SystemC-based Design Space Exploration of a 3D Graphics Acceleration SoC for Consumer Electronics

Tse-Chen Yeh[1], Tsung-Yu Ho[1], Hung-Yu Chen[1], and Ing-Jer Huang[1]

[1] Department of Computer Science and Engineering, National Sun Yat-sen University, Kaohsiung, 804, Taiwan
{tcyeh, tyho, hychen}@esl.cse.nsysu.edu.tw
ijhuang@cse.nsysu.edu.tw

Abstract. In order to solve the system performance bottleneck of a 3D graphics acceleration SoC, we exploit design space exploration on performance evaluation and benchmark characteristics using SystemC. We find out the bottleneck according to the simulation results of 9 hardware/software configurations and find out the tradeoffs between different configurations. The performance issues of SoC have been explored under the low-cost constraints, such as cache size effect, hardware accelerations and memory traffic. In conclusions, we provide the performance/cost tradeoffs and 3D graphics benchmark features for designing a 3D graphics SoC.

Keywords: design space exploration, SystemC modeling, 3D graphics SoC, transaction-level modeling.

1 Introduction

Consumer electronics are cost-sensitive, such as digital television (DTV), and customers demanded higher performance with lower price. To develop a system-on-a-chip (SoC) as discussed above needs the hardware/software configurations to meet the performance under the cost constraint. In order to find the tradeoffs between performance and cost, design space exploration using SystemC has been evaluate the system performance in [1].

The fast hardware/software configuration can be achieved by using CoWare™ Platform Architect, which provides the hardware/software integration and simulation platform using SystemC [2]. Firstly, software written by C/C++ can be translated effortlessly into SystemC hardware model because SystemC is a C++ class library. Secondly, CoWare™ Platform Architect provides IP library and cycle-accurate transactional bus simulator (TBS) for IP reuse which can shorten the design flow from re-design the IP models and raise the simulation results up to the bus cycle accurate [3].

In this paper, we focus on 3D graphics hardware acceleration SoC for DTV system. The design flow starts from pure software simulation on the target platform, and then the software will be accelerated by additional hardware or architecture refinement according to performance bottlenecks we have found out. After acceleration, the new

bottleneck will be revealed. Thus we continue to simulate new accelerated platform then overcome new bottlenecks repeatedly until the required performance has been approximated.

The organization of this paper is as follows. In Section 2, we describe the simple 3D graphics pipeline, 3D graphics acceleration SoC platform and platform building methodology. The hardware/software configurations are presented in Section 3. And design space exploration will be implemented in Section 4. Finally, we conclude the overall analysis and performance/cost tradeoffs for 3D graphics acceleration SoC Design.

2 Preliminary

3D graphics has been developed maturely on workstation and personal PCs from 1990. These high performance solutions are not suitable for the embedded systems or consumer electronics because they also paid high cost and high power consumption. Although the 3D graphics workloads have been investigated by static and dynamic forms [9, 10] and one is on the mobile device [11], the system performance still has been profiled on instruction-level evaluation.

Due to the proposed architecture of 3D graphics SoC will be applied on DTV applications, the differences will be introduced in the following section opposed to the conventional 3D graphics processing.

2.1 3D Graphics Rendering Pipeline

To obtain the software functionality, we exploit a tile-based simple 3D graphics pipeline (shown in Fig. 1) written by C/C++. The tile-based rasterization can reduce amount of memory accesses and is suited to the embedded system. In the simple pipeline, the geometry processing includes the viewing transformation, lighting and perspective transformation. After geometry computation, the triangle information will be passed to the tile divide function which handles the tile setup information. The raster processing begins at the scan conversion, shades fragment without texture, and the final results will be written into frame buffer for display after passing Z-test. In this paper, the texturing will not be modeled for performance evaluation.

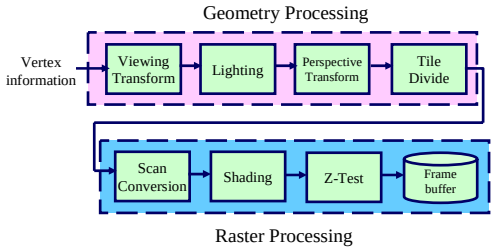


Fig. 1. Simple 3D graphics pipeline for performance analysis.

2.2 3D Graphics Acceleration SoC

The basic combinations of a system includes core processor, system bus and system memory. In order to accelerate 3D graphics processing, a 3D graphics engine has to handle the geometry and rasterization acceleration. Furthermore, the processing results should be display via a display engine (DE) which has responsibility to move processed data from frame buffer in memory to the display device. Finally, the direct memory access (DMA) will be appended for more efficient memory access consideration. All the hardware components connect the system bus via bus interface (BI).

Fig. 2 shows the final platform of 3D graphics acceleration SoC. We choose ARM926EJ-S for the system processor core and AMBA 2.0 AHB bus to be the system bus. Taking low cost into account, we select the single port SDRAM to be the system memory which contains vertex information, temporary data, frame buffer and Z buffer. To fit the DTV applications, the frame buffer and Z buffer are located with 640x480 screen size.

To approximate the realistic system memory traffic, the bus contentions from DE should be taken into consideration. The DE is a bridge between frame buffer and DTV display, and get the frame buffer data with read burst transfer periodically. According to the DTV frame rate, the data consuming frequency is 25 MHz which occupied 100 MB/sec of system bandwidth under 200 MHz system clock frequency.

In terms of 3D graphics engine in Fig. 2, the functions in 3D graphics pipeline have been translated into hardware model. Because of the complexity of geometric computation, we divide the geometry module into 3 pipeline stages and each stage runs 16 cycles [5]. For design a low cost raster module we choose the tile-based implementation to reduce area cost. Thus we first have to design a tile divider (TD) that receives the triangle data output from geometry module to generate tile list [6].

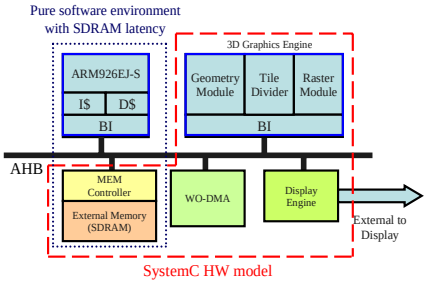


Fig. 2. System functional block of 3D graphics acceleration SoC.

2.3 Platform Building

CoWare™ Platform Architect integrates the IP library and hardware models into system platform for simulation. Both of the processor and AHB bus IP are provided by the IP library. The processor IP contains an ARM instruction set simulator (ISS)

for software execution. And AMBA library use a cycle-accurate TBS and a set of Transaction-Level Modeling (TLM) API for IP integration [7]. To declare ports type pre-defined in AMB bus library, hardware can read from or write to other components connected to the same bus by API function calls. All of the bus arbitrations and read/write latency are managed by TBS.

For the cycle-accurate simulation, we model hardware modules at transaction level (TL) with timing information [4]. The power of TLM is to separate hardware operations into computation and communication. The computation means the function of hardware, and the communication presents the read/write operations of hardware. Because we use SystemC as modeling language, the functions of simple 3D graphics pipeline can be migrated into computation part of our hardware model without any modification. With regard to communication, the primitive SystemC ports of our hardware modules will be replaced by AHB ports for platform integration.

We use dot line to enclose the simulation environment for pure software in Fig. 2. And the rest blocks enclosed by dash line are the hardware modules modeled by SystemC. Through the SystemC modeling at transaction level, we can evaluate the system performance at early design stage and the simulation time of TLM is faster than the Register-Transfer Level (RTL) simulation.

3 System Configurations

In this section, the 3D graphics benchmarks and hardware/software configurations will be described. Total 9 hardware/software configurations will be introduced for our design space exploration, 4 of these configurations are used to evaluate software acceleration, and others are used to explore different hardware solutions.

3.1 Simulation Environment

We choose three benchmarks for 3D graphics rendering shown in Fig. 3, and features of these benchmarks are listed in Table 1. Helicopter benchmark has more geometric computations than raster processing in proportion compared with Teapot benchmark. And the Elephant benchmark has the most complexity on whether geometric or raster processing. All of the simulation statistics will be collected by TBS for the performance analysis.

(a) Teapot (b) Helicopter (c) Elephant

Fig. 3. 3D graphics benchmarks for design space exploration.

Table 1. Features of three 3D graphics benchmarks.

	Teapot	Helicopter	Elephant
Total vertices number	12,288	15,414	87,840
Processed tiles	135	64	141

3.2 Hardware/software Configurations

Table 2 gives the detail hardware/software configurations for design space exploration. In this table, "pSW" means 3D graphics pipeline execution without any hardware acceleration. And "ADS" enclosed by parentheses indicate the simulation use ARMulator, i.e. ARM ISS of ARM Developer Suite, without SDRAM model and DE contentions. And "$" suffix refers to software performance improved by adding instruction/data (I/D) cache. "PA" is the abbreviation of Platform Architect, which means the software execute with SDRAM model.

Table 2. Hardware/software configurations for design space exploration.

Configuration Name	hardware/software Configurations							
	Function			Bus Interface	SDRAM latency	Cache	Display Engine	Geometry FIFO
	Clear Buffer	Geometry	Raster	Transfer Mode				
pSW (ADS)	SW	SW	SW	N/A	N/A	N/A	N/A	N/A
pSW (ADS $)	SW	SW	SW	N/A	N/A	ON	N/A	N/A
pSWD (PA)	SW	SW	SW	N/A	ON	N/A	ON	N/A
pSWD (PA $)	SW	SW	SW	N/A	ON	ON	ON	N/A
GED_s	SW	HW	HW	Single	ON	N/A	ON	N/A
GED_b	SW	HW	HW	Burst	ON	N/A	ON	N/A
GDGED_b	GDMA	HW	HW	Burst	ON	N/A	ON	N/A
WDGED_b	WDMA	HW	HW	Burst	ON	N/A	ON	N/A
WDGED_Fb	WDMA	HW	HW	Burst	ON	N/A	ON	ON

The prefix "GE" of fifth and sixth configurations indicate the 3D graphics pipeline accelerated by 3D Graphics Engine shown in Fig. 2, and the appending character "s" and "b" means hardware read/write in single transfer or burst transfer mode individually. For bus cycle-accurate simulation, we insert execution cycles into hardware models which refer to the synthesizable RTL design [4-5]. Furthermore, the suffix "D" indicates the DE will occupy 15% bus bandwidth.

"Clear Buffer" is not the function of 3D graphics pipeline, but is necessary which reset values in frame buffer and Z buffer before rendering the next frame. The last three configurations with prefix "GD" or "WD" mean the different DMAs used to improve the performance of "Clear Buffer". We use "GD" to indicate a generic DMA which can read/write memory, and use "WD" to represent a write-only DMA which only write a fixed value from the internal register to reset memory, thus the specific DMA can reduce memory traffic by decreasing lots of read operations from memory.

"Geometry FIFO" will be used for the comparisons of performance improvement on geometry module with or without a ping-pong buffer. In the "Function" column, the "HW" indicates the configuration use hardware implementation and "SW" indicates software implementation. In "Transfer Mode" column, the "Single" means

the single transfer mode of AMBA bus, and the “Burst” means the burst transfer mode of AMBA bus.

4 Design Space Explorations

In the section, we exploit these hardware/software configurations introduced in previous section on design space explorations. Bus and memory overhead will be explored in section 4.1, 3D graphics engine will be appended in section 4.2, and the efficiency of different transfer mode of AHB bus will be discussed in section 4.3. The new bottlenecks will revealed by our explorations in section 4.4. In addition, an imagined geometry FIFO buffer will be modeled for advanced performance/cost tradeoff in section 4.5. Finally, the benchmark characteristics will be extracted for advanced survey on system performance of 3D graphics in section 4.6.

4.1 Cache Size Optimization without Hardware Acceleration

The design space exploration starts from software evaluation without any hardware acceleration. And we apply cache to achieve more efficient performance on pure software executions. Fig. 4 shows the execution cycles variations of cache size from 128 bits to 16k bits are estimated in “ADS”, in contrast to “PA” started from 4k bits due to processor IP limitation. We adapt I/D cache of 4kb/4kb for performance optimization. Either “pSW (ADS$4)” or “pSW (PA$4)” configurations can gain at least 36.7% performance improvement shown in Table 3. The ideal bus assumes the bus without transaction latency, and the subtraction of execution cycles of “ADS” and “PA” configurations is the bus and SDRAM overhead in pure software solutions.

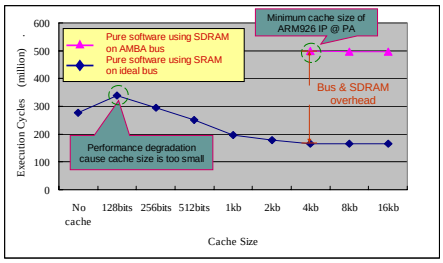


Fig. 4. Performance improvement by cache size variation of Teapot benchmark.

Table 3. Cache size optimization on software execution.

	pSW (ADS) (cycles)	pSW (ADS$4) (cycles)	improvement (ADS)	psW (PA) (cycles)	pSW (PA$4) (cycles)	improvement (PA)
Clear Buffer	2,935,822	2,167,247	26.2%	9,003,353	5,778,533	35.8%
Geometry	133,002,461	66,931,465	40.1%	396,033,200	209,441,410	47.1%
Raster	139,867,002	97,181,194	30.5%	460,224,000	284,340,000	38.2%
Total	412,376,118	260,850,843	36.7%	865,260,553	499,559,943	42.3%

4.2 Performance of Hardware Acceleration

Due to the "pSW" results, the geometry and raster processing are the bottlenecks on software execution, because the best case of drawing Helicopter benchmark with I/D cache need 1.216 second for one frame at 200MHz system clock frequency shown in Table 4. The DTV applications only can tolerant the frame rate down to 5~6 frame per second under the lowest resolution. To overcome these bottlenecks, we exploit hardware acceleration on geometry and rasterization parts. To compare "pSW" with cache optimization with "GE" configurations, the system performance can be improved approximate 99% ((Execution cycle of software – Execution cycle of hardware) / Execution cycle of software) by hardware acceleration.

Table 4. Comparisons of execution time at 200MHz system clock frequency.

	Teapot	Helicopter	Elephant
pSW (PA$4)	2.498 sec	1.216 sec	9.078 sec
GED_s	0.070 sec	0.067 sec	0.108 sec
GED_b	0.063 sec	0.053 sec	0.070 sec

4.3 Efficiency of Bus Transfer Mode

Due to the cost consideration, we use the SDRAM to be the system memory. Amount of memory read/write accesses occur significant memory latency. The AMBA AHB provides burst mode transfer which can overlap these SDRAM CAS and row change latency [8], but may degrade few bus traffic if too many burst transfers need to compete.

We use "GE_s" and "GE_b" configurations to find out the benefit by using burst transfer. Notice that benefit will not reveal if we use low-latency SRAM to be system memory. The burst transfer benefits are shown in Table 5, and the results show approximate 33.1% improvement in Teapot benchmark, approximate 47.4% and 49.7% improvement at least in Helicopter and Elephant benchmarks.

Table 5. Single transfer vs. Burst transfer.

	Teapot	Helicopter	Elephant
Geometry Speedup	36%	52%	49.7%
Raster Speedup	33.1%	47.4%	54.7%

4.4 Performance Bottleneck on Clear Buffer

After the hardware acceleration, the hidden problems have been enhanced. "Clear Buffer" occupies 86.9% execution time of the total system performance compared with the software solution only 1.0% and becomes the new performance bottleneck shown in Fig. 5.

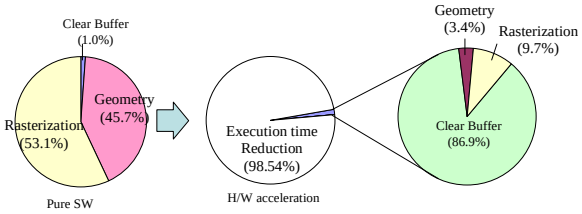


Fig. 5. New bottleneck on Clear Buffer after hardware acceleration.

Because the "ClearBuffer" contains the reset operations to the whole frame buffer in memory, we append a generic DMA (GDMA) or write-only DMA (WO-DMA) mentioned in Section 3 to automatically clear frame buffer and Z buffer. Thus large mount of memory traffic between processor and memory can be reduced. Table 6 shows the performance improvement by comparing the execution cycles of "GED_b" with "GDGED_b" and "WDGED_b" configurations. The WO-DMA will be adapted for this system configuration, because the performance has improved 89.4% at least on "Clear Buffer" execution.

Table 6. Performance improvement on Clear Buffer.

	Teapot	Helicopter	Elephant
Generic DMA	50.5%	41.77%	41.40%
Write-only DMA	90.76%	89.69%	89.20%

4.5 Performance of Geometry Processing

In order to accelerate the geometry processing, we design a ping-pong buffer to prefetch the vertex information that can prevent the waiting time if bus is too busy to provide the necessary data. "WDGD_Fb" configuration refers to appending the ping-pong buffer.

To extract the benefit of ping-pong buffer for geometry processing, we use the "WDGD_b" and "WDGD_Fb" configurations into comparisons. The performance has been improved at least 13.8% on geometry processing by appending 8x4 bytes ping-pong buffer. Finally, the performance/cost tradeoffs are listed in Table 7, and the hardware costs are estimated by synthesis in [5, 6]. And the performance variations of different configurations are shown in Fig. 6. Also notice Fig. 6 uses logarithm scale on Y axis and start from 100k cycles.

Table 7. Performance/cost tradeoffs of different hardware/software configurations.

hardware/software configuration	Performance improvement			hardware cost (gate count)
	Teapot	Helicopter	Elephant	
Software only	0%	0%	0%	N/A
I/D cache	42.60%	72.51%	35.70%	8kb SRAM
GE	98.54%	98.81%	99.51%	542.8k
GE + WO-DMA	99.69%	99.74%	99.45%	542.8k + 6.2k
GE + WO-DMA+ Ping-pong buffer	99.80%	99.75%	99.44%	542.8k + 6.2k + 32B SRAM

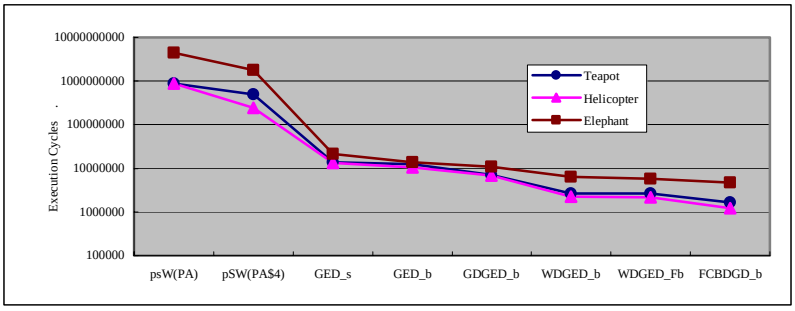
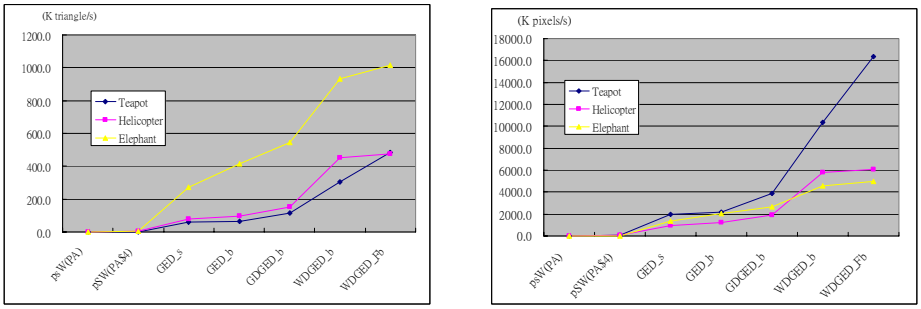


Fig. 6. Performance improvement of different hardware/software configurations.

4.6 Characteristics of 3D Graphics Benchmarks

3D graphics workloads have been investigated in [9-11] at instruction-level. After evaluating the system performance of 9 hardware/software configurations, the characteristics of benchmarks can be extracted from 7 configurations because the ISS of ADS can not simulate too large benchmarks, such as Helicopter or Elephant.



(a) Geometry characteristics (b) Rasterization characteristics

Fig. 7. Benchmark characteristics of 3D graphics acceleration.

The characteristics of 3D graphics can be divided into geometry and rasterization parts. The triangle rate or vertex rate give the index of geometry performance, and the pixel fill rate give the index to rasterization performance. The benchmark characteristics are shown in Fig. 7. Although the slops of these benchmarks are similar, the amplitudes of variation are not the same. In geometry processing, the Elephant benchmark in Fig. 7(a) gives the significant amplitude because it contains the largest the vertex number of object compared with Teapot and Helicopter benchmarks. On the other side, Teapot benchmark in Fig. 7(b) gives the highest processing area per triangle, thus it can get the largest performance improvement on the rasterization processing. In summary, we use hardware/software configurations

not only for design space exploration, but also for extracting the benchmark characteristics.

5 Conclusions

The market of consumer electronics always changes by following consumers' requirements. In this paper, we use the design space exploration for performance evaluation and the performance/cost tradeoffs by exploiting 9 hardware/software configurations. And we also characterize the benchmark features for advanced survey on 3D graphics processing. The performance evaluation and benchmark characteristics can give many benefits and references for system designers to determine their system architecture and hardware/software configurations.

Notice that memory traffic is still the performance bottleneck of a low-cost SoC which applied on 3D graphics rendering. And texture mapping, memory allocation, comparisons of alternative architecture modeling, power/performance analysis, and multi-layer bus architecture for 3D graphics acceleration SoC will become open issues in our future works.

References

1. Jang, H.O., Kang, M., Lee, M.J., Chae, K., Lee, K., Shim, K.: High-level System Modeling and Architecture Exploration with SystemC on a Network SoC: S3C2510 Case Study. DATE, Vol 1. (2004) 538-543
2. Platform Architect Datasheet. CoWare, *http://www.coware.com/PDF/products/PlatformArchitect.pdf*
3. Model Library Datasheet. CoWare, *http://www.coware.com/PDF/products/ModelLibrary.pdf*
4. Black, D.C., Donovan, J.: SystemC: From The Ground Up. Springer Science+Business Media, (2004)
5. Hsiao, S.F., Huang, T.Y., Tieng, T.C.: Design and Verification of a Platform-Based Low-Cost 3D Graphics Geometry Engine Using Area-Reduced Arithmetic Function Units. 17th VLSI Design/CAD (2006) 8-11
6. Chang, T.N., Tsai, C.H., Tsai, M.C., Lin, H.L.: Design of a Low-cost 3-D Graphic Rendering Accelerator. 17th VLSI Design/CAD (2006) 533-536
7. AMBA TLM API. CoWare, *http://www.coware.com/*
8. Synchronous DRAM. Micron, *http://wwwmicron.com/sdram*
9. Chiueh, T.-C., Lin, W.-J.: Characterization of Static 3D Graphics Workloads. ACM SIGGRAPH/EUROGRAPHICS, (1997)
10. Mitra, T., Chiueh, T.-C.: Dynamic 3D Graphics Workload Characterization and the Architectural Implications. 32nd AISM, (1999) 62-71
11. Mochocki, B. C., Lahiri, K., Cadambi, S., Hu, X. S.: Signature-Based Workload Estimation for Mobile 3D Graphics. 43rd DAC, (2006) 592 - 597