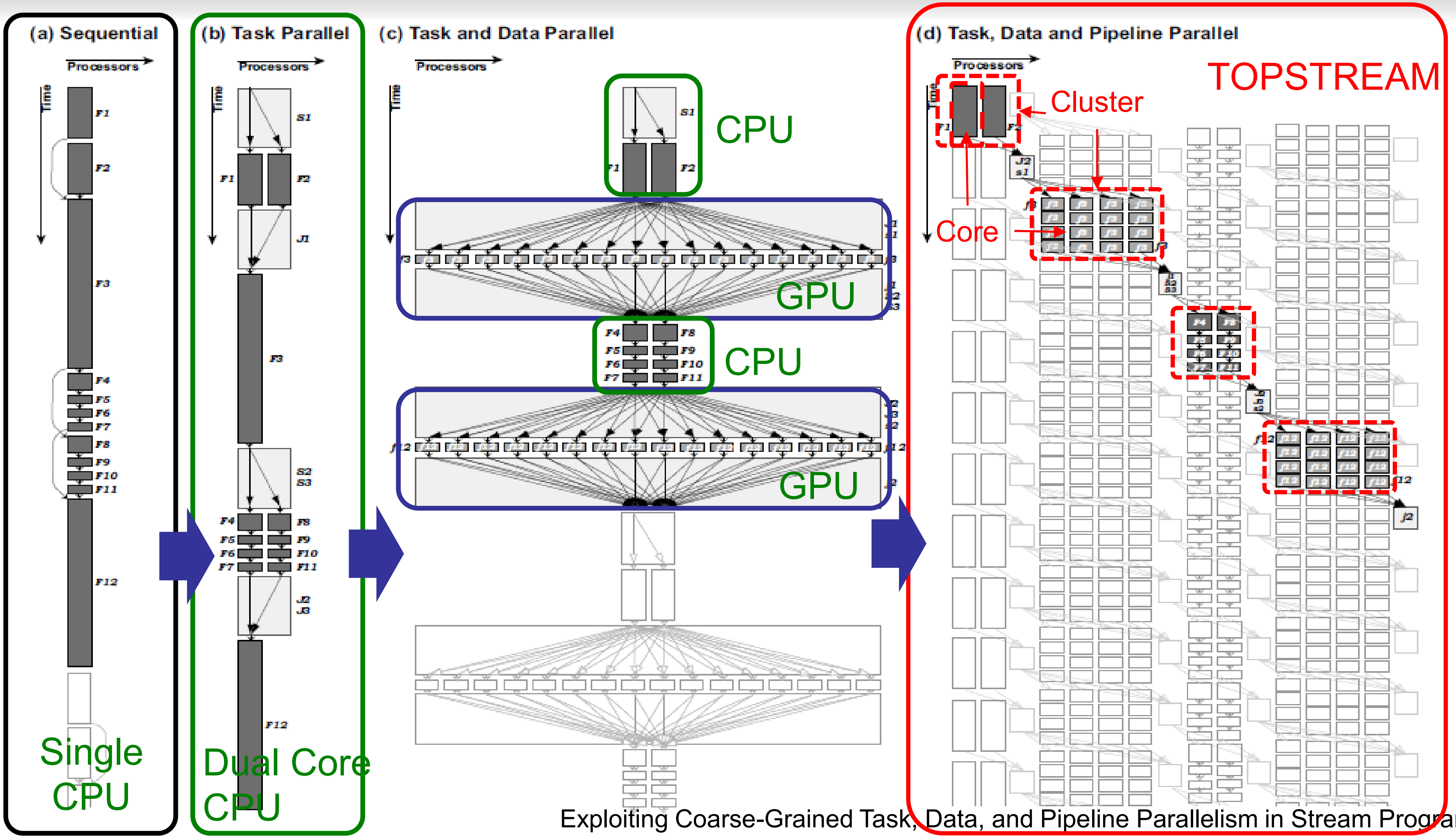
TOPS Systems Corp.

Multicore / Manycore provider in Japan

Agenda

- *Porting Application onto Heterogeneous Manycore*
- *Case Study : Real-Time Ray Tracing, 800TFLOPS on Desk Top Machine*
- *Architecture & Algorithm Co-Design*
- *Deep Performance Analysis*
- *Software Partitioning into Kahn Process Network*
- *System Performance Modeling*
- *System Performance Simulation*
- *Debugging Issues and Challenges*
- *Working on Better Solutions*
- *Conclusions*

Parallel Processing Goal

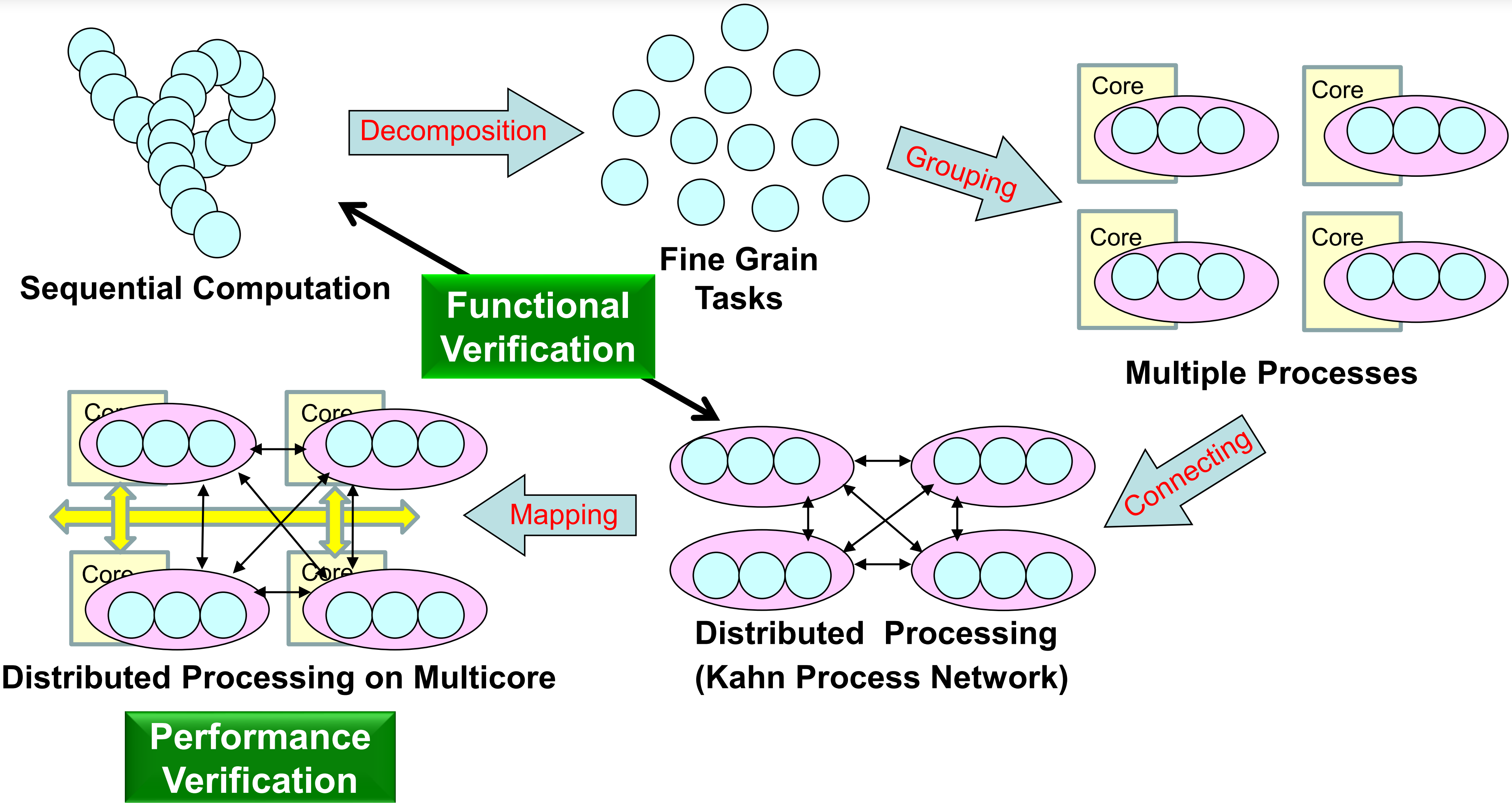


Exploiting Coarse-Grained Task, Data, and Pipeline Parallelism in Stream Programs
Michael I. Gordon, William Thies, and Saman Amarasinghe, MIT

Exploit more parallelism for higher performance

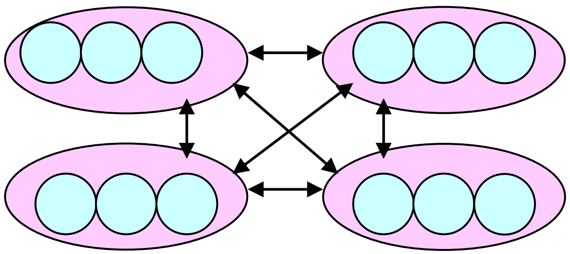
Our Multi-/Many-core SW Development Flow

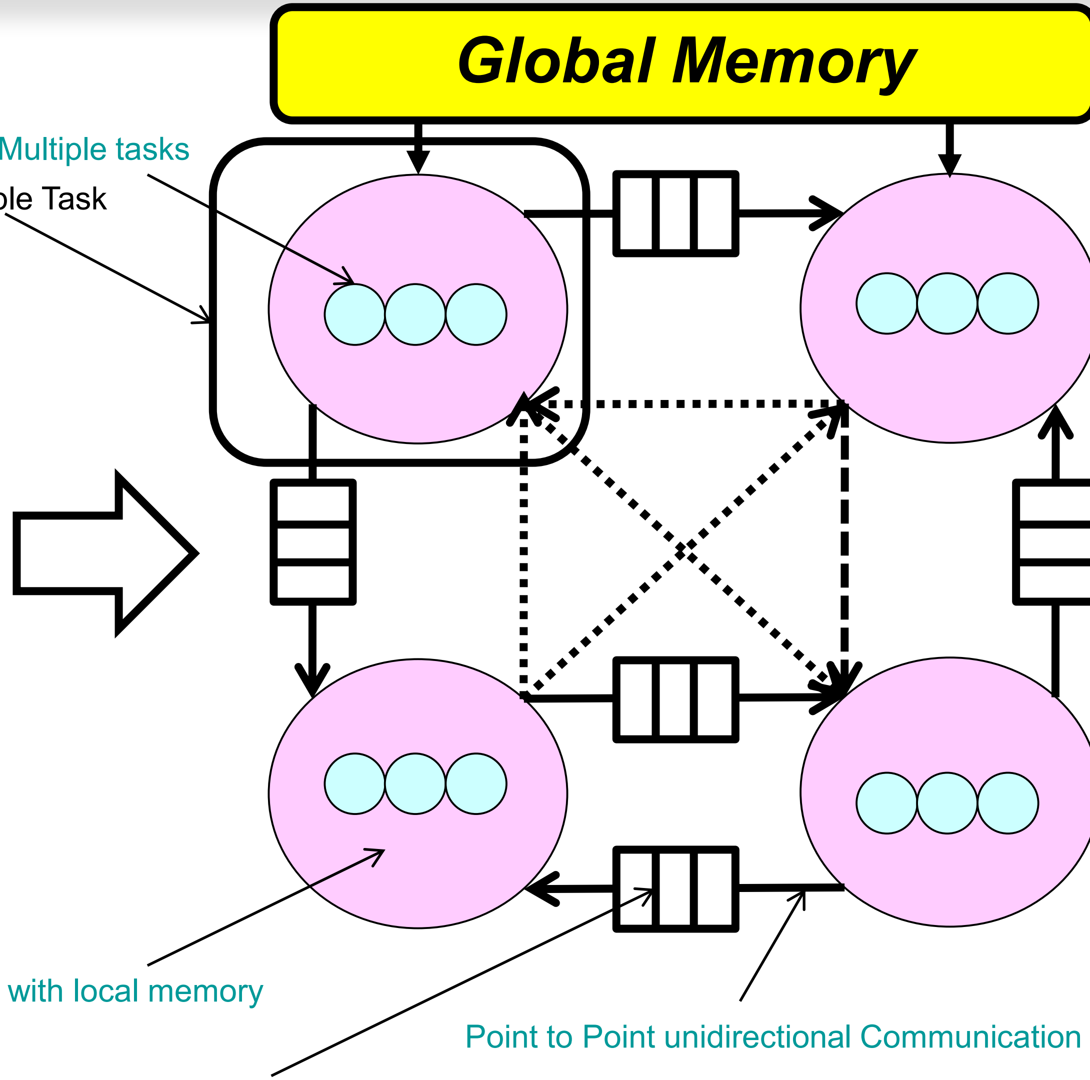
Sequential to Distributed Processing



Debugging of both Functionality and Performance

Programming Model based on KPN





Allows read only access to Global Memory

Experiences : Sequential to Distributed Processing

- Computer Vision
  - SIFT ; 10 cores, 4 cores, 8 cores
  - Haar-Like ; 4 cores
  - SVM ; 4 cores
- Computer Graphics
  - Ray Tracing ; 73 cores
- Codec
  - H.264 Decoder ; 10 cores
  - JPEG Decoder ; 10 cores
- Wireless Communication
  - 802.11b MAC and Baseband ; 4 cores

Distributed Processing on Heterogeneous Multicore / Manycore

Case Study：TOPSTREAM™ RTRT

・Intel CPU: 100k Chips
・NVIDIA GPU: 20k～30k Chips
・TOPSTREAM: 9 Chips

Ultra-Accurate Real-Time Ray Tracing

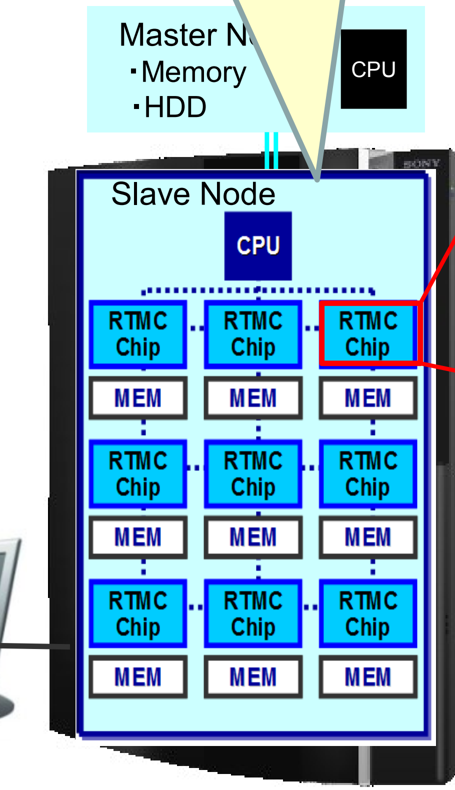
✓ Color Model with 35 bands
✓ Rendering on Free Surface (Bezier)
✓ HD (1920 x 1080 pixels) @ 30frame/s

Performance Requirement

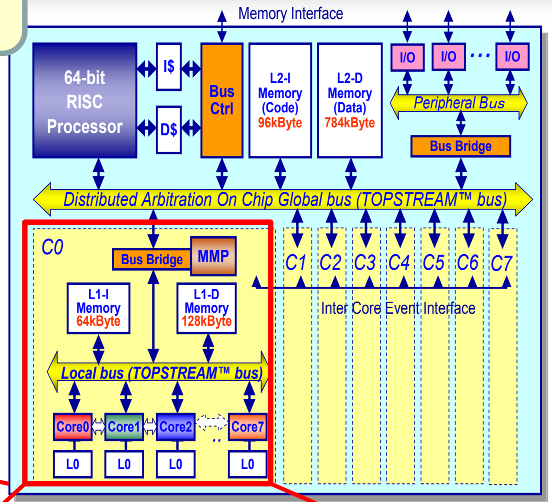
➢ ≒800 TFLOPS / system; 88TFLOPS / chip

LSI Design (Estimated)

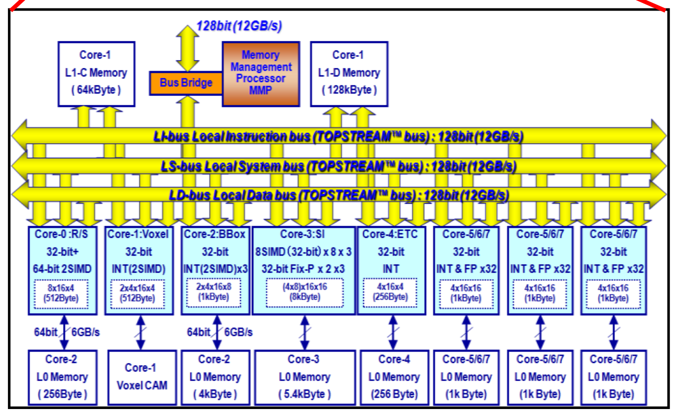
➢ Technology : TSMC 45nm
➢ Clock Frequency : 750 MHz
➢ Chip Size : 17mm × 17mm ;
Logic : 267.7MGate
Memory : 23Mbit



(Desk Top Machine)
・60cm × 60cm × 20cm＝7,200cm^3
・Power Consumption : 1000W(max)



TOPSTREAM™ RTRT
(73 Heterogeneous Manycore)



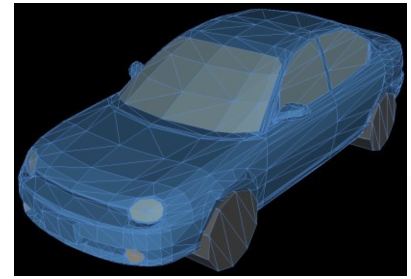
(A Cluster includes 9 Heterogeneous Core)

(Image Generated by Visual Simulation)

※Joint R&D with TOYOTA Moror & NIHON UNISYS

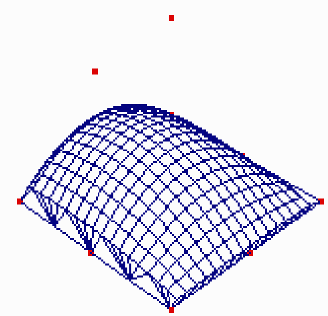
Heterogeneous Many Core : 0.88TFLOPS/W

Heterogeneous Multi-Core drives Computer Graphics Paradigm Shift

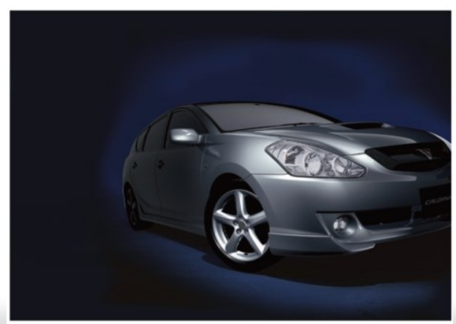


- Synthesis
  - Animations, movies, video games
  - Algorithms : Polygon based Ray Tracing
  - Computer Performance Requirement : **～ 1TFLOPS**
- Reproduction
  - Replace prototypes and samples
  - Industrial Design & Showrooms
    - Automotives, Buildings, Houses, etc.
  - Algorithms : Natural Surface based Ray Tracing, Photon Mapping
  - Computer Performance Requirement : 100's TFLOPS ～

$$S(u,v)=\sum_{i=0}^{n}\sum_{j=0}^{m}B_i^n(u)B_j^m(v)P_{ij}$$





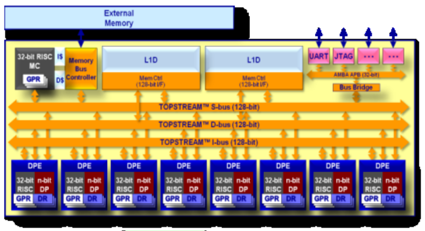

Architecture-Algorithm Co-Design for Application Domain Specific Computing

Performance vs. Power Optimization

Performance = $f \times$ IPC

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

TOPSTREAM™ Architecture



Reduces Walls
- ILP Wall
- Memory Wall
- Power Wall

TOPSTREAM™ Platform IP

- Patents
- TS-ISIM (ISS)
- TOPS_Lib (RTL)
- Base HW (RTL)

Requirements
88TFLOPS@100W
750MHz,

Application
"Ray Tracing"
"Photon Mapping"

Architecture-Algorithm Co-Design

- Analysis
- Architectural Optimization
- Algorithmic Optimization
- Performance & Power Simulation

HW Spec | System Spec | SW Spec

HW/SW Co-Design

- HW design
- SW design
- HW/SW Co-Verification

SW Partitioning

Legacy Software (Sequential)

Splitting

Fine Grain Tasks

Grouping

Process Elements

Co-operating

Distributed Processing KPN model

Mapping

Distributed Processing Multi-Core

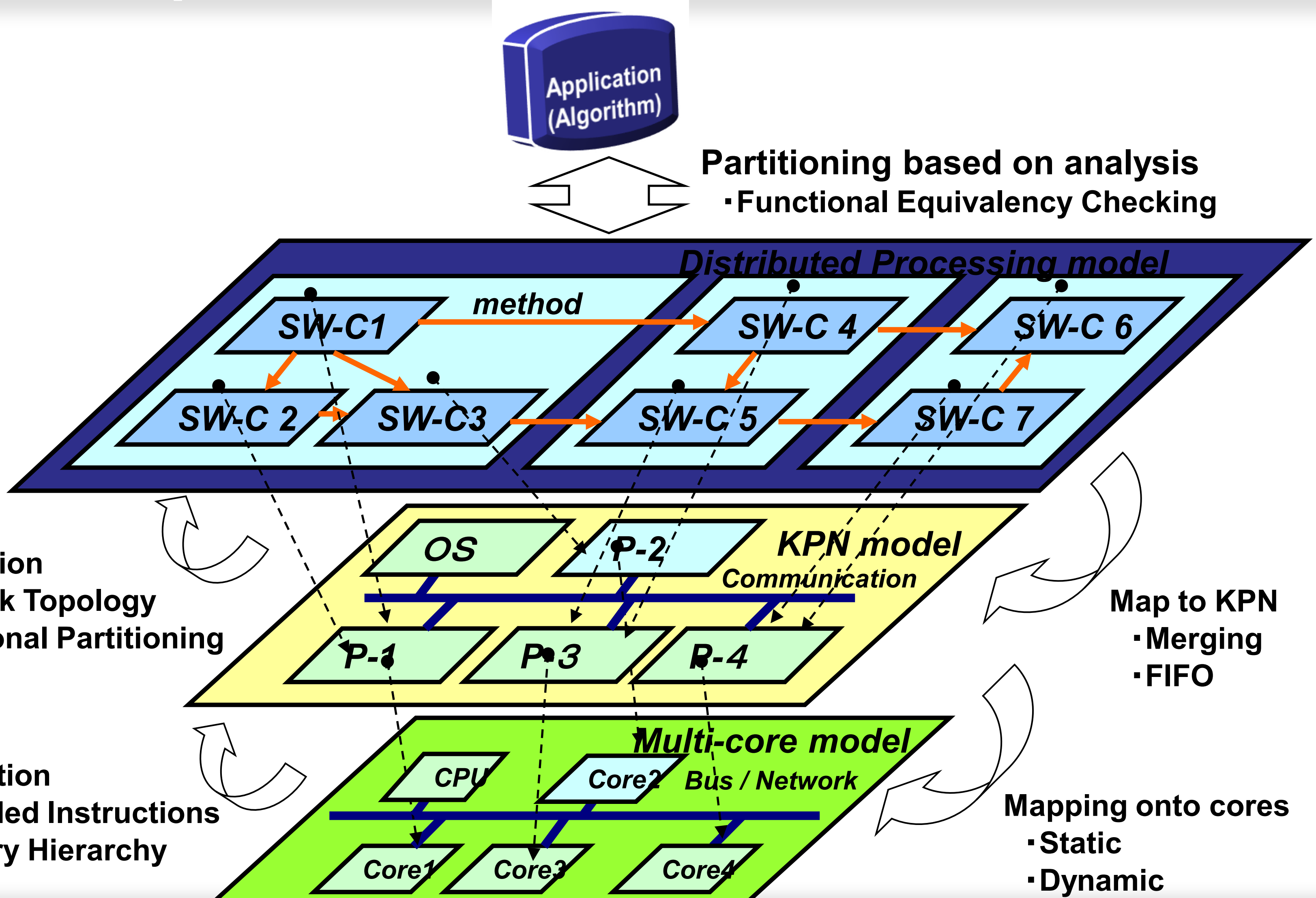
TOPS

Architecture & Algorithm Co-Design

Optimizations go Bidirectional

Partitioning based on analysis

- Functional Equivalency Checking



Optimization

- Network Topology
- Functional Partitioning

Map to KPN

- Merging
- FIFO

Optimization

- Extended Instructions
- Memory Hierarchy
- etc.

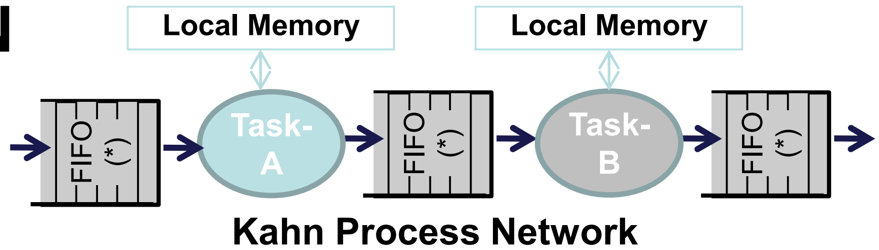
Mapping onto cores

- Static
- Dynamic

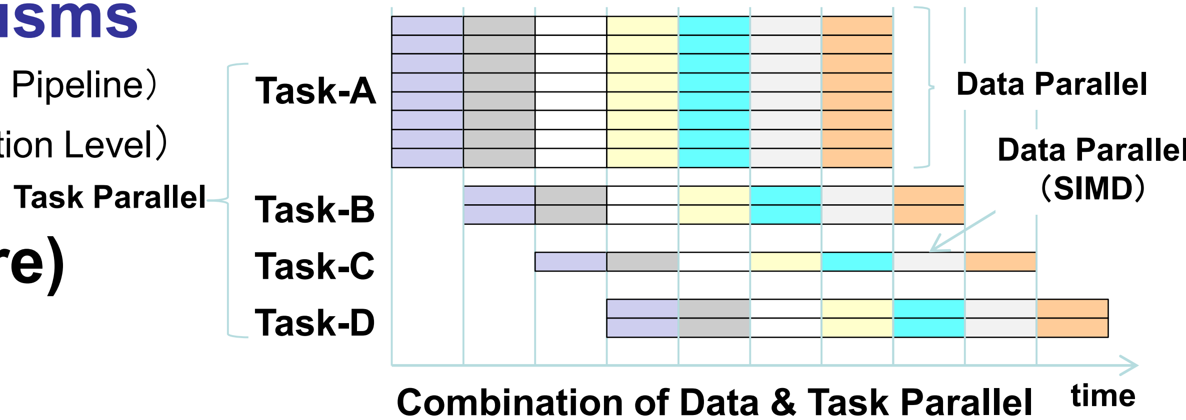
Can expect more than 10 X of Performance Improvement

System Level Architecture

- **Distributed Processing with KPN**
  - Non-Shared Memory Processes
  - Zero-Overhead Message Passing Mechanism （ZOMP）

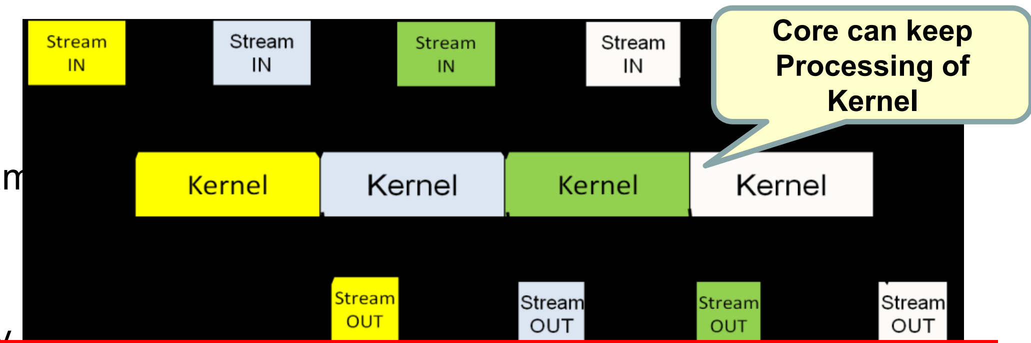


- **Combination of Parallelisms**
  - Distributed Parallel Processing（Task、Pipeline）
  - Data Parallelism（High-Level、Instruction Level）



- **Stream Processing (Core)**
  - Kernel
  - Stream-In (Read Message)
  - Stream-Out (Write Message)

- **Optimization of Core**
  - Support Stream Processing：background Stream
  - Complex Inst：Reduction of Kernel cycle
  - FIFO support mechanism
  - Reduction of energy for instruction / data supply



Combination of Parallelisms, Stream Processing, and ASIP

Basic concept of stream processing:
“Maximize processor efficiency”

Conventional processing

Mixed (Processing / Load / Store)

Stream processing (on a conventional processor)

Stream IN	Kernel	Stream OUT	Stream IN	Kernel	Stream OUT

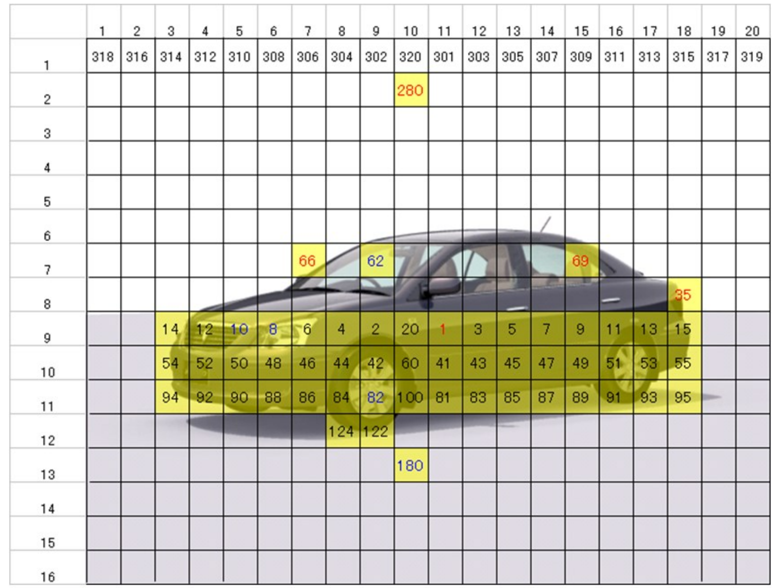
Careful Scheduling of Stream-In and Stream-Out

Real-Time Ray Tracing

Performance Analysis Result Examples

- ❖ Performance Requirement Analysis
  - ❖ Performance / Frame
  - ❖ Performance / Area
  - ❖ Performance / Ray
  - ❖ Performance / Ray Type
  - ❖ Performance / Function
    - ❖ Computation / Function
    - ❖ Memmory Access / Function

Functional Block	Processing Time		Operations (Count)					Memory Access (Count)			
	ticks	%	Integer	FP	Branch	Transfer	Others	Load	%	Store	%
Ray Generation	4010	2.5%	2858	5118	1966	8383	1.291666	1535	3.1%	2121	4.4%
			15.6%	27.9%	10.7%	45.7%	0.0%				
Space Check	3757	2.4%	938	843	795	487	0	1715	3.5%	1811	3.7%
			30.6%	27.5%	[illegible]	15.9%	0.0%				
Voxel Traverse	35543	22.5%	14910	15780	8076	3952	0	16754	34.4%	24698	50.7%
			34.9%	36.9%	18.9%	[illegible]	0.0%				
Bbox Check	21427	13.6%	4983	9500	6281	10203	0	13286	27.3%	13356	27.4%
			16.1%	30.7%	20.3%	32.9%	0.0%				
Surf Intersect	85403	54.2%	5253	9778	6156	[illegible]	0	838	1.7%	1162	2.4%
			16.6%	31.0%	19.5%	32.9%	0.0%				
Poly Intersect	997	0.6%	496	354	[illegible]	27	0	583	1.2%	813	1.7%
			38.1%	27.2%	[illegible]	2.0%	0.0%				
Trim Rough	1284	0.8%	37	169	59	193	0	118	0.2%	180	0.4%
			8.1%	36.9%	12.9%	42.1%	0.0%				
Trim Check	2582	1.6%	613	706	575	546	0	846	1.7%	1162	2.4%
			25.1%	28.9%	23.6%	22.4%	0.0%				
Depth Test	358	0.2%	144	114	118	32	0	[illegible]	0.7%	494	1.0%
			35.3%	27.9%	28.9%	7.9%	0				
Haikei Intersect	283	0.2%	410	354	330	211	0	268	0.5%	394	0.8%
			31.4%	27.1%	25.3%	16.2%	0				
Create Node	2070	1.3%	1241	1102	899	360	0.625	629	1.3%	919	[illegible]
			34.4%	30.6%	25.0%	10.0%	0.0%				
Total	157714	100.0%	31883	43817	25681	34777	2	36890		47110	



Area Based Processing

Load distribution by Ray types

Ray Type	Counts	Average[s]	Peak [s]	ratio
All Ray type	3621990	0.0002510	0.941605	3751
Primary Ray	1399163	0.0004463	0.941605	2110
Reflective Ray	558323	0.0002523	0.388640	1541
Permeable Ray	160701	0.0002167	0.344437	1590
Shadow Ray	1479873	0.0000720	0.740380	1027[illegible]

・Memory Allocation
・Memory Hierarchy

・Processing unit

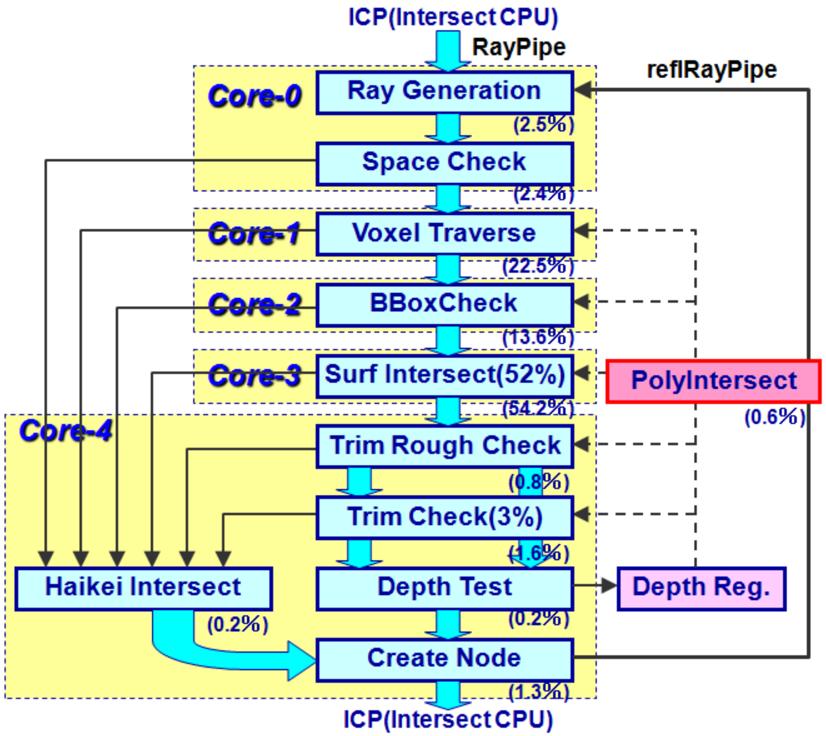
・Special Instruction

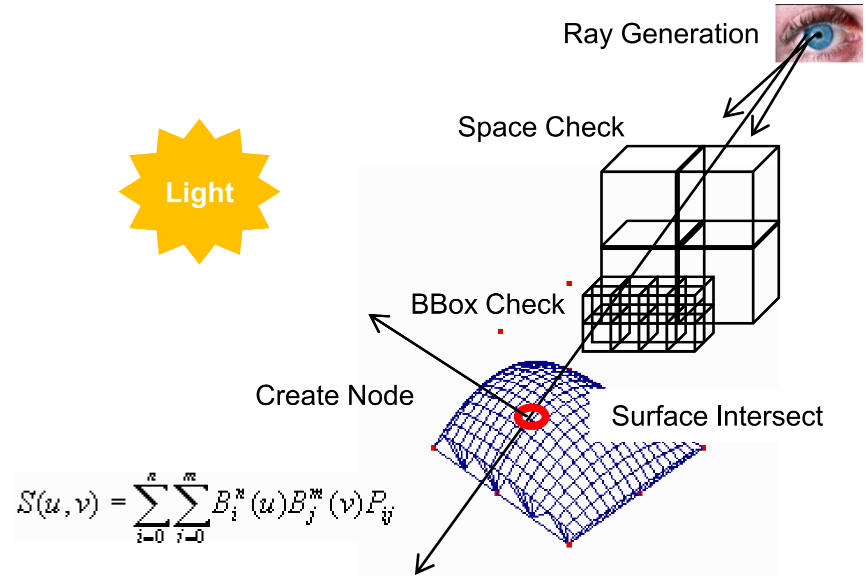
Big Challenge was Dynamic Huge Load Changes (max. 3751 x)

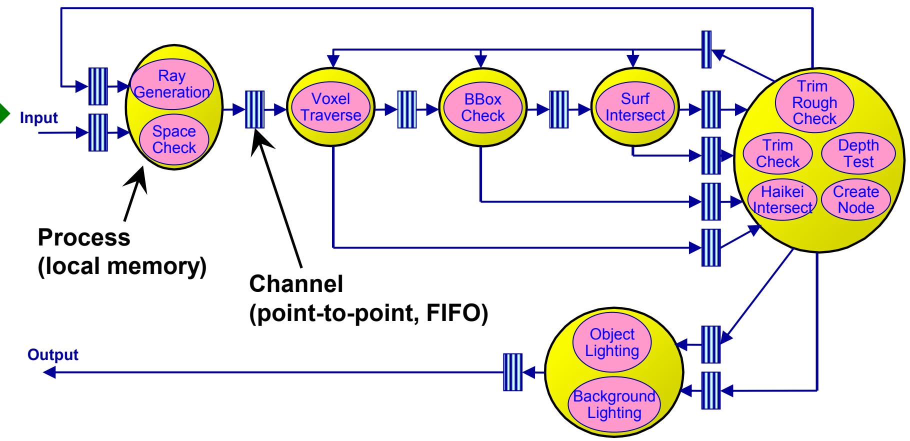
Partitioning and KPN model for Ray Tracing

Rendering on Intersect with Color and Lighting

- Partitioning of Ray Tracing Process
  - Based on processing flow : Functional partitioning







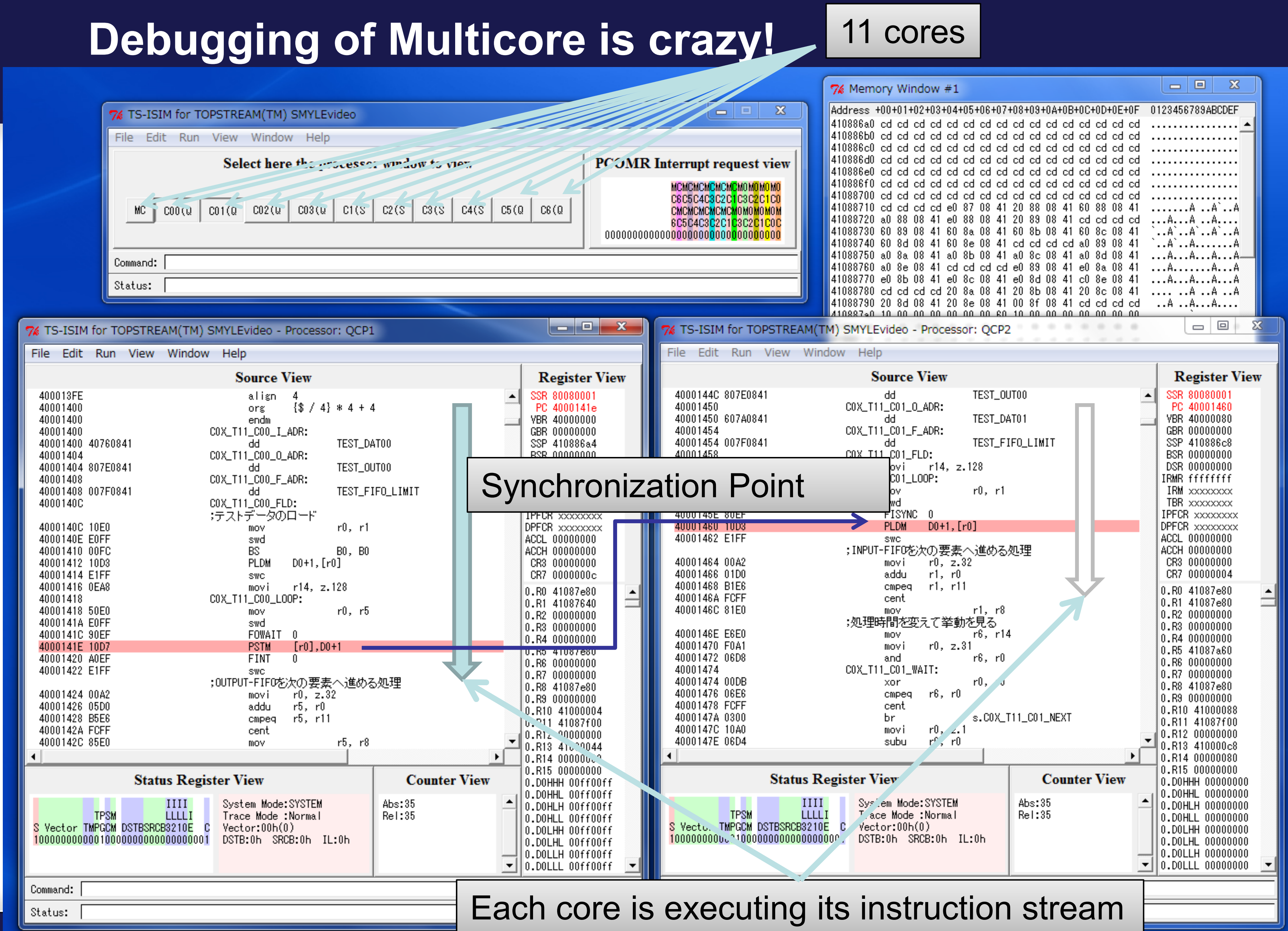
Kahn Process Network (KPN) model for Ray Tracing

- M

Two Levels of Functional Verification
1st Level : Each Process
2nd Level : Whole KPN

Equivalency Test with a number of Input / Output Data Set

Debugging of Multicore is crazy!



Human's nature is

- For typical engineers,
  - can follow "a single instruction stream" for debugging
  - make mistakes with "Two instruction stream"
  - No way with "Three instruction stream"

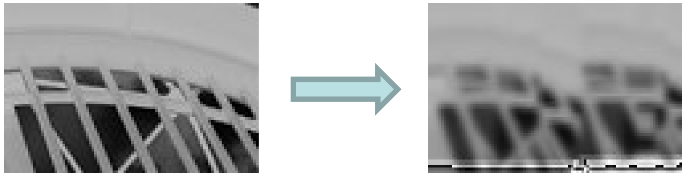
↓

Key for Multicore Debugging
Extract "One Stream" of information, and concentrate on it.

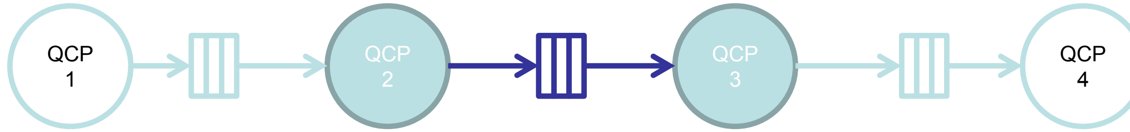
↓

Provide tools to be able to concentrate on debugging

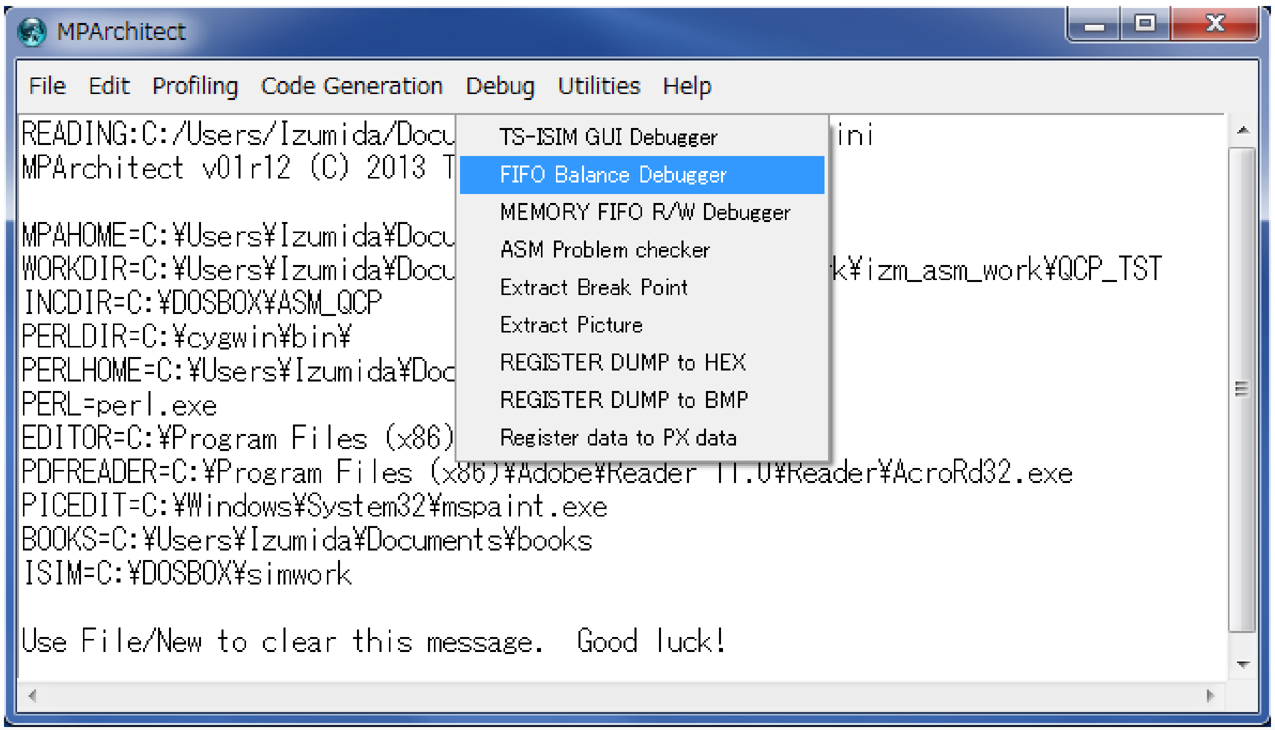
Debugging of application with KPN model



Something wrong on Filter Function

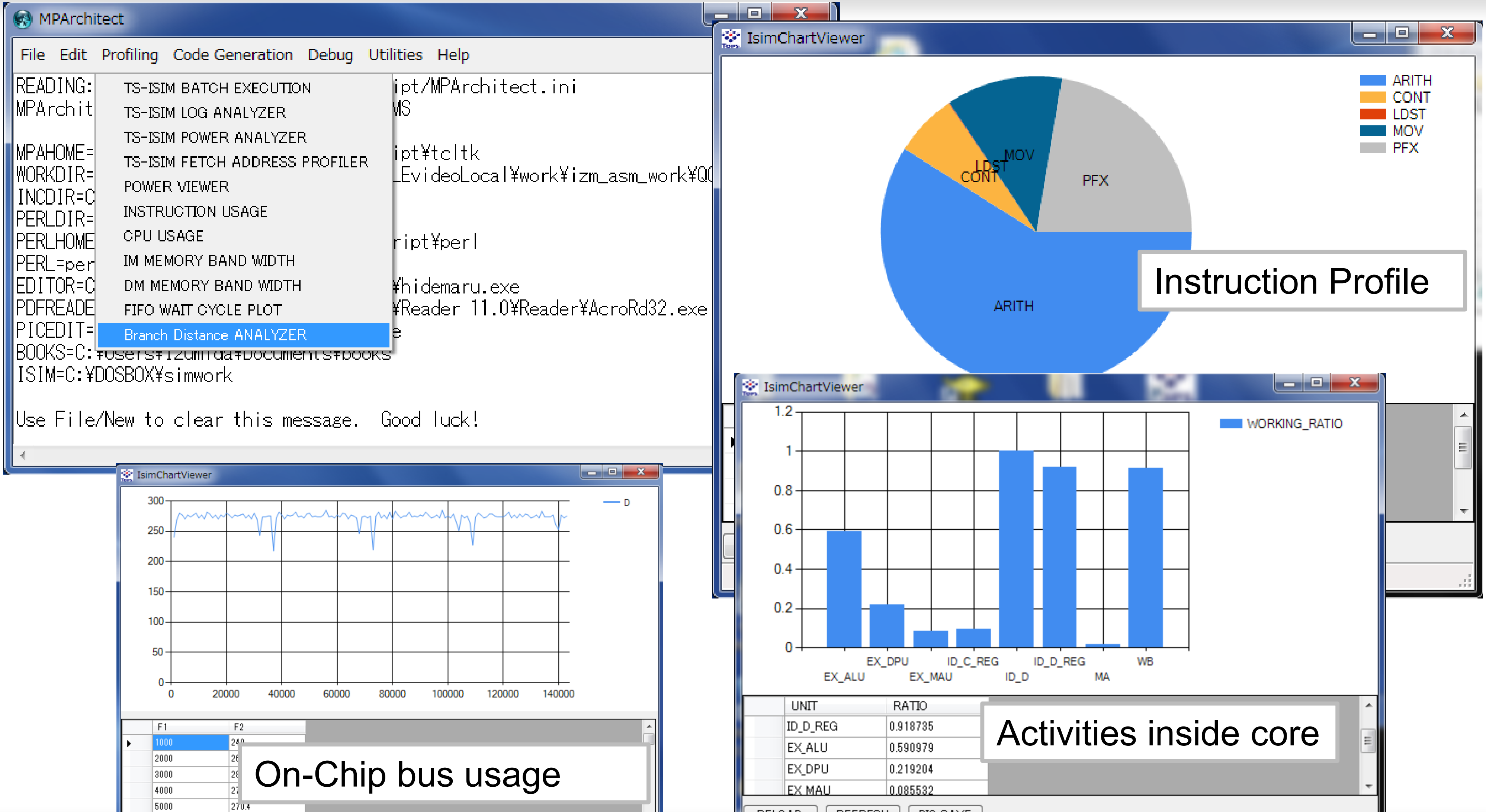


Focus on FIFO




```
105,FOWAIT_1<=QCP3
106,(1)QCP3:WRITABLE_0:41028360
107,FINT_1=QCP3
108,FISYNC_1=>QCP3
127,FOWAIT_1<=QCP2,(1)QCP3:UR_0
128,(1)QCP3:UR_0,(1)QCP2:WRITABLE_1:410192e0
135,FINT_1=QCP2,(1)QCP3:READABLE_16:410192e0
151,FINT_1=QCP3
183,FOWAIT_1<=QCP2
184,(1)QCP2:WRITABLE_19:4101e300
190,FISYNC_1=>QCP3
191,FINT_1=QCP2,(1)QCP3:READABLE_32:4101e300
207,FINT_1=QCP3
239,FOWAIT_1<=QCP2
240,(1)QCP2:WRITABLE_36:41023320
246,FISYNC_1=>QCP3
247,FINT_1=QCP2,(1)QCP3:READABLE_48:41023320
263,FINT_1=QCP3
292,FOWAIT_1<=QCP2
293,(1)QCP2:WRITABLE_52:41028340
```

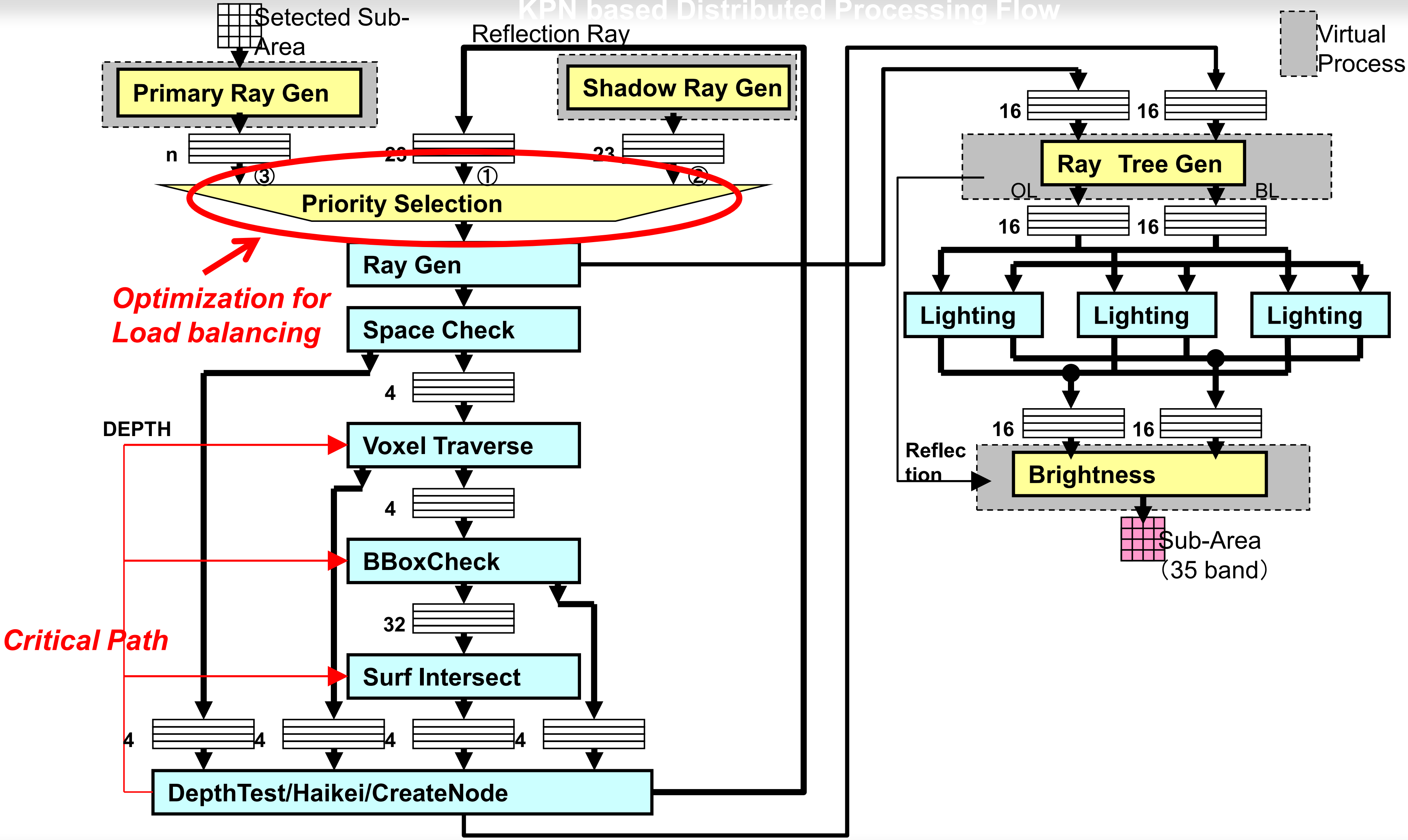
MPArchitect provides several tools



Activity monitor helps programming for Low Power

Performance Considerations
Real-Time Ray Tracing

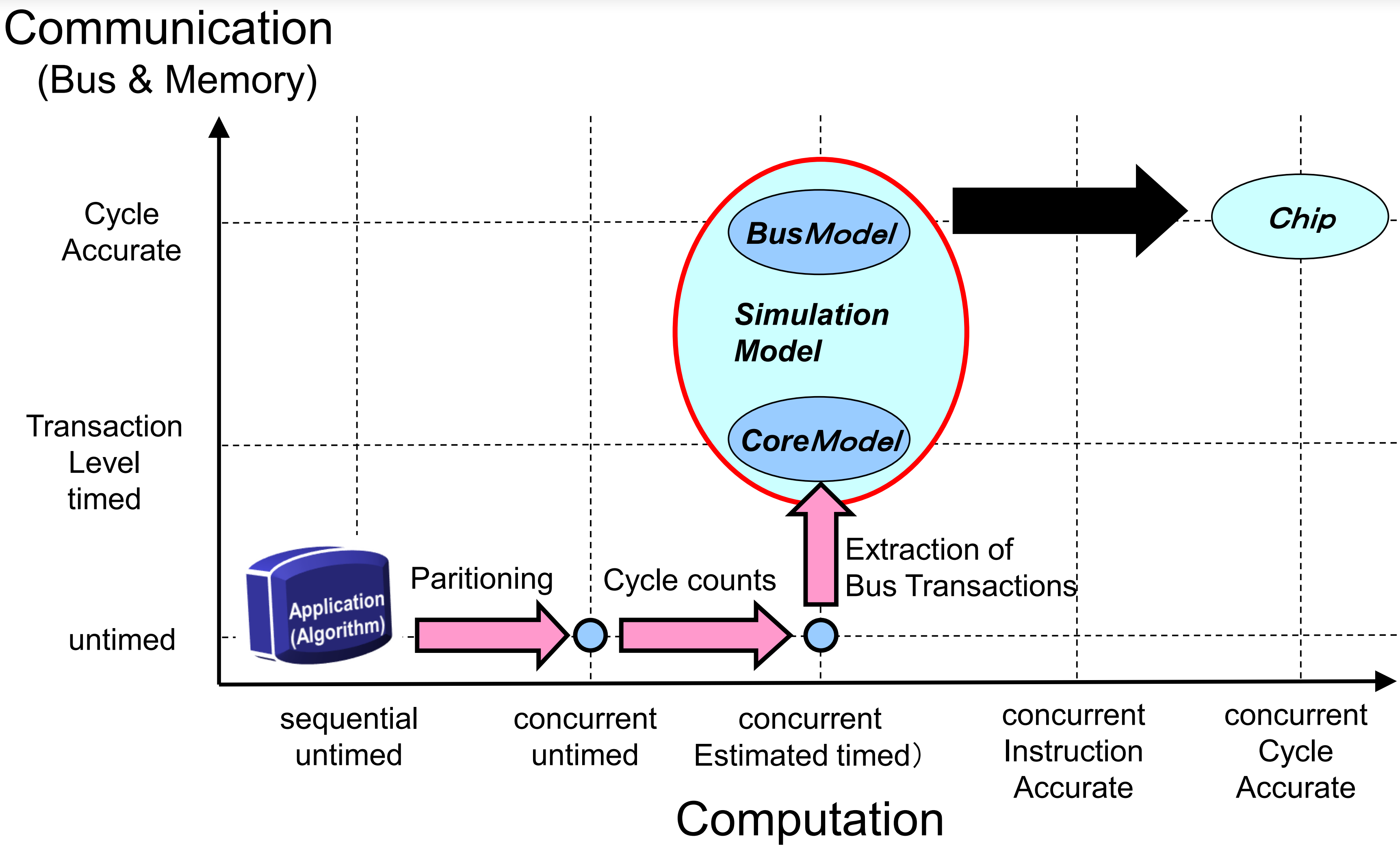
KPN based Distributed Processing Flow



Critical Loop and Buffers for Load Ballancing

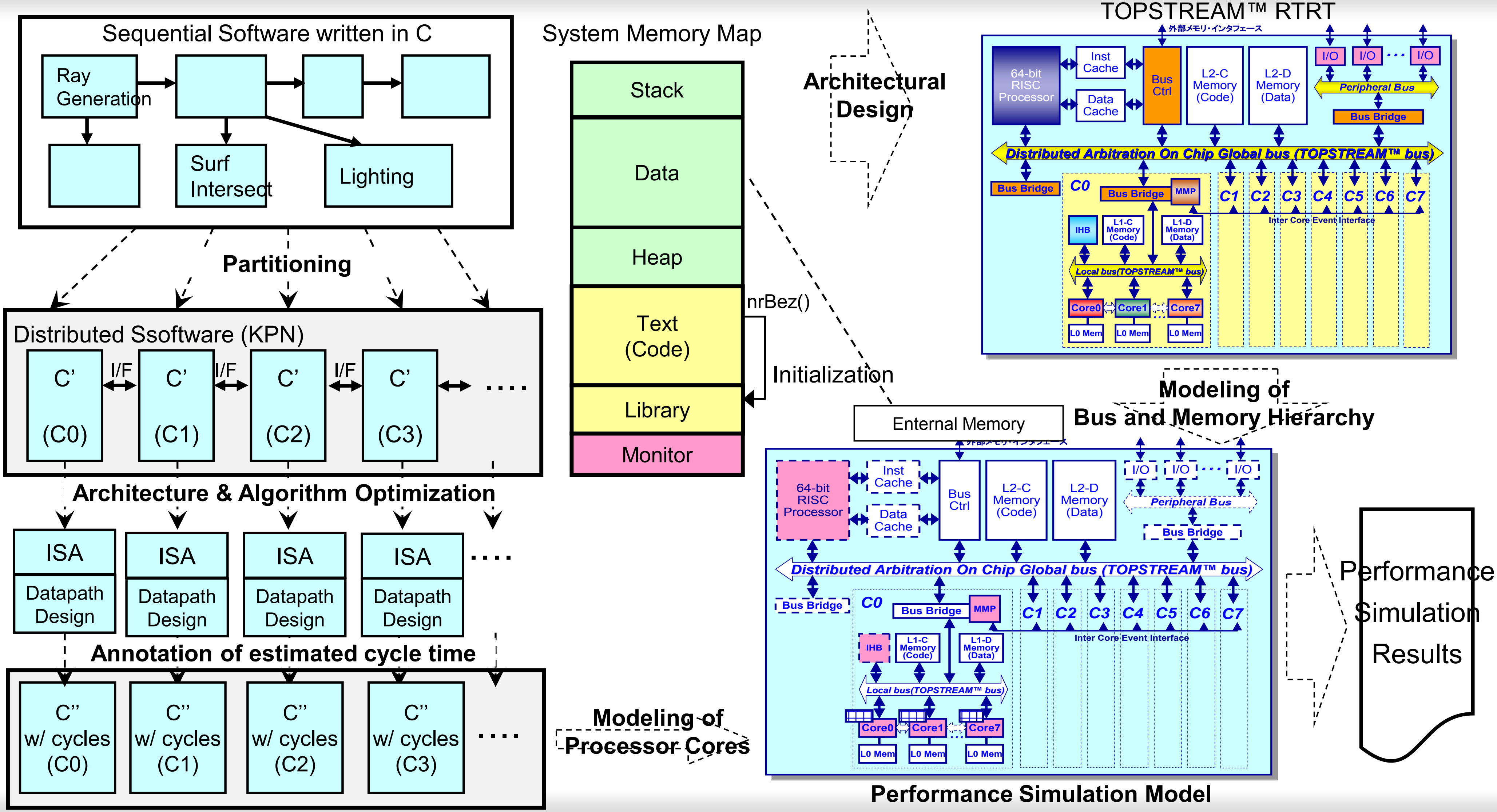
Performance Simulation

Architecture Simulation Model & Accuracy

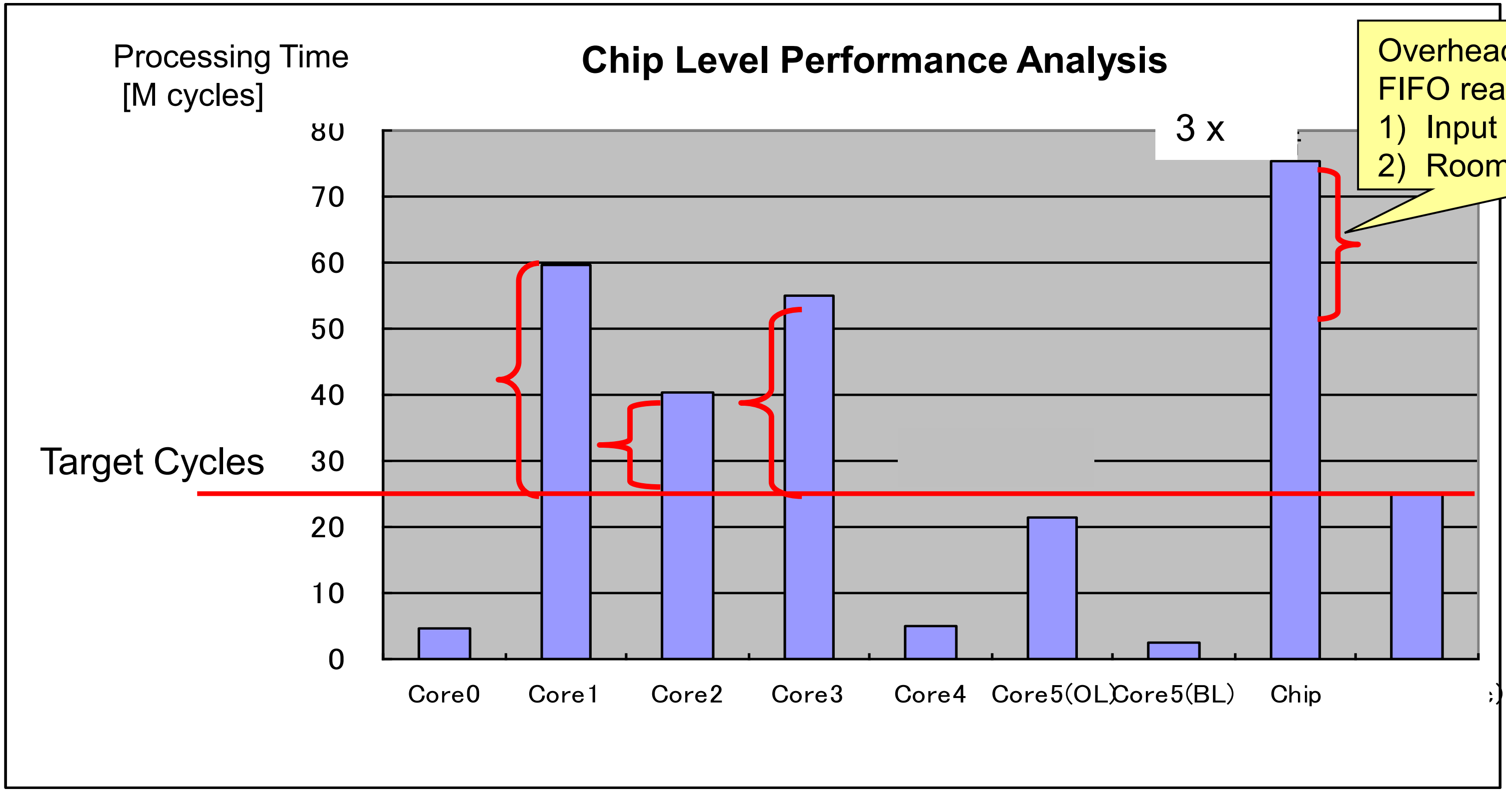


Result of Trade-Off between speed and accuracy

Hardware and Software Modeling Flow



Performance Simulation Results



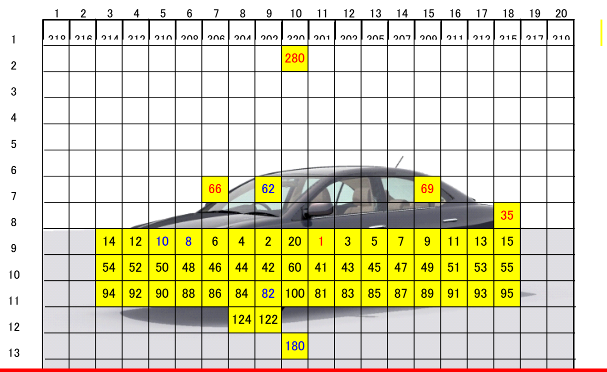
- 2 X Improvement of Performance by
  - **Optimizing FIFO depth by SW**
  - **Changing priority for critical loop control ; Ray Generation**
  - **Extending Application Specific Instructions**

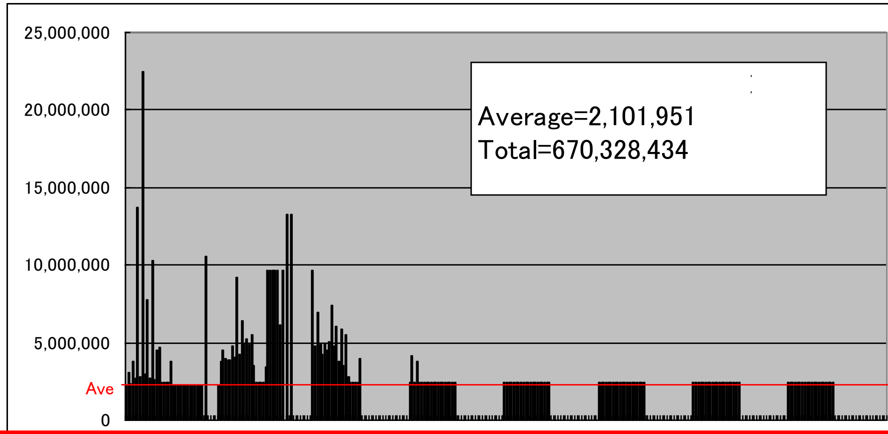
Figured out where to improve performance

Performance Simulation

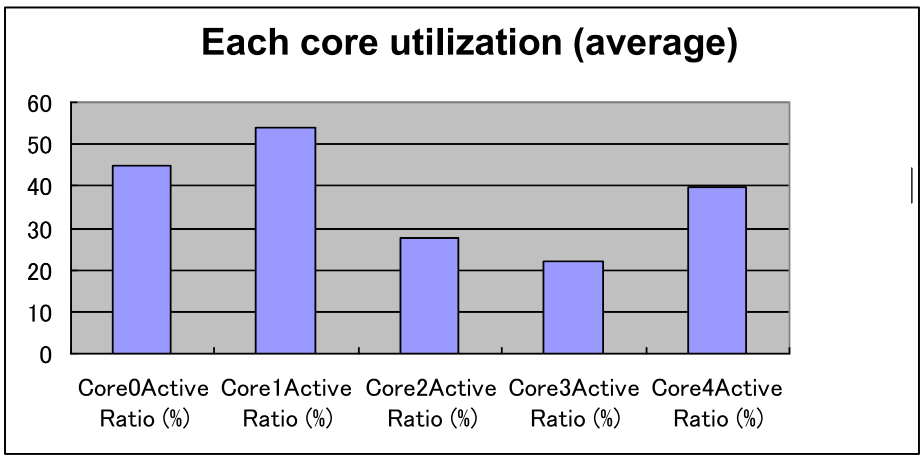
Simulation results（Cycles/Area）

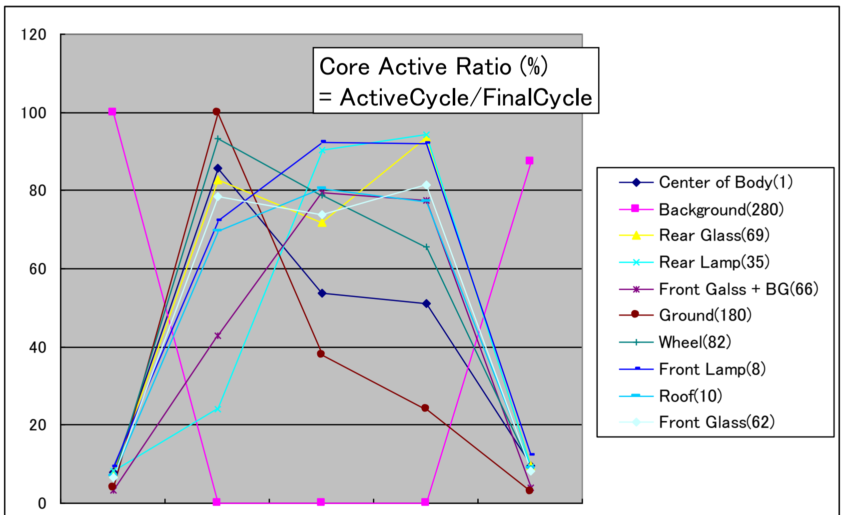
Area based processing





Better Load balancing : max. 3751 x ⇒ max. 9 x

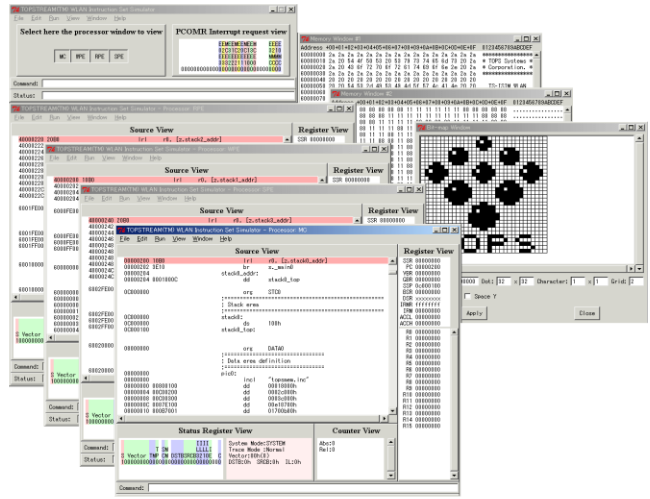




Average core utilization : 35%

Fast Verification Environment

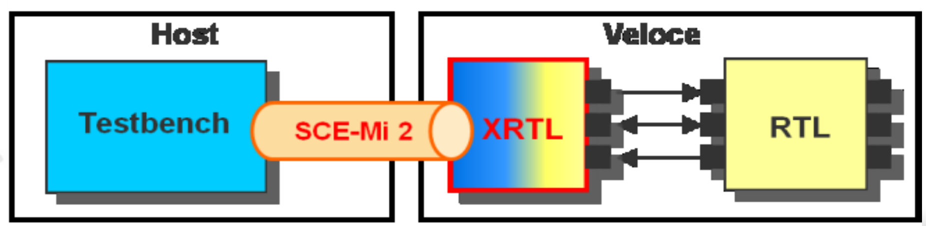
- Software Verification
  - TS-ISIM : Multicore ISS @ 17MIPS
  - Virtual COM port
- System Verification
  - Standard Methodology : OVM
  - RTL Simulator : Questa
    - Test Bench : Virtual System
  - Hardware Emulator : Veloce
    A. Stand Alone : Core Level HW/SW Co-Verification
    B. Virtual System : Multicore System
      ◆ Veloce (Emulator) connection to WS through TBX



TS-ISIM (in house tool)

Supported by

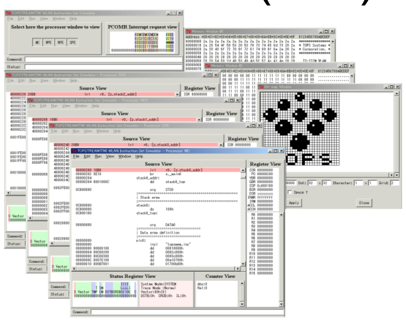
Mentor Graphics



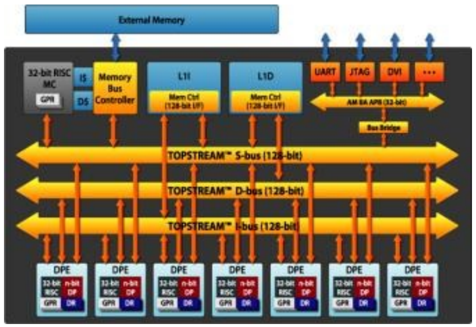
ISS vs. RTL Comparison at Instruction Level

- Objective
  - Speed-up Debugging of Applications at System Level

TS-ISIM(ISS)



RTL



Instruction Trace

FA/PFA, Code, Rs1, Rs2, Rd, PSW, MRW, Addr/PAddr, Data
FA/PFA, Code, Rs1, Rs2, Rd, PSW
FA/PFA, Code, Rs1, Rs2, Rd, PSW
.....

Compare by Inst.

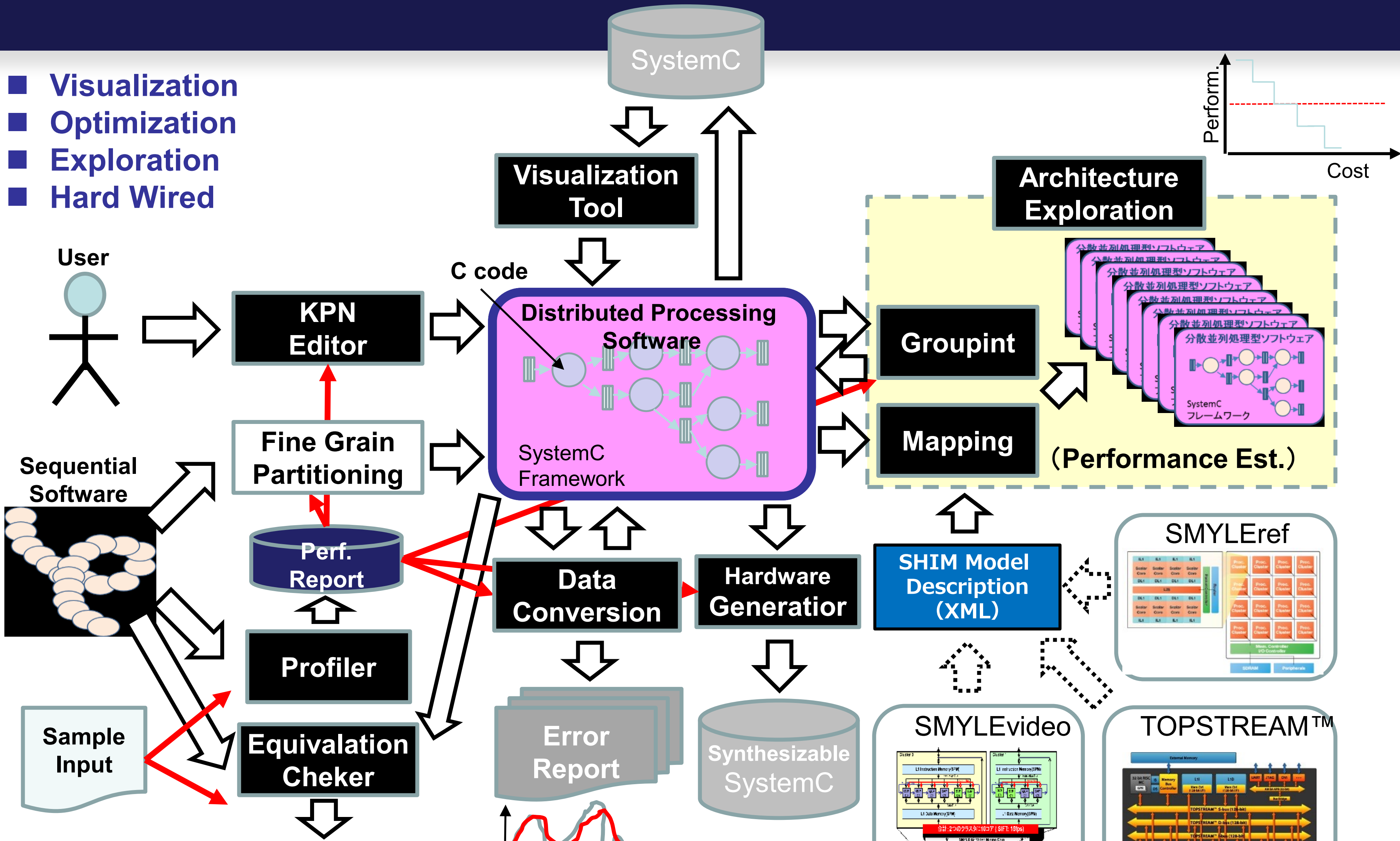
Instruction Trace

FA/PFA, Code, Rs1, Rs2, Rd, PSW, MRW, Addr/PAddr, Data
FA/PFA, Code, Rs1, Rs2, Rd, PSW
FA/PFA, Code, Rs1, Rs2, Rd, PSW
.....

Bug is inside an instruction
(1000 x performance with Hardware Emulator)

Challenges to provide tool sets for KPN programming

- Visualization
- Optimization
- Exploration
- Hard Wired



Goal : Exploration of best SW and best HW architecture

＜様々なマルチ・メニーコア＞

TOPS

Thank you for your attention!