Practical Issues in Implementing Fast and Accurate SystemC-Constructed Virtual Platform Simulation

Authors: Yu-Fu Yeh[1,2] and Chung-Yang (Ric) Huang[2]

Affiliation: [1]ICL @ ITRI; [2]GIEE @ NTU

Presenter: Yu-Fu Yeh

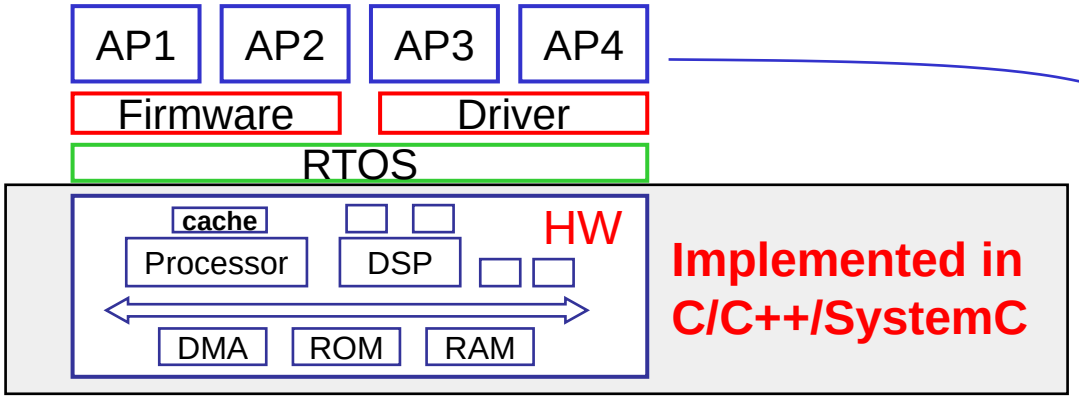
工業技術研究院
Industrial Technology
Research Institute

Outlines

- Introduction
- Preliminaries
  - Simulation overhead in virtual platform simulation
  - Synchronization reduction by an asynchronous discrete event simulation scheme
- QuteVP+ Implementations:
  - QuteVP+ engine
  - QuteVP+ utility library
- Experimental Results
- Conclusions

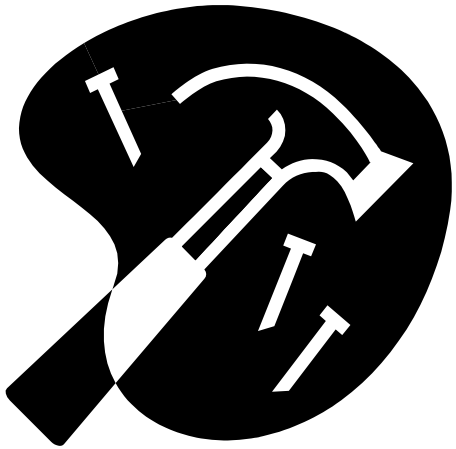
What is virtual platform simulation

- A software-constructed hardware simulation platform
  - Hardware components are constructed by software language (e.g: SystemC)
  - Software program can be executed on the processor model (e.g: Instruction set simulator, ISS)
- Usually contains everything for a system
  - Hardware: processor, bus, memory, DMA…
  - Software: OS, firmware, drivers, embedded programs
- Objectives:
  - System design optimization, architecture exploration, system-level verification

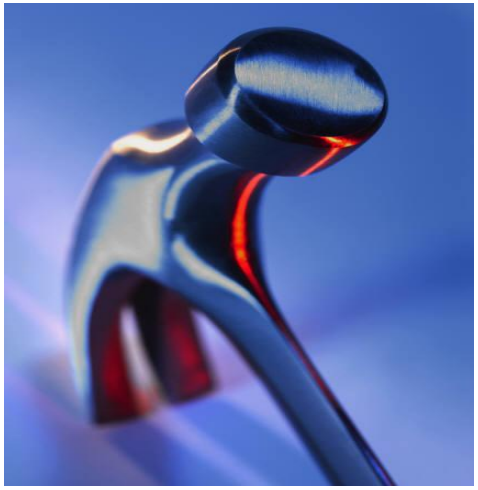


The problem in virtual platform simulation

- A trade-off between simulation efficiency and simulation accuracy

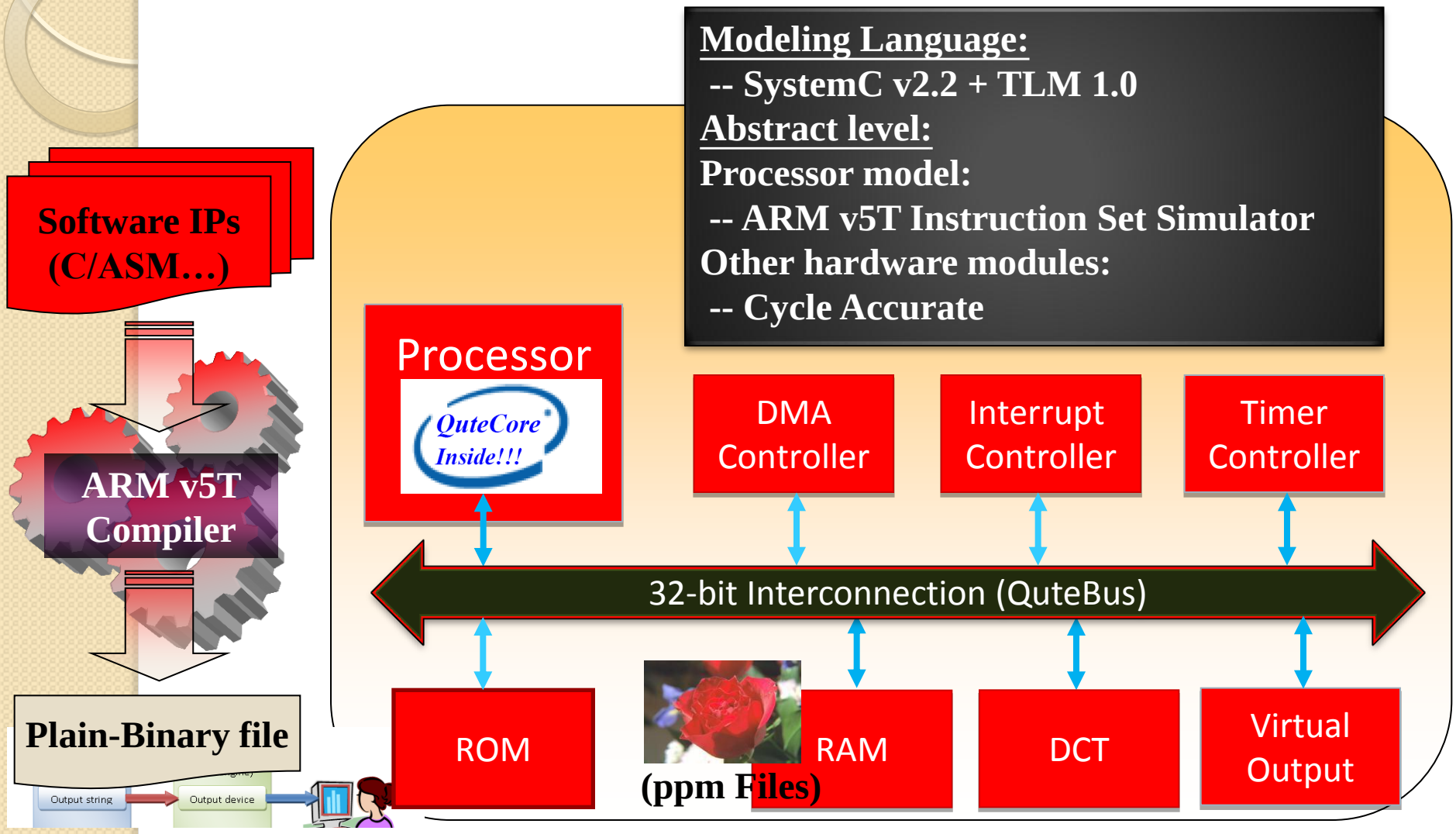


v.s



- Simulated models with higher abstract levels
  - better simulation efficiency
  - E.g. functional simulation
- Simulated models with lower abstract levels
  - More accurate outcome
  - E.g. cycle accurate simulation

Introduction



Introduction

Our experience of SoC-based virtual platform simulation

- m13 Version
  - 2007.05 Finished
  - Cycle Accurate
  - Pin Accurate
  - About 12.0 KIPS
- m17 Version
  - 2008.05 Finished
  - Cycle Accurate
  - OSCI TLM 1.0
  - About 27.5 KIPS

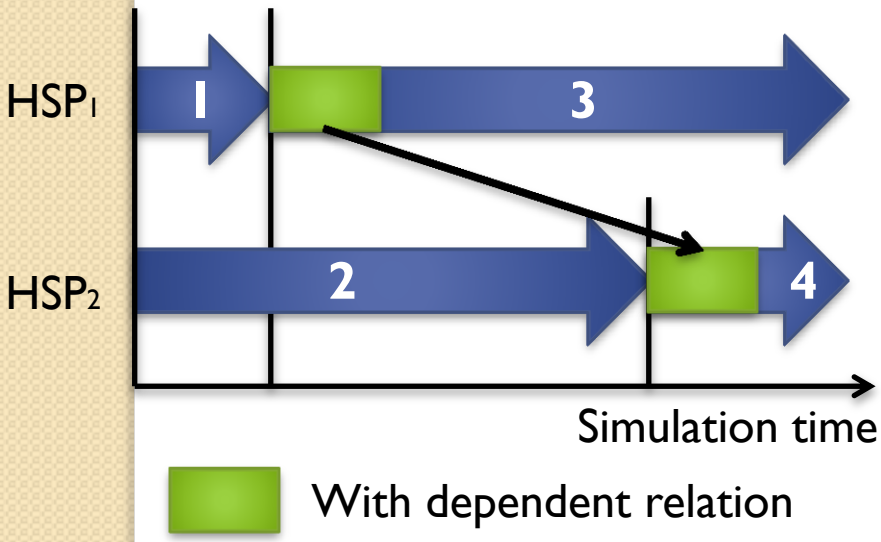
UNACCEPTABLE SIMULATION SPEED

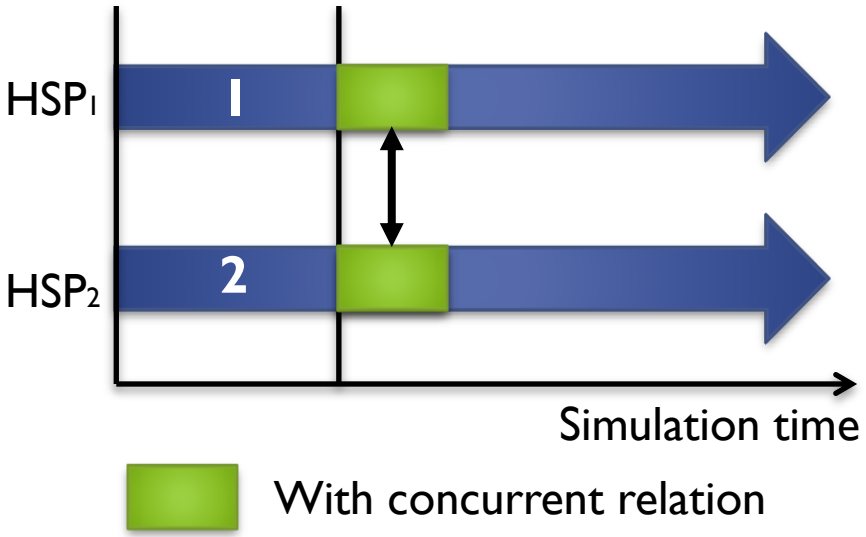
Introduction

- QuteVP+, a simulation framework, is proposed to follow our proposed simulation scheme to conduct fast and accurate SystemC-constructed virtual platform simulation

Preliminaries

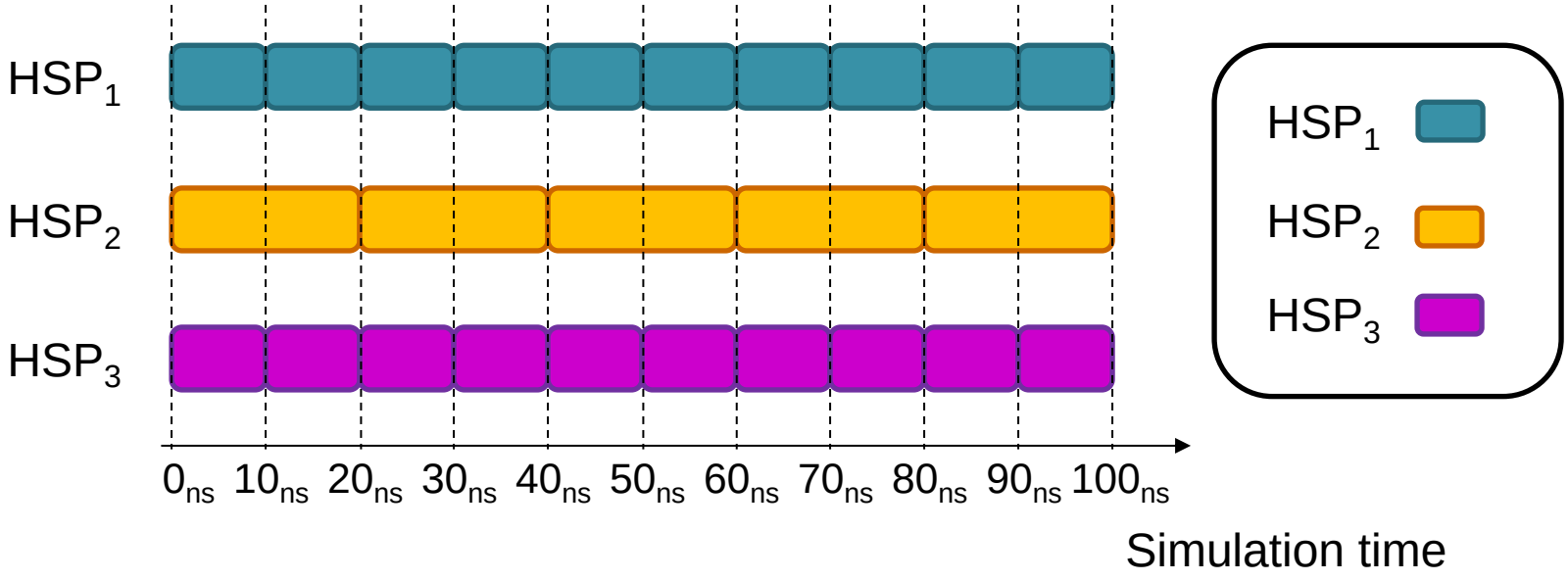
- Virtual platform simulation must consider dependent and concurrent relations among hardware components
  - Schedule **H**ardware **S**imulation **P**rocess (HSP) in a proper chronological order





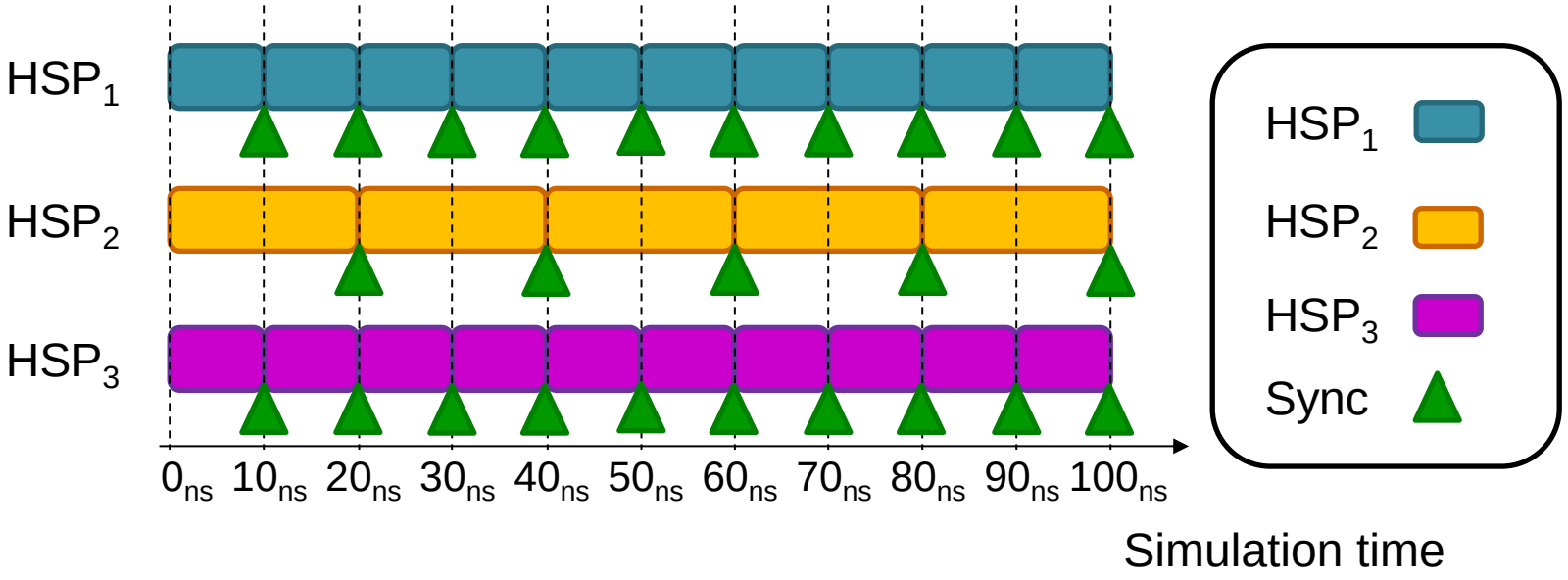
Preliminaries

- To accurately mimic the concurrent hardware behavior, SystemC simulator schedules the HSPs created by SC_METHOD, SC_THREAD with **sync**hronous **d**iscrete-**e**vent **s**cheme (Sync-DES), or called clock-step simulation scheme (CSSM)



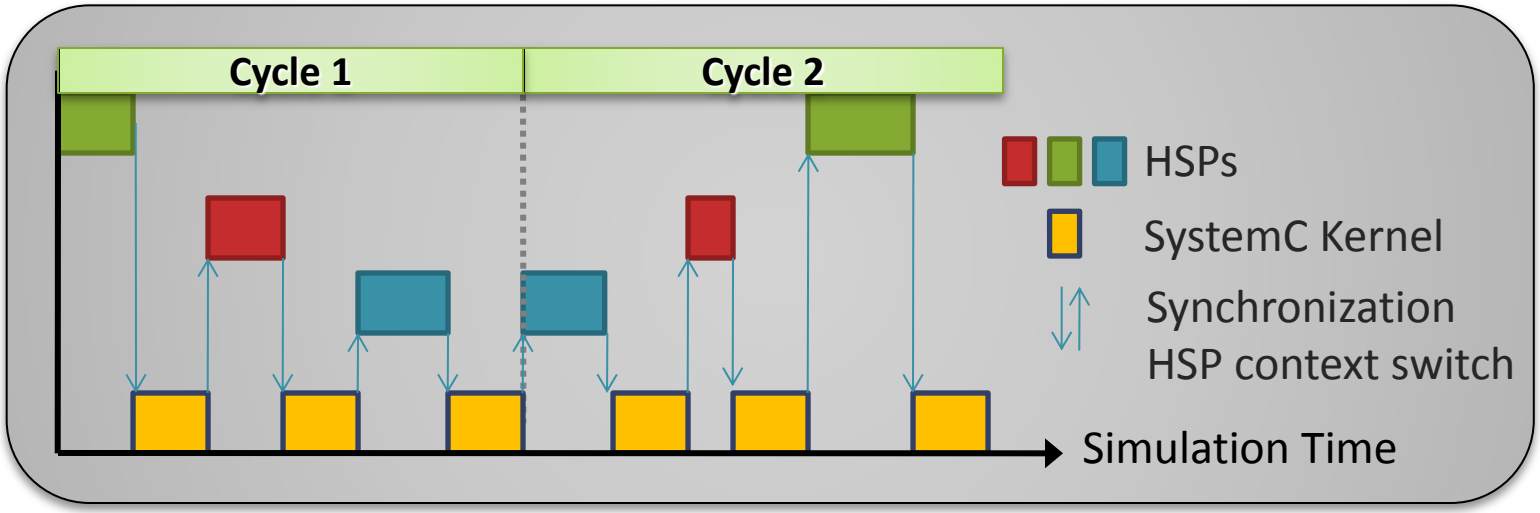
Preliminaries

- To schedule HSPs, SystemC kernel evokes synchronization (thread context switches), during simulation



Preliminaries

- Each module gets scheduled one or multiple times in one clock cycle
  - Using serial simulator to mimic concurrent behavior
  - Synchronizing HSPs with big simulation overhead

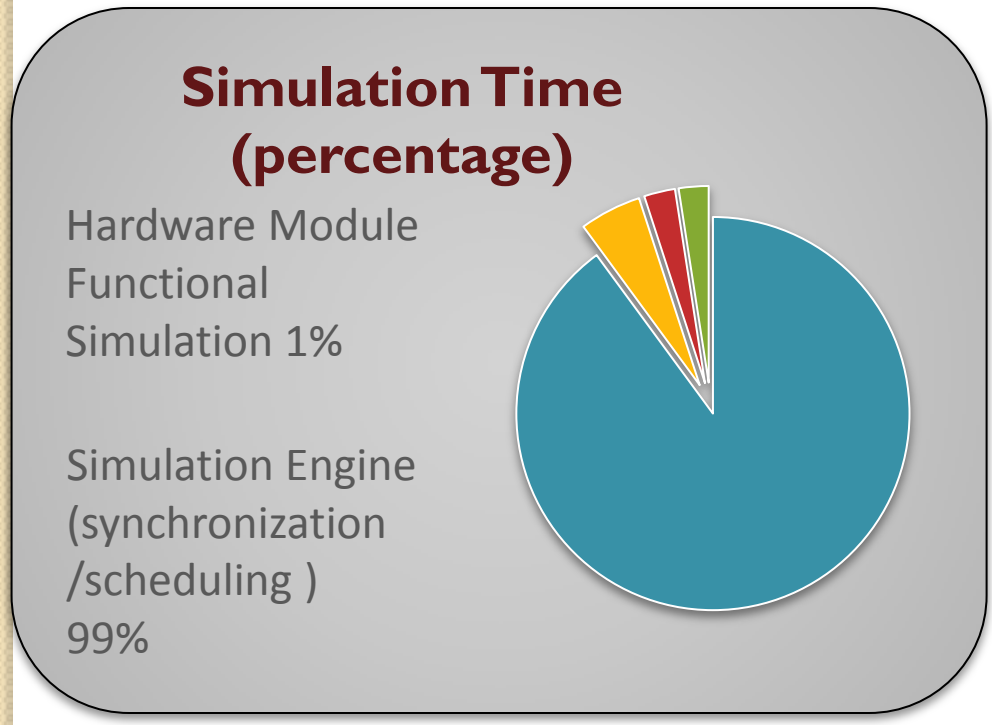


Context switches across one clock cycle over simulation time chart

What's the problem?

Observation

- The biggest bottleneck of SystemC simulation is in the simulation kernel

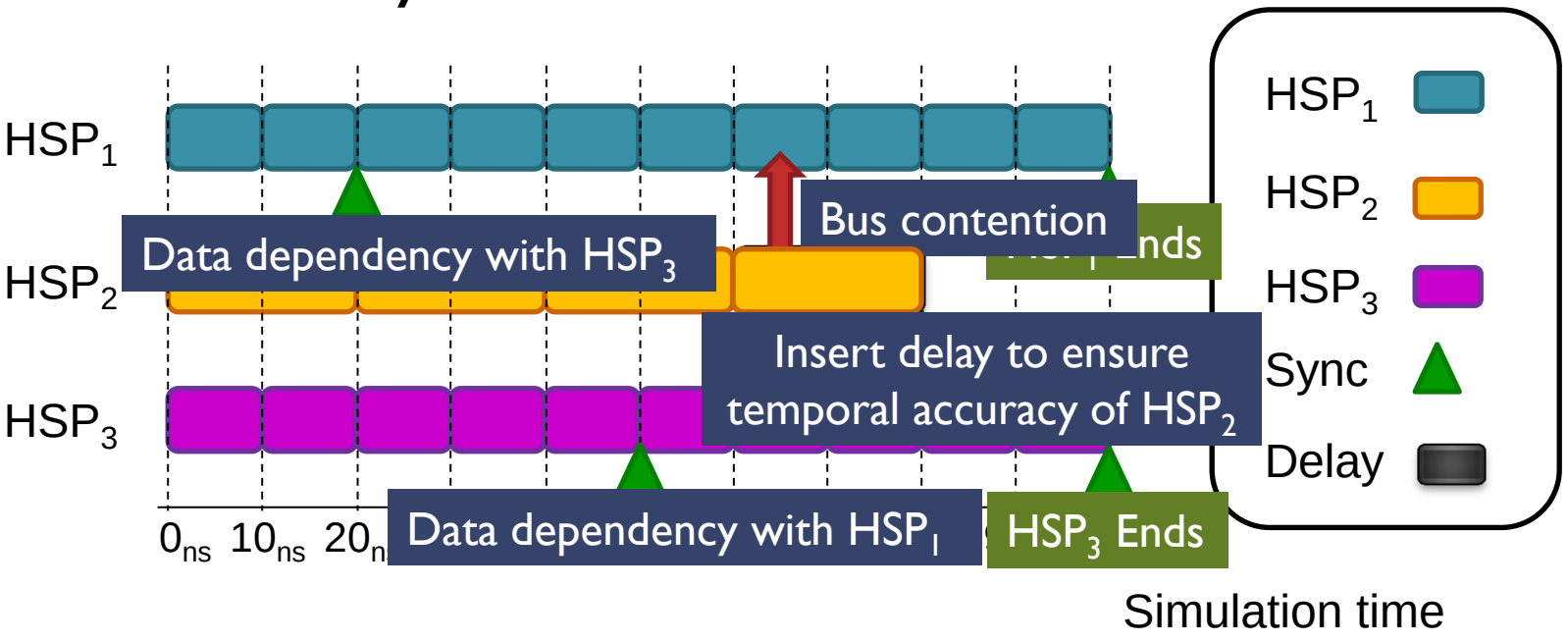


Virtual platform simulation time profiling

- Serious simulation overhead
  - Context Switches
  - Scheduling
  - Data copy

Asynchronous discrete even simulation for synchronization reduction

- In contrast to sync-DES, asynchronous discrete event simulation (async-DES) scheme benefits synchronization reduction

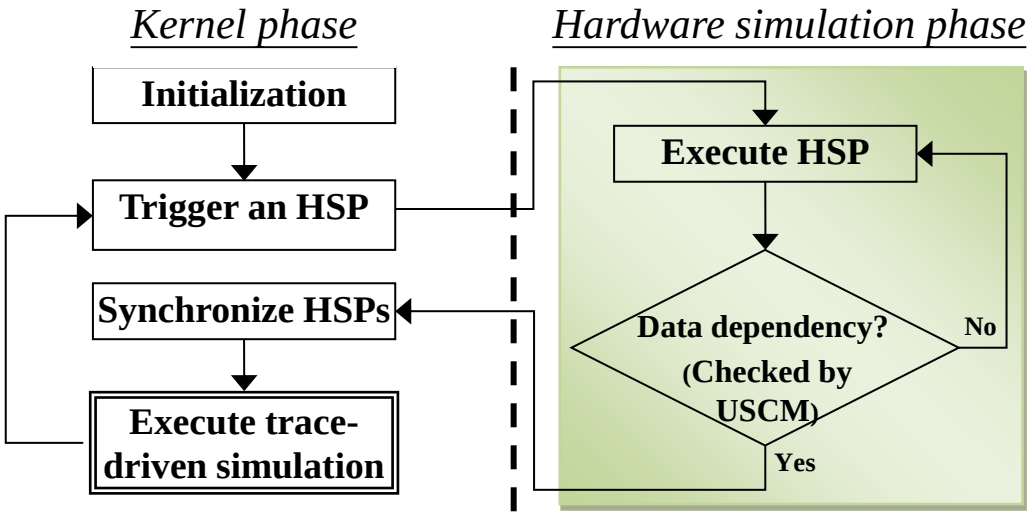


Asynchronous discrete event simulation (async-DES)

- In the virtual platform simulation with "**async-DES**", there are two requirements
  - A synchronization checking mechanism to **"avoid dependency violation"**
  - A timing reconstruction technique to "**maintain temporal accuracy"**

Synchronization reduction with our proposed simulation scheme

- This work is based on our proposed simulation scheme [1] with USCM[2] and Trace-drive simulation to conduct fast and accurate MPSoC virtual platform simulation



[1]: Y.F. Yeh, H.S. Lin and C.Y.(Ric) Huang, "A Fast and Accurate MPSoC Virtual Platform Simulation with Ultra Synchronization Checking Method and Trace-driven simulation", accepted by IEEE transactions of Computer-Aided Designs of Integrated Circuits and Systems, 2013, Jan.

[2]: Y.F. Yeh, C.Y. (Ric) Huang, C.A. Wu, and H.S. Lin, "Speeding Up MPSoC Virtual Platform Simulation by Ultra Synchronization Checking Method", in Proc. IEEE Design Automation and Test in Europe (DATE), Mar 211, pp. 1-6.

Implementation of QuteVP+

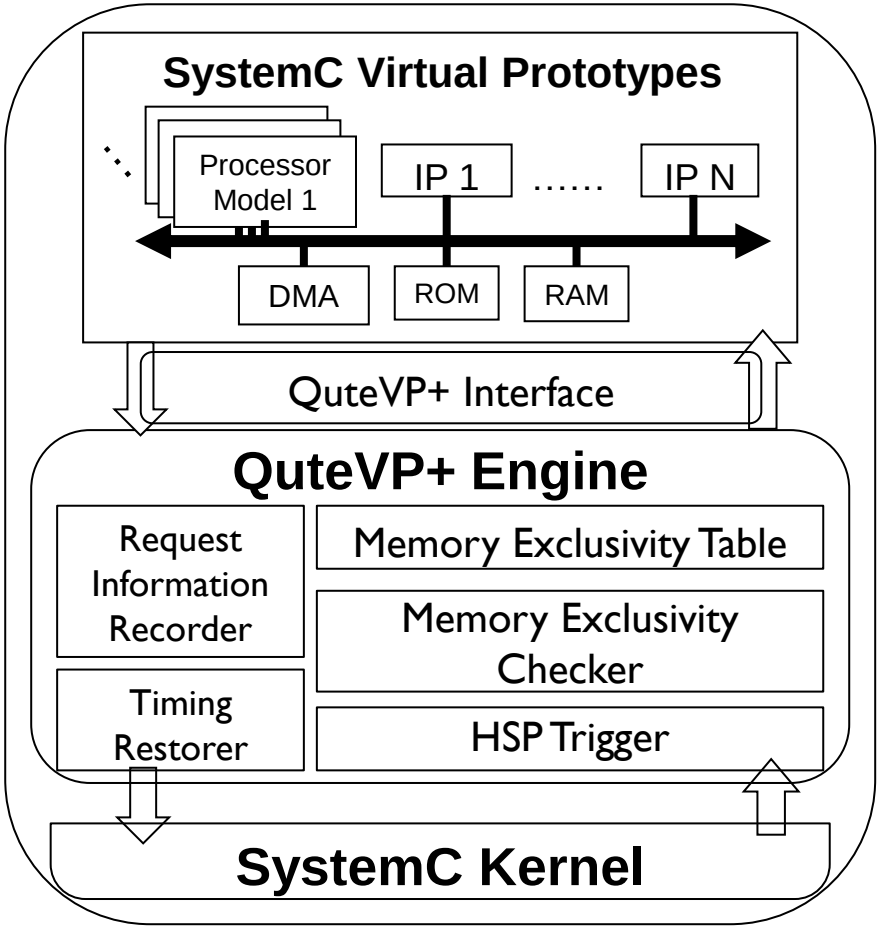
- Goal:
  - Realize the introduced async-DES scheme on SystemC-Constructed virtual platform simulation
- Difficulties
  - The simulation scheme in SystemC follows Sync-DES (Clock-step simulation scheme)
  - Modifying simulation scheme, the simulation must solve issues in “compatibility” and “adaptability”

Implementation of QuteVP+

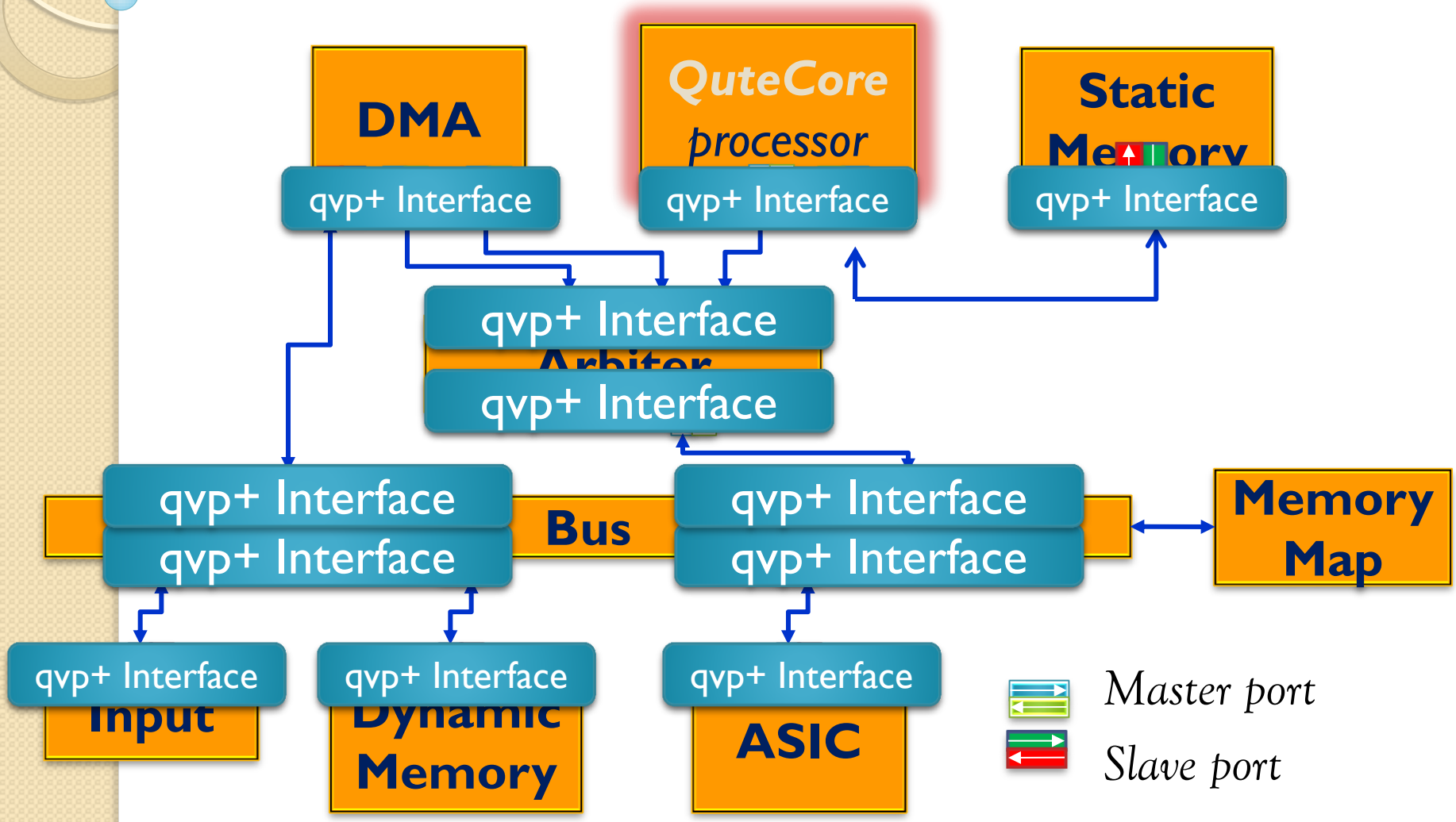
- Compatibility
  - Ensure the replacement of simulation scheme without affecting the primitive SystemC-defined functions
    - E.g. the functions, such as event notify(), wait() is relevant to the scheduling behavior
- Easy to use
  - Consider the convenience to adapt the parallel out-of-order execution approach on SystemC-Constructed virtual platform

Implementation of QuteVP+

- QuteVP+ Overview

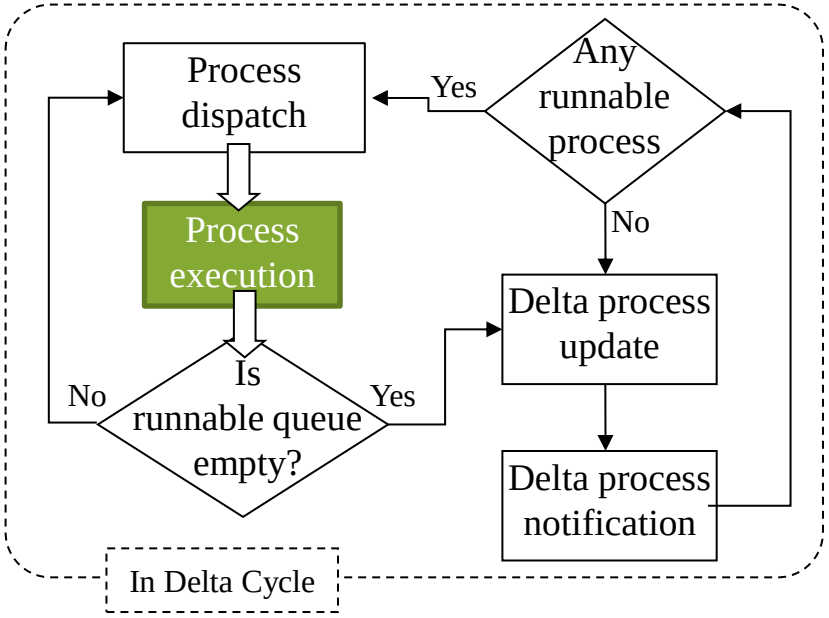


Use QuteVP+ interface to connect hardware model and QuteVP+ Engine



Implementation of QuteVP+

- QuteVP+ creates an independent process to manipulate HSPs by out-of-order execution
  - Delta cycle scheduling
    - HSPs use timeless wait function for synchronization
      - QuteVP+ engine enables async-DES scheduling in each delta cycle
    - Record simulation traces of each HSP for trace-driven simulation
      - Maintain dependency relation
      - Reconstruct accurate simulation time
    - Process notification



Implementations of QuteVP+

- QuteVP+ utility library
  - While requesting a memory access, an HSP can use our utility library to check data dependency

```
#include <QuteVP_Utility.h>
…. // QVP+ communication channel inheritance
void ARM_ISS::send_request() {
  // replace TLM communication function call, e.g. m_master_port->nb_put(mReqs);
  // by calling data-dependency checking function
  // and execute synchronization if necessary
  if (QuteVP_Engine->DataDependencyChecker(mReq, mResp)) {
    wait(sync_ok_event);
    QuteVP_Engine->RequestTransmitter(mReq, mResp);
  }
  if (mReq.get_command()==MEM_READ)
      M_resp_data = mResp.get_data();
}
```

Implementations of QuteVP+

- QuteVP+ utility library
  - RequestTransmitter() performs "direct data access" to reduce data copy

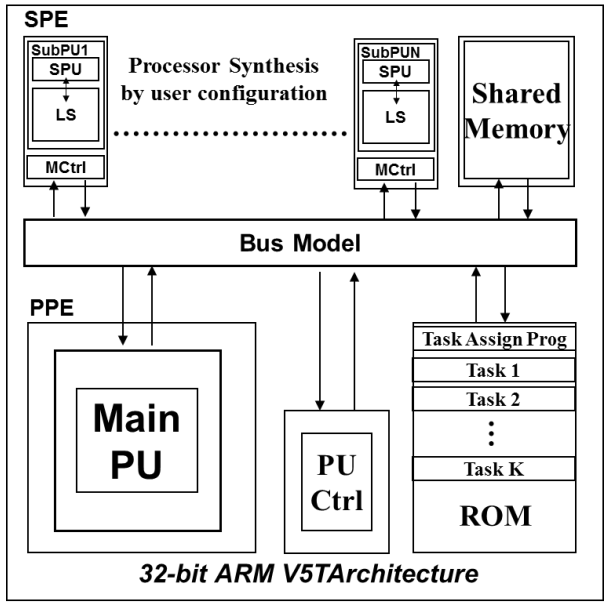
```
#include <QuteVP_Utility.h>
// Using targetID mapping to the corresponding get_request function
qvp_response Qutevp_Engine::RequestTransmitter(qvp_request& mReq) {
  targetAddr = mReq.get_address();
  targetID = findIDfromMemMap(targetAddr);
  return pHSP[targetID].HSPptr()->get_request(mReq);
}

// Calculate the request address to seek the target ID
unsigned int pSysc::findIDfromMemMap(unsigned int& Addr) {
  ….
  return TargetID
}
```

Experimental results

- We compare the simulation efficiency of MPSoC virtual platform simulation where QuteVP+ performs with different simulation approaches
  - **CSSM**: Clock-Step Simulation Method, the synchronous discrete event simulation scheme that the primitive SystemC follows
  - **USCM**: USCM, the asynchronous discrete event simulation scheme in our previous work

Experimental results

- We construct an CELL-Like MPSoC virtual platform and JPEG encode and sparse matrix multiplication programs as test software cases running on the MPSoC virtual platform
- Experimental environment
  - Workstation with Intel xeon CPU (qual-core*2) 2.2 GHz, 16GB RAM
  - CentOS kernel 2.6
  - Virtual platform constructed with SystemC v2.2



Experimental results

- The comparison of synchronization count (Sync-Count) with CSSM and USCM

Sparse matrix multiplication				
#CPU	#Inst	Simulation Cycle	Sync-Count by CSSM	Sync-Count by USCM
1	169,314,561	254,261,472	722,859,678	28,423,967
2	121,085,896	156,417,989	460,436,669	12,508,290
4	107,747,016	124,834,536	365,547,740	7,564,476
8	105,501,921	114,852,199	335,037,092	3,891,842
16	112,941,976	119,699,619	349,944,254	2,075,848
32	135,121,633	145,001,298	419,101,187	1,220,659

JPEG-Encode				
#CPU	#Inst	Simulation Cycle	Sync-Count by CSSM	Sync-Count by USCM
1	512,513,774	771,969,678	2,212,558,094	87,409,800
2	386,970,594	507,338,577	1,520,454,360	45,790,479
4	314,622,268	376,936,991	1,093,803,850	21,489,726
8	287,006,102	327,838,743	926,966,612	10,945,309
16	272,113,534	308,490,886	860,594,492	5,436,347
32	264,576,313	295,742,881	831,027,316	2,918,037

Experimental results

$$Simulation\ speed = \frac{\sum_{i=0}^{n=\#CPU} the\ number\ of\ simulated\ instructions}{Simulation\ runtime_{Simulation\ scheme}\ (sec)}$$

$$Speedup\ Ratio_{(Scheme_A with\ respect\ to\ Scheme_B)} = \frac{Simulation\ speed_{Scheme\_A}}{Simulation\ speed_{Scheme\_B}}$$

#CPU	SMM			JPEG-Enc		
	$CSSM_{(KIPS)}$	$USCM_{(KIPS)}$	Speedup Ratio	$CSSM_{(KIPS)}$	$USCM_{(KIPS)}$	Speedup Ratio
1	11.4	1376.5	121.0	13.7	1331.2	96.8
2	16.3	1121.2	68.6	17.4	1236.3	70.9
4	21.7	1007.0	46.3	20.7	1096.2	53.0
8	32.4	933.6	28.8	29.7	1043.7	35.1
16	24.3	824.4	33.9	23.3	901.0	38.7
32	21.0	682.4	32.6	22.5	747.4	33.2

KIPS means "Kilo Instruction Per Second

Rate of DDC.Runtime	The number of simulating processors (#CPUs)					
	1	2	4	8	16	32
DDC.Runtime % (USCM)	0.23%	0.22%	0.24%	0.27%	0.23%	0.23%

The rate of simulation runtime for data-dependency checking (DDC) with respect to total simulation runtime

Conclusions

- QuteVP+ can performs an async-DES to reduce unnecessary synchronization and reconstruct accurate simulation time
- QuteVP+ can
  - Perform the better simulation speed over 100+ times with respect to the conventional SystemC-based virtual platform
  - Offer accurate simulation outcome