A Scalable GPU Architecture based on Dynamically Reconfigurable Embedded Processor

Won-Jong Lee, Sang-Oak Woo, Kwon-Taek Kwon, Sung-Jin Son, Kyoung-June Min,
Gyeong-Ja Jang, Choong-Hun Lee, Seok-Yoon Jung, Chan-Min Park, Shi-Hwa Lee

SAIT, SAMSUNG ELECTRONICS Co., Ltd.

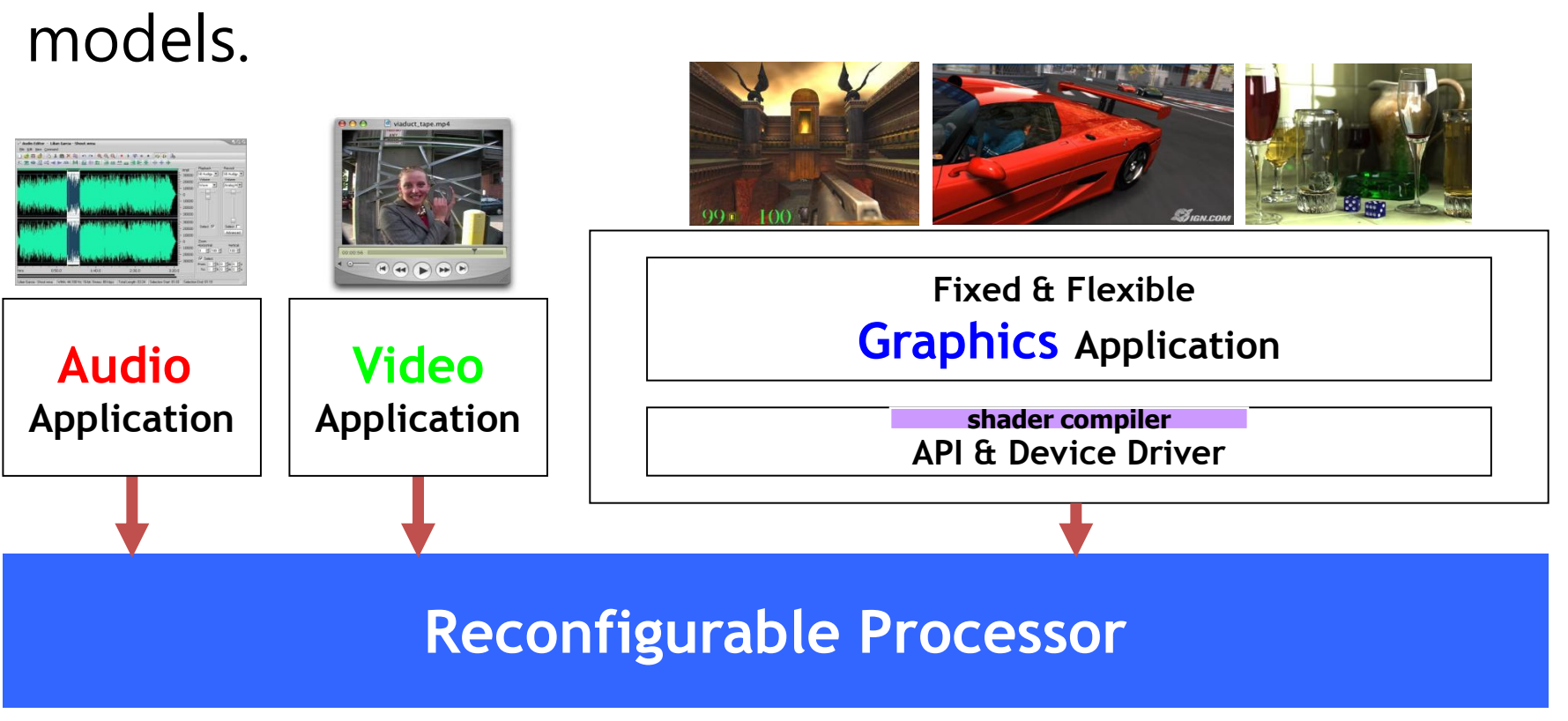
HIGH-PERFORMANCE GRAPHICS
VANCOUVER, CANADA AUGUST 5-7, 2011

Motivation

- Mobile devices such as hand-helds, smart phones, digital multimedia broadcasting (DMB) terminals, PDAs, tablet PCs and portable gaming consoles are widely used all over the world.
- The market has accepted mobile devices, which are multifunctional devices that will, in future, take the place of many portable, consumer electronic devices, such as cameras and music players. This requires an application processor that can balance the different performance requirements of these functions with energy efficiency.
- The increasing cost of ASIC has been driving designers to choose more flexible solutions, as new chip architectures should deliver high performance while maintaining low power consumption, area, and costs in shorter time-to-market environments. As a result, **reconfigurable processors (RPs)** have become increasingly important in recent years.

  ex) Reconfigurable Processor (RP) for multi-media applications:
  H.264/AVC [1,2], Video [3], Audio [4], Graphics [5]

- In this work, we present a new approach that utilizes an **RP to design a tile based GPU,** a core component of today's mobile application processors. Experimental results show that the use of RP based GPUs can be a versatile graphics solution, as they support scalable, flexible architecture models.



Audio Application

Video Application

Fixed & Flexible Graphics Application

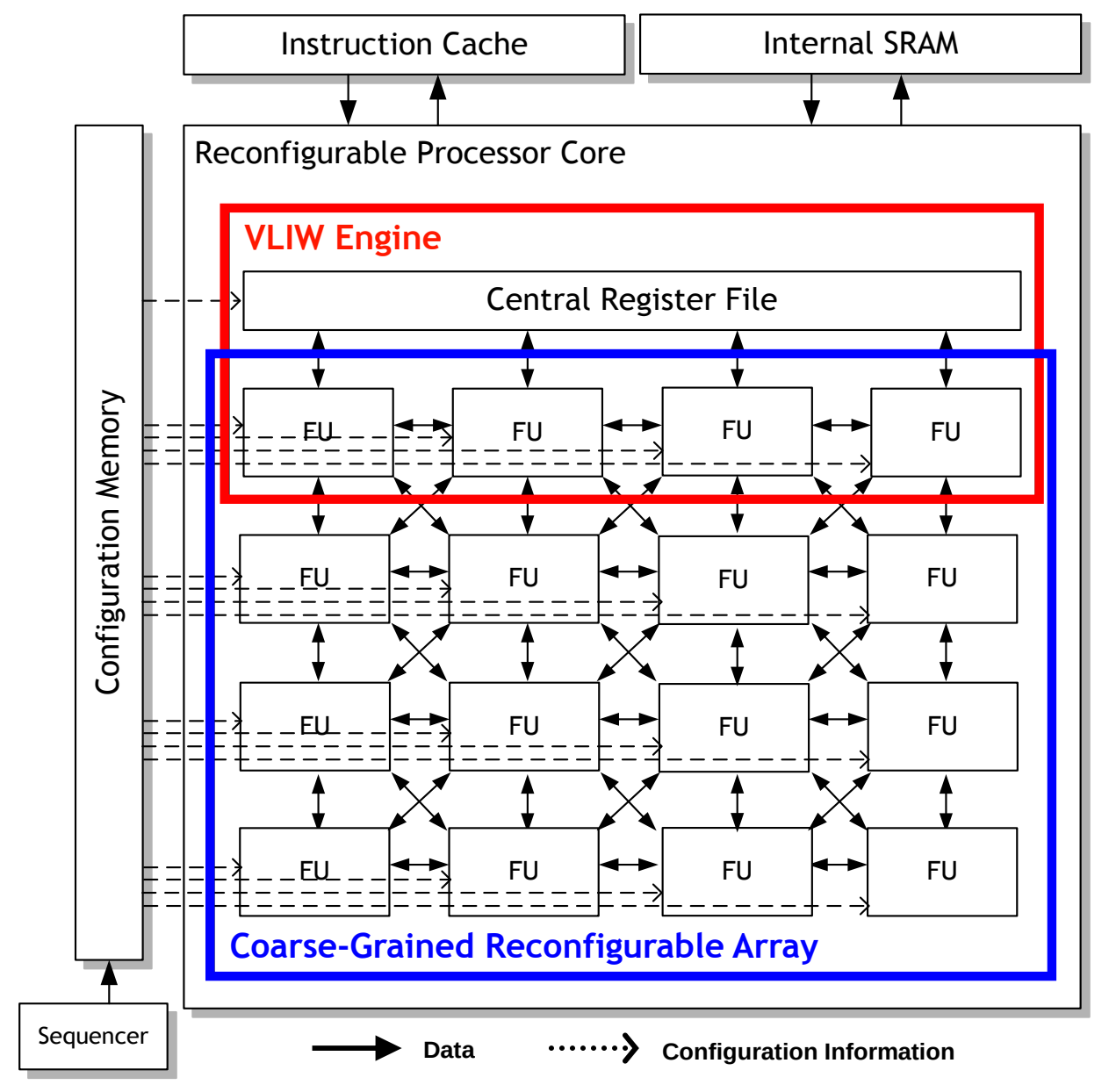
shader compiler

API & Device Driver

Reconfigurable Processor

Samsung Reconfigurable Processor

SRP System Architecture



Instruction Cache

Internal SRAM

Reconfigurable Processor Core

VLIW Engine

Central Register File

Configuration Memory

Coarse-Grained Reconfigurable Array

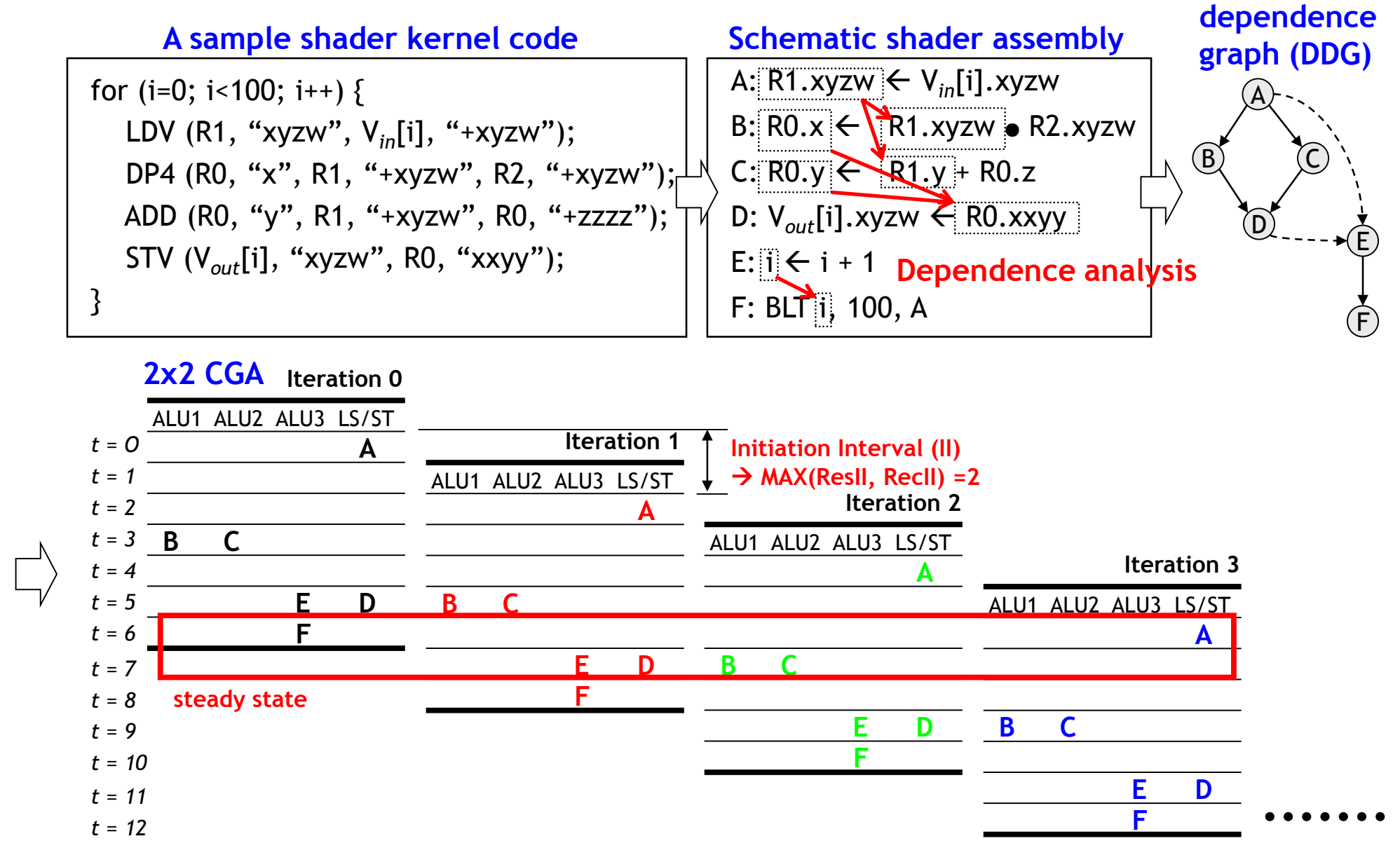
Sequencer

Data

Configuration Information

Software Pipelining via Modulo Scheduling :

Assumption: 2x2 FUs (3 ALU, 1 LD/ST), latencies (LDV=3, DP4=2, others=1)



A sample shader kernel code

```
for (i=0; i<100; i++) {
    LDV (R1, "xyzw", V_in[i], "+xyzw");
    DP4 (R0, "x", R1, "+xyzw", R2, "+xyzw");
    ADD (R0, "y", R1, "+xyzw", R0, "+zzzz");
    STV (V_out[i], "xyzw", R0, "xxyy");
}
```

Schematic shader assembly

```
A: R1.xyzw ← V_in[i].xyzw
B: R0.x ← R1.xyzw • R2.xyzw
C: R0.y ← R1.y + R0.z
D: V_out[i].xyzw ← R0.xxyy
E: i ← i + 1
F: BLT i, 100, A
```

Dependence analysis

Data dependence graph (DDG)

2x2 CGA

Iteration 0

Iteration 1

Iteration 2

Iteration 3

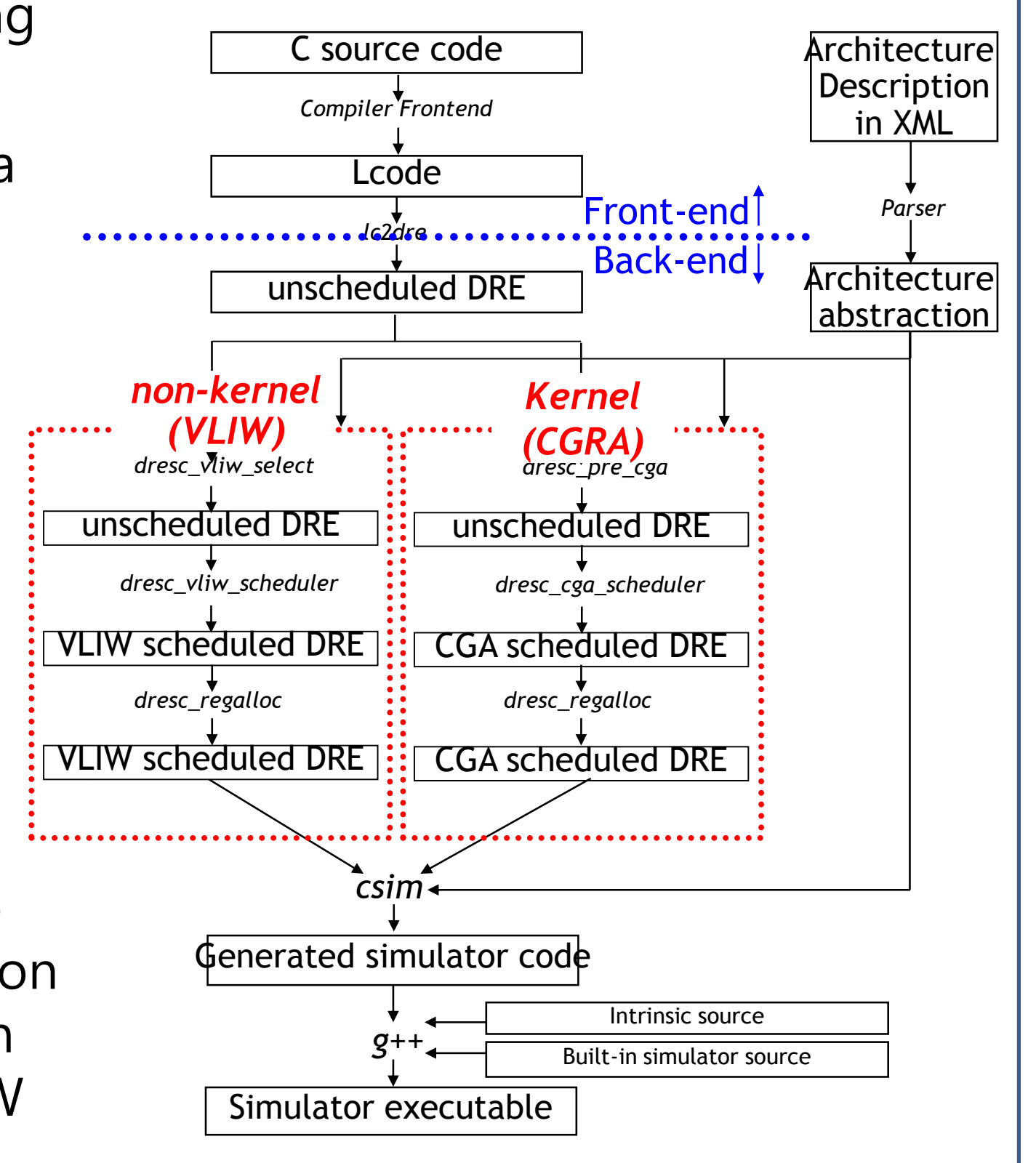
ALU1 ALU2 ALU3 LS/ST

Initiation Interval (II) → MAX(ResII, RecII) =2

steady state

- The **Samsung Reconfigurable Processor(SRP)** is a proprietary low-power DSP core developed by Samsung Electronics Ltd., as an elaboration on ADRES [6].
- The SRP is a flexible architecture template that allows a designer to easily generate different instances by specifying different configurations in the target architecture.
- The SRP includes a tightly coupled VLIW engine and a coarse grained reconfigurable array (CGRA). The VLIW engine is useful for general purpose computations such as function invocation and branch selection. The CGRA makes full use of the **software pipeline technique via modulo scheduling [7]** to allow loop acceleration.
- Consequently, the CGRA exhibits a high IPC rate, up to the maximum number of FU arrays. During the execution phase, a central pointer allows dynamic reconfiguration of the FUs within a given cycle. The integration of VLIW and CGRA allows the SRP to more effectively exploit instruction- and loop-level parallelism.

SRP tool-chain/SDK



C source code

Compiler Frontend

Lcode

lc2dre

Front-end

Back-end

unscheduled DRE

Architecture Description in XML

Parser

Architecture abstraction

non-kernel (VLIW)

dresc_vliw_select

unscheduled DRE

dresc_vliw_scheduler

VLIW scheduled DRE

dresc_regalloc

VLIW scheduled DRE

Kernel (CGRA)

dresc_pre_cga

unscheduled DRE

dresc_cga_scheduler

CGA scheduled DRE

dresc_regalloc

CGA scheduled DRE

csim

Generated simulator code

g++

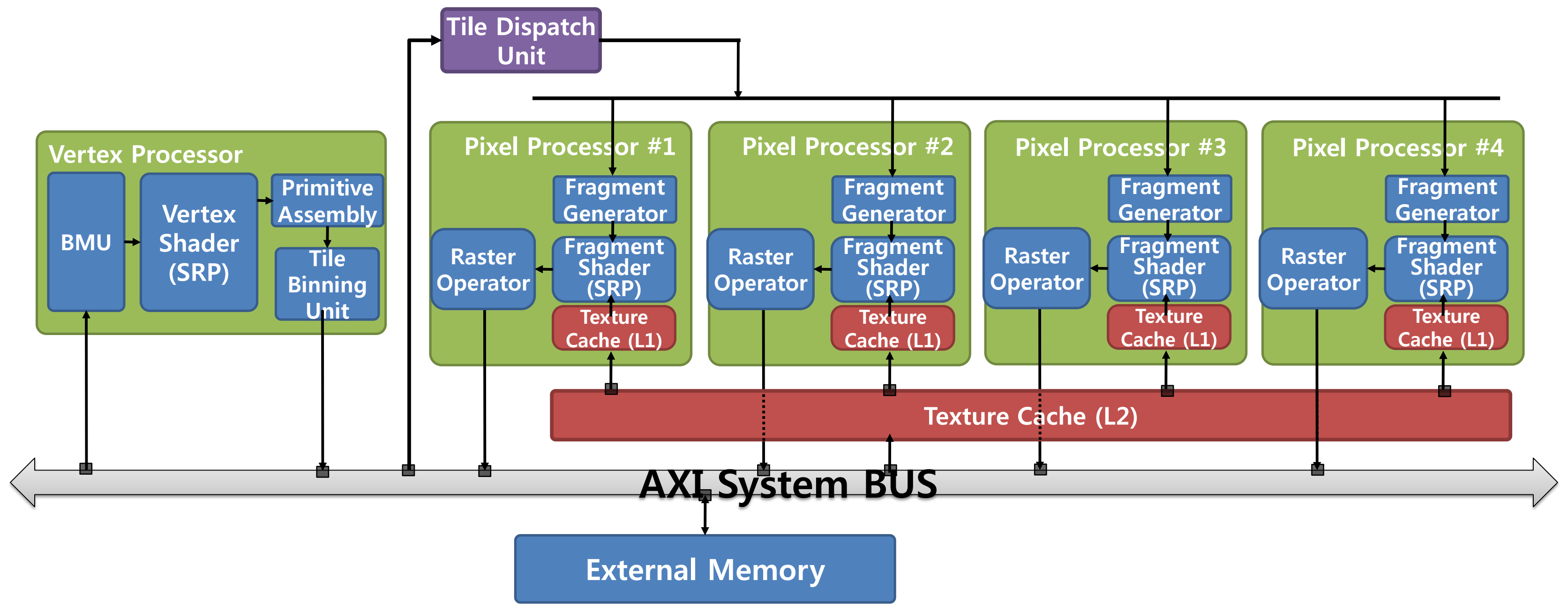
Intrinsic source

Built-in simulator source

Simulator executable

Multi-core SRP based GPU

Multi-core SRP based GPU Architecture



Tile Dispatch Unit

Vertex Processor

BMU

Vertex Shader (SRP)

Primitive Assembly

Tile Binning Unit

Pixel Processor #1

Pixel Processor #2

Pixel Processor #3

Pixel Processor #4

Fragment Generator

Raster Operator

Fragment Shader (SRP)

Texture Cache (L1)

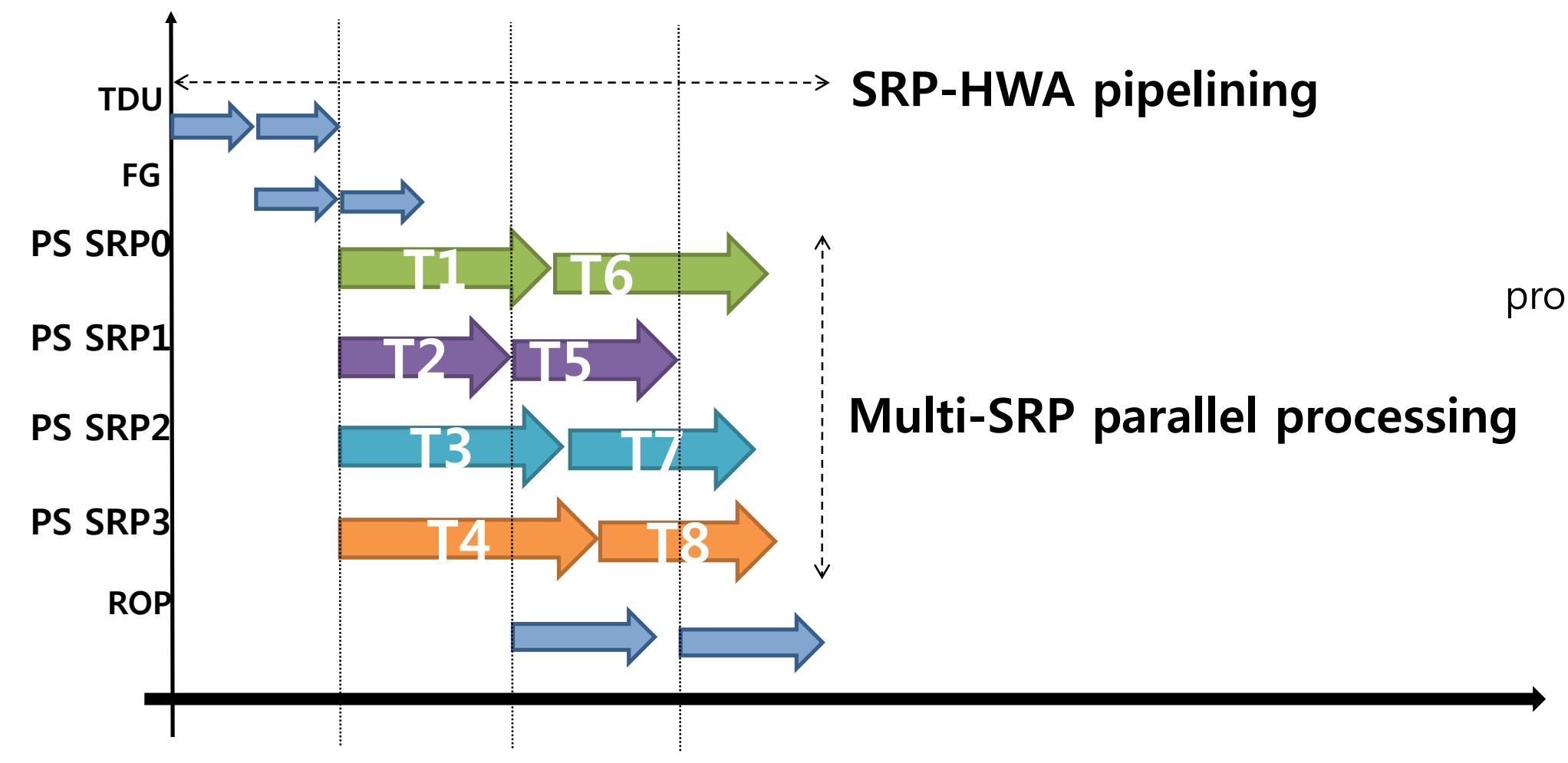
Texture Cache (L2)

AXI System BUS

External Memory

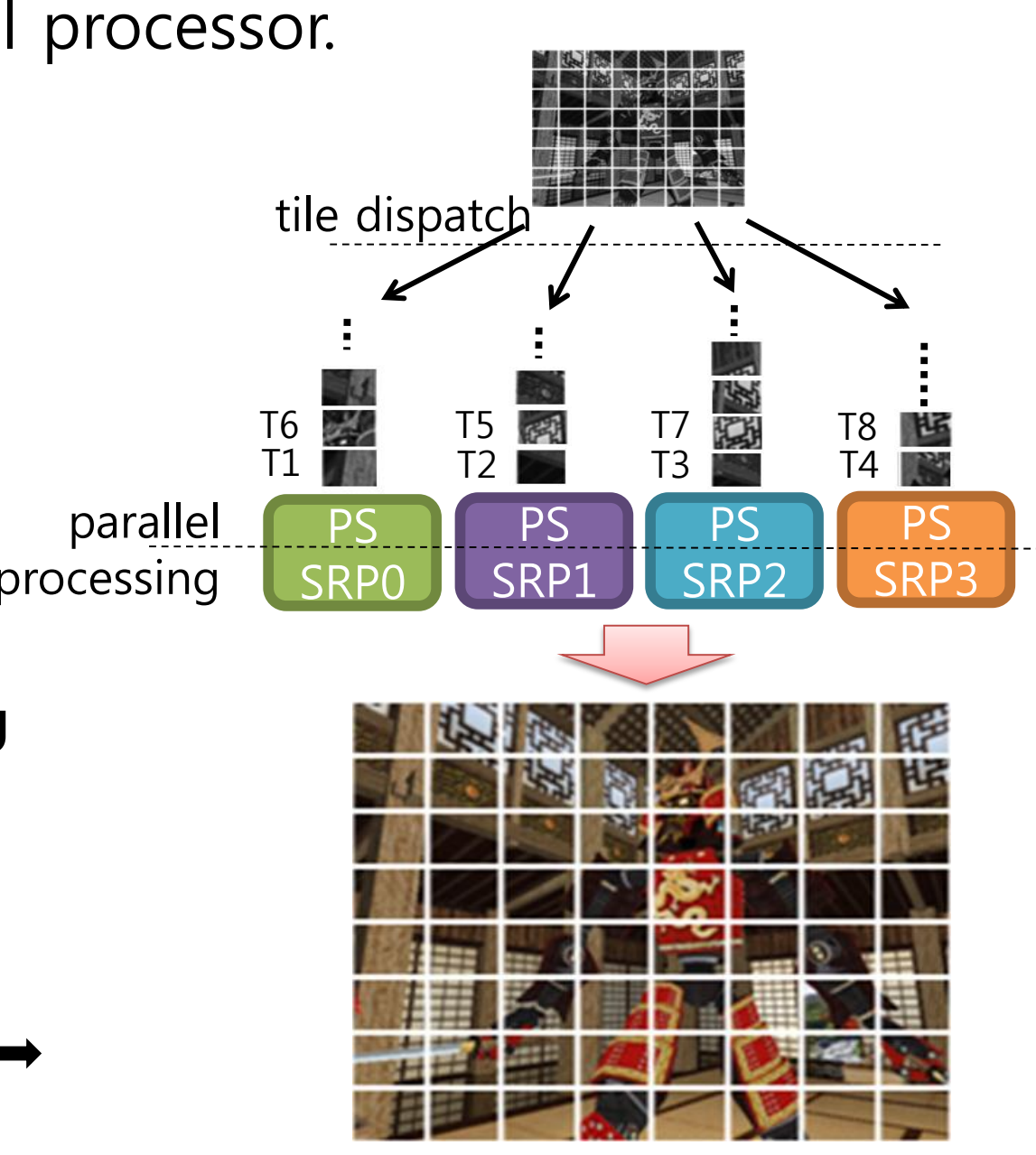
- Tile based rendering (TBR) is adopted for power efficient architecture as the commodity embedded GPU (e.g. Imagination's SGX [8] or ARM's Mali series [9]).
- Each SRP core corresponds to a vertex and a pixel shader. Other units, requiring the least programmability as special purpose hardware for TBR, are embedded inside.
- Specifically, these units are batch management units (BMUs), primitive assembly and tile binning (PATB) units, tile dispatch units (TDUs), fragment generators (FGs) and raster operators (ROPs).
- Texture units are implemented as intrinsic in the SRP core and have two-level cache architecture. All components are connected to a 64bit AXI system bus.
- Although the current architecture consists of one vertex and four pixel shaders, other configurations (e.g. multiple unified shaders) can be easily implemented using the SRP's reconfigurable feature.
- **The SRP and the hardware units are executed in a fully pipelined manner** to exploit the high throughput rate of this arrangement.
- **Parallel Tile based Rendering with Load Balancing**
  - TDU was designed for **load-balanced parallel rendering**.
  - The TDU reads the triangles (tiles) from SDRAM in a swizzled order (e.g. space filling curve) and dispatches them to an idle pixel processor.

Parallel Tile based Rendering



SRP-HWA pipelining

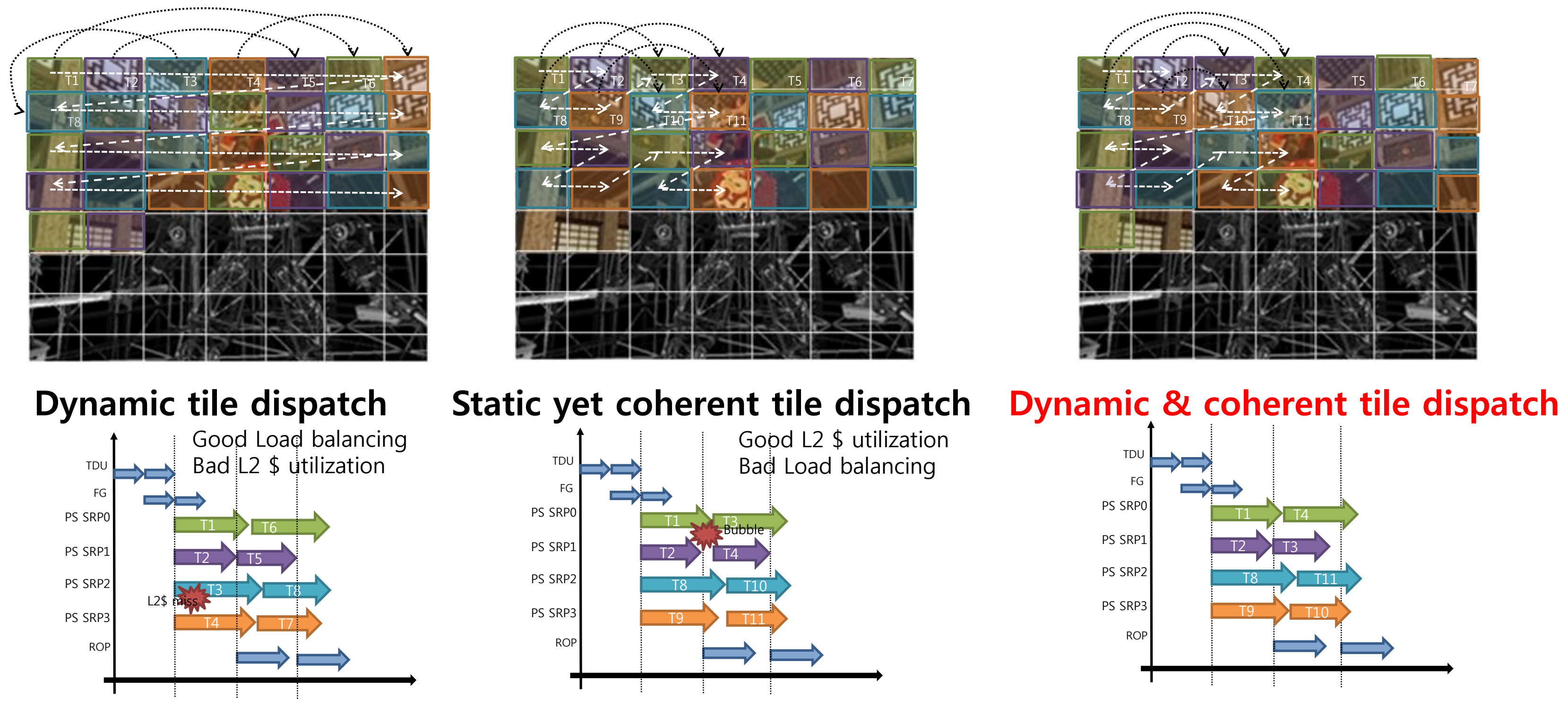
Multi-SRP parallel processing



tile dispatch

parallel processing

  - This dynamic coherent scheme allows our GPU to achieve both **load balancing** and **superior L2 cache utilization**.



Dynamic tile dispatch

Good Load balancing
Bad L2 $ utilization

Static yet coherent tile dispatch

Good L2 $ utilization
Bad Load balancing

Dynamic & coherent tile dispatch

Results

Implementation

- Our GPU is verified and evaluated by examining its performance during cycle-accurate simulation, verilog RTL simulation, and FPGA targeting. The SRP and the hardware units were synthesized to perform targeting on a Xilinx Virtex5 LX330 FPGA board running at 25 MHz with single port SDRAM modules and an LCD screen.
- Because of size limitations, the current FPGA implementation could equip itself with only two pixel shaders;, therefore, RTL simulation is used only for evaluating the performance for different numbers of pixel shaders.

Benchmarks

- Cyber Samurai from Rightware's OpenGL|ES benchmarks [10] has been thoroughly tested. Other various benchmark suites such as 3DMark Mobile ES 2.0 [10], and Glbenchmark [11] are under testing.

Cyber samurai

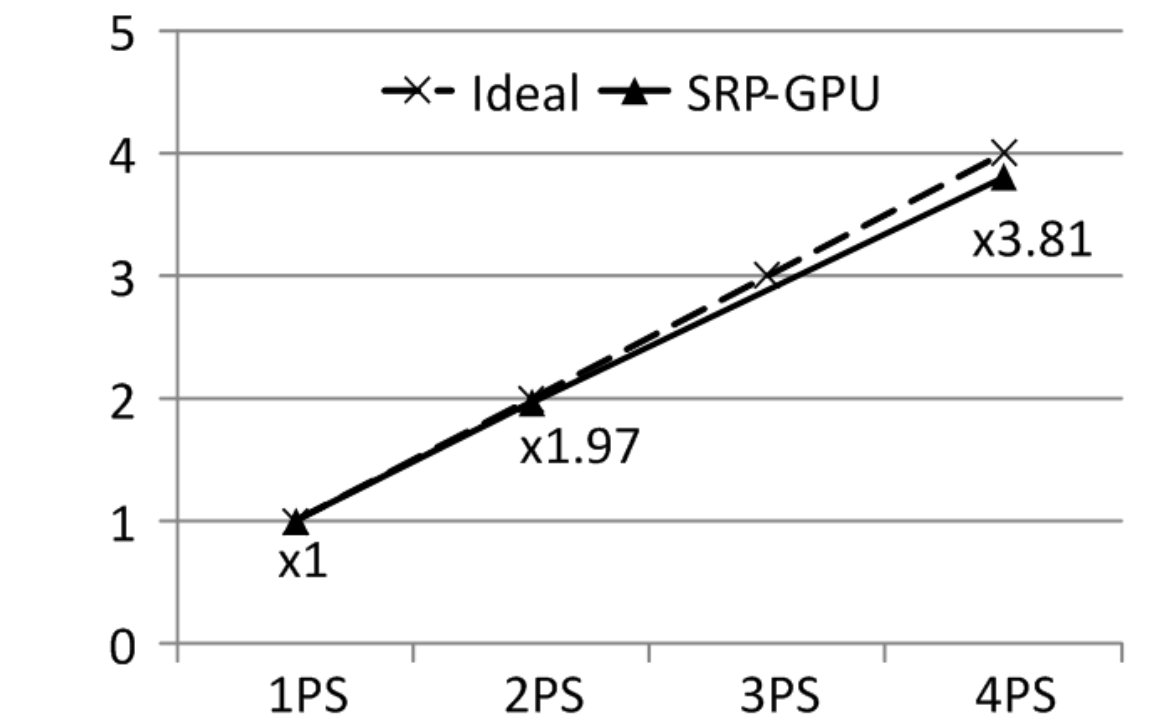
Taiji

Hover jet

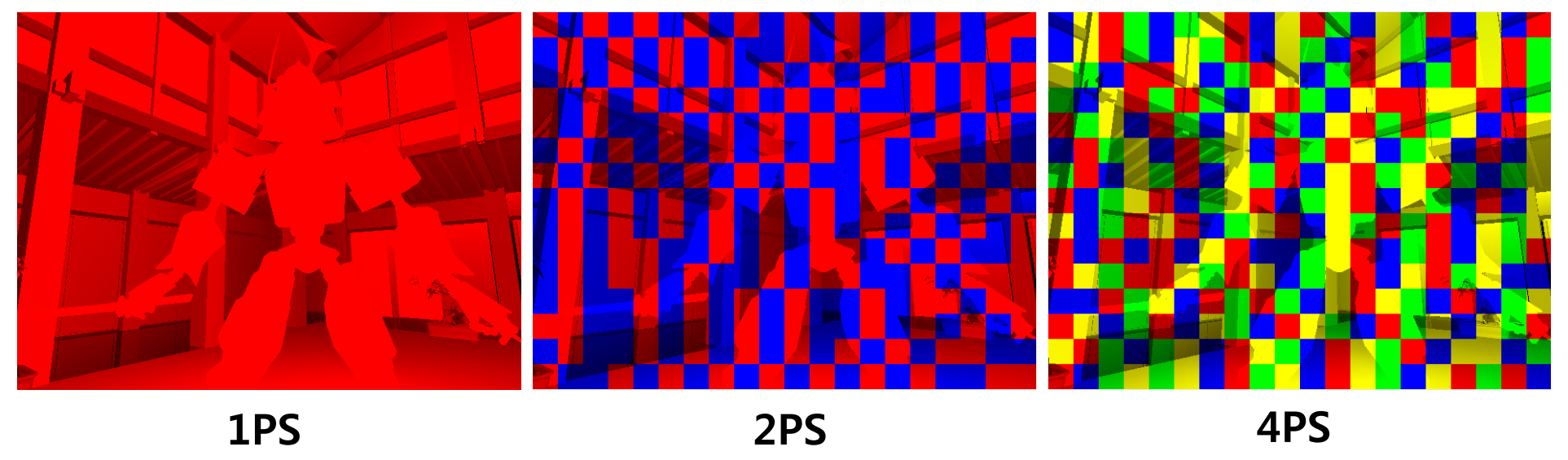
Egypt

Pro

Performance Evaluation



Relative performance result



1PS

2PS

4PS

Visualization of workload distribution
(Red:PS1, Blue:PS2, Green:PS3, Yello:PS4)

- The above figure shows a comparison of the performance result as a function of the scaling of the number of pixel shaders. The very effective use of pipelining and parallel rendering could allow our GPU to scale well, up to **x3.81** with 4 pixel shaders.
- Our current GPU is predicted to render at up to **89.6 fps with WVGA resolution at a 333 MHz core clock speed**.
- To achieve better performance with this device, we are currently redesigning the architecture to support more advanced features such as a **unified shader** and a **multithreaded streaming CGRA**.
- Finally, we expect that our GPU will be a core intellectual property for future application processors.

References

[1] S. C. Goldstein et al, "PipeRench: A reconfigurable architecture and compiler," IEEE Computer, Vol. 33, No. 4, pp. 70–77 (2000).
[2] Silicon Hive (Intel), HiveFlex Series, http://www.silicon-hive.com (2011).
[3] B. Mei et al, "Architecture Exploration for a Reconfigurable Architecture Template," IEEE Design & Test of Computers, Vol. 22, No. 2. pp. 90-101 (2005).
[4] M. B. Taylor et al, "The Raw Microprocessor: A Computational Fabric for Software Circuits and General-Purpose Programs," IEEE Micro, Vol. 22, No. 2, pp. 25-35 (2002).
[5] H. Singh et al," MorphoSys: An Integrated Reconfigurable System for Data-Parallel and Computation-Intensive Applications," IEEE Transaction on Computers, Vol. 49, No. 5, pp. 465–481 (2000).
[6] B. Mei et al, "ADRES: An Architecture with Tightly Coupled VLIW Processor and Coarse-Grained Reconfigurable Matrix," Proceedings of International Conference on Field-Programmable Logic and Applications (FPL), pp. 622-625, Tampere, Finland (2005).
[7] B. Mei et al, "Exploiting Loop-Level Parallelism on Coarse-Grained Reconfigurable Architectures Using Modulo Scheduling," Proceedings of Design, Automation, and Test in Europe (DATE), pp. 575-581, Paris, France (2003).
[8] Imagination, PowerVR series, http://www.imgtec.com/powervr/powervr-graphics.asp (2011)
[9] ARM, Mali series, http://www.arm.com/products/multimedia/mali-graphics-hardware/index.php (2011)
[10] RightWare, benchmarking software, http://www.rightware.com/en/Benchmarking+Software (2011)
[11] Glbenchmakr, http://www.glbenchmark.com (2011)

joe.w.lee@samsung.com