MOBILE 3D GRAPHICS SoC

From Algorithm to Chip

Jeong-Ho Woo
Korea Advanced Institute of Science and Technology, Republic of Korea
Ju-Ho Sohn
LG Electronics Institute of Technology, Republic of Korea
Byeong-Gyu Nam
Samsung Electronics, Republic of Korea
Hoi-Jun Yoo
Korea Advanced Institute of Science and Technology, Republic of Korea

John Wiley & Sons (Asia) Pte Ltd

MOBILE 3D GRAPHICS SoC

MOBILE 3D GRAPHICS SoC

From Algorithm to Chip

Jeong-Ho Woo
Korea Advanced Institute of Science and Technology, Republic of Korea
Ju-Ho Sohn
LG Electronics Institute of Technology, Republic of Korea
Byeong-Gyu Nam
Samsung Electronics, Republic of Korea
Hoi-Jun Yoo
Korea Advanced Institute of Science and Technology, Republic of Korea

John Wiley & Sons (Asia) Pte Ltd

Copyright © 2010 John Wiley & Sons (Asia) Pte Ltd, 2 Clementi Loop, # 02-01, Singapore 129809

Visit our Home Page on www.wiley.com

All Rights Reserved. No part of this publication may be reproduced, stored in a retrieval system or transmitted in any form or by any means, electronic, mechanical, photocopying, recording, scanning, or otherwise, except as expressly permitted by law, without either the prior written permission of the Publisher, or authorization through payment of the appropriate photocopy fee to the Copyright Clearance Center. Requests for permission should be addressed to the Publisher, John Wiley & Sons (Asia) Pte Ltd, 2 Clementi Loop, #02-01, Singapore 129809, tel: 65-64632400, fax: 65-64646912, email: enquiry@wiley.com.

Designations used by companies to distinguish their products are often claimed as trademarks. All brand names and product names used in this book are trade names, service marks, trademarks or registered trademarks of their respective owners. The Publisher is not associated with any product or vendor mentioned in this book. All trademarks referred to in the text of this publication are the property of their respective owners.

This publication is designed to provide accurate and authoritative information in regard to the subject matter covered. It is sold on the understanding that the Publisher is not engaged in rendering professional services. If professional advice or other expert assistance is required, the services of a competent professional should be sought.

Other Wiley Editorial Offices

John Wiley & Sons, Ltd, The Atrium, Southern Gate, Chichester, West Sussex, PO19 8SQ, UK

John Wiley & Sons Inc., 111 River Street, Hoboken, NJ 07030, USA

Jossey-Bass, 989 Market Street, San Francisco, CA 94103-1741, USA

Wiley-VCH Verlag GmbH, Boschstrasse 12, D-69469 Weinheim, Germany

John Wiley & Sons Australia Ltd, 42 McDougall Street, Milton, Queensland 4064, Australia

John Wiley & Sons Canada Ltd, 5353 Dundas Street West, Suite 400, Toronto, ONT, M9B 6H8, Canada

Wiley also publishes its books in a variety of electronic formats. Some content that appears in print may not be available in electronic books.

Library of Congress Cataloging-in-Publication Data

Mobile 3D graphics SoC : from algorithm to chip / Jeong-Ho Woo ... [et al.].
p. cm.
Includes index.
ISBN 978-0-470-82377-4 (cloth)
1. Computer graphics. 2. Mobile computing. 3. Systems on a chip. 4. Three dimensional display systems.
I. Woo, Jeong-Ho.
T385.M62193 2010
621.3815–dc22

2009049311

ISBN 978-0-470-82377-4 (HB)

Typeset in 10/12pt Times by Thomson Digital, Noida, India.
Printed and bound in Singapore by Markono Print Media Pte Ltd, Singapore.
This book is printed on acid-free paper responsibly manufactured from sustainable forestry in which at least two trees are planted for each one used for paper production.

Contents

Preface

This is a book about low-power high-performance 3D graphics for SoC (system-on-chip). It summarizes the results of 10 years of "ramP" research at KAIST (ramP stands for RAM processor) – a national project that was sponsored by the Korean government for low-power processors integrated with high-density memory. The book is mostly dedicated to 3D graphics processors with less than 500 mW power consumption for small-screen portable applications

Screen images continue to become ever-more dramatic and fantastic. These changes are accelerated by the introduction of more realistic 3D effects. The 3D graphics technology makes vivid realism possible on TV and computer screens, especially for games. Complicated and high-performance processors are required to realize the 3D graphics. Rather than use a general-purpose central processing unit (CPU), dedicated 3D graphic processors have been adopted to run the complicated graphics software.

There is no doubt that all the innovations in PC or desktop machines will be repeated in portable devices. Cellphones and portable game machines now have relatively large screens with enhanced graphics functions. High-performance 3D graphics units are included in the more advanced cellphones and portable game machines, and for these applications a low power consumption is crucial. In spite of the increasing interest in 3D graphics, it is difficult to find a book on portable 3D graphics. Although the principles, algorithms and software issues have been well dealt with for desktop applications, hardware implementation is more critical for portable 3D graphics. We intend to cover the 3D graphics hardware implementation especially emphasizing low power consumption. In addition, we place emphasis on practical design issues and know-how. This book is an introduction to low-power portable 3D graphics for researchers of PC-based high-performance 3D graphics as well as for beginners who want to learn about 3D graphics processors. The HDL file at the end of the book offers readers some first-hand experience of the algorithms, and gives a feel of the hardware implementation issues of low-power 3D graphics.

This book would not have been possible without help from many colleagues and supporters. First we would like to thank Dr Sejeong Park of Mediabridge, Dr Yongha Park of Samsung, Dr Chiwon Yoon of Samsung, and Dr Ramchan Woo of LG for their pioneering efforts in mobile 3D graphics research at the Semiconductor Systems

laboratory in KAIST. Professor Kyuho Park of KAIST, Professor T. Kuroda of Keio University, and Dr Ian Young of Intel helped us to begin our research on low-power 3D graphics. We would like also to thank Professor Young-Joon Park of Seoul National University, Dr Heegook Lee of LG, and Dr Huh Youm of Hynix for their help with the ramP project. Last but not least, we would like to thank James and his team at John Wiley for their care in the birth of this book.

1

Introduction

1.1 Mobile 3D Graphics

Mobile devices are leading the second revolution in the computer graphics arena, especially with regard to 3D graphics. The first revolution came with personal computers (PC), and computer graphics have been growing in sophistication since the 1960s. To begin with it was widely used for science and engineering simulations, special effects in movies, and so on, but it was implemented only on specialized graphics workstations. From the late 1980s, as PCs became more widely available, various applications were developed for them and computer graphics moved on to normal PCs – from specific-purpose to normal usage.

Three-dimensional graphics are desirable because they can generate realistic images, create great effects on games, and enable slick effects for user interfaces. So 3D graphics applications have been growing very quickly. Almost all games now use 3D graphics to generate images, and the latest operating systems – such as Windows 7 and OS X – use 3D graphics for attractive user interfaces. This strongly drives the development of 3D graphics hardware. The 3D graphics processing unit (GPU) has been evolving from a fixed-function unit to a massively powerful computing machine and it is becoming a common component of desktop and laptop computers.

A similar revolution is happening right now with mobile devices. The International Telecommunications Union (ITU) reports that 3.3 billion people – half the world's population – used mobile phones in 2008, and Nokia expects that there will be more than 4 billion mobile phone users (more than double the number of personal computers) in the world by 2010 [1]. In addition, mobile devices have been dramatically improved from simple devices to powerful multimedia devices; a typical specification is 24-bit color WVGA (800×480) display screen, more than 1 GOPS (giga-operations per second) computing power, and dedicated multimedia processors including an image signal processor (ISP), video codec and graphics accelerator.

Mobile 3D Graphics SoC: From Algorithm to Chip Jeong-Ho Woo, Ju-Ho Sohn, Byeong-Gyu Nam and Hoi-Jun Yoo
© 2010 John Wiley & Sons (Asia) Pte Ltd

So 3D graphics is no longer a guest on mobile devices. A low-cost software-based implementation is used widely in low-end mobile phones for user interfaces or simple games, while a high-end dedicated GPU-based implementation brings PC games to the mobile device.

Nowadays, 3D graphics are becoming key to the mobile device experience. With the help of 3D graphics, mobile devices have been evolving with fruitful applications ranging from simple personal information management (PIM) systems (managing schedules, writing memos, and sending e-mails or messages), to listening to music, playing back videos, and playing games. Just as with the earlier revolution in the PC arena, 3D graphics can make mobile phone applications richer and more attractive – this is the reason why I have used the phrase "second revolution."

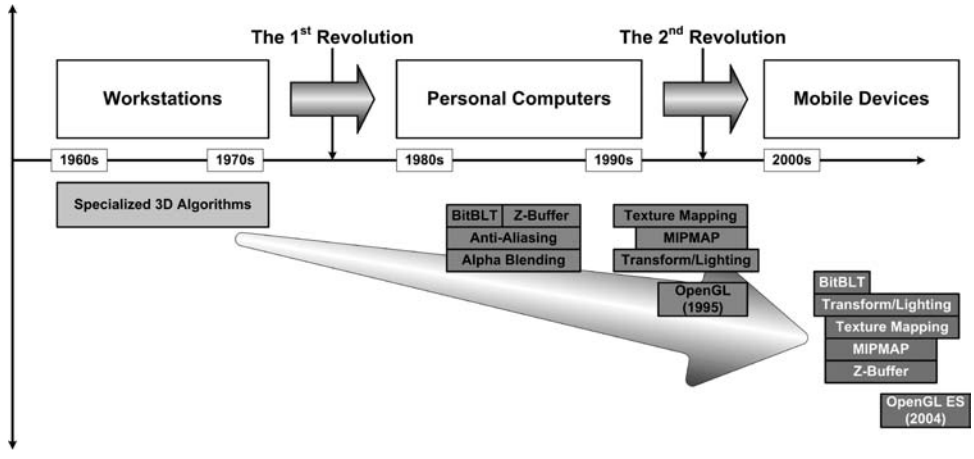


Figure 1.1 A history of 3D graphics

Development of mobile 3D graphics was started basically in the late 1990s (Figure 1.1). Low-power GPU hardware architectures were developed, and the software algorithms of PCs and workstations were modified for mobile devices. Software engines initially drove the market. Among them, two notable solutions – "Fathammer's X Forge" engine and "J-phone's Micro Capsule" – were embedded in Nokia cellular phones and J-phone cellular phones. Those software solutions do provide simple 3D games and avatars, but the graphics performance is limited by the computation power of mobile devices. So new hardware solutions arrived to the market. ATI and nVidia introduced "Imageon" and "GoForce" using their knowledge of the PC market. Besides the traditional GPU vendors like nVidia and ATI, lots of challengers introduced great innovations (Figure 1.2). Imagination Technology's MBX/SGX employs tile-based rendering (discussed in Chapter 5) to reduce data transactions between GPU and memory. Although tile-based rendering is not widely

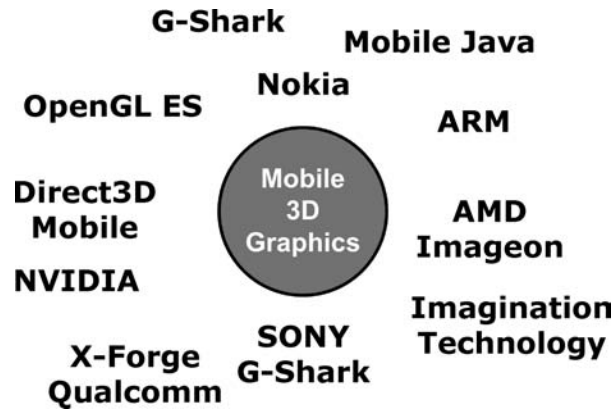


Figure 1.2 Mobile 3D graphics

used on the PC platform, it is very useful in reducing power consumption so that the MBX/SGX has become one of the major mobile GPUs on the market. FalanX and Bitboys developed their own architectures – FalanX Mali and Bitboys Acceleon – and they provided good graphics performance with low power consumption. Although those companies merged into ARM and AMD, respectively, their architectures are still used to develop mobile GPUs in ARM and AMD.

1.2 Mobile Devices and Design Challenges

As mentioned in the previous section, mobile devices have evolved at a rapid pace. To satisfy various user requirements there are lots of types of mobile device, such as personal digital assistant (PDA), mobile navigator, personal multimedia player (PMP), and cellular phone. According to their physical dimensions or multimedia functionality, these various devices can be categorized into several groups, but their system configurations are very similar. Figure 1.3 shows two leading-edge mobile devices and their system block diagram. Recent high-performance mobile devices consist of host processor, system memories (DRAM and Flash memory), an application processor for multimedia processing, and display control. Low-end devices do not have a dedicated application processor, to reduce hardware cost. Evolution of the embedded processor and display devices has led to recent exciting mobile computing.

1.2.1 Mobile Computing Power

In line with Moore's law [2], the embedded processors of mobile devices have been developing from simple microcontroller to multi-core processors and the computing

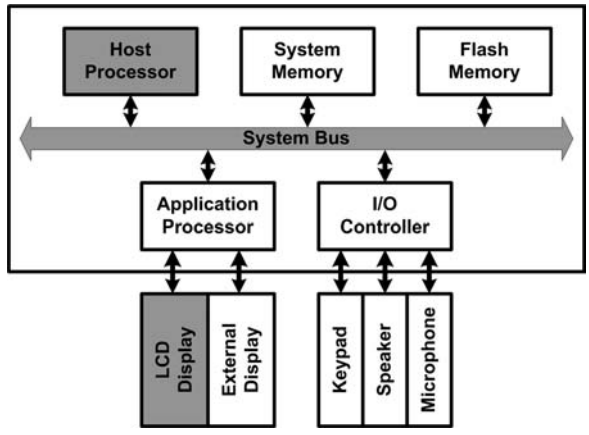


Figure 1.3 Mobile devices and their configuration

power has kept increasing roughly 50% per year. To reduce power consumption, an embedded processor employs RISC (Reduced Instruction Set Computer) architecture, and the computing power already exceeds that of the early Intel Pentium processors.

Typically, recent mobile devices have one or two processors as shown in Figure 1.4. Low-end devices have a single processor so that multimedia applications are implemented in software, while high-end devices have two processors, one for real-time operations and the other for dedicated multimedia operations. The host processor performs fundamental operations such as running the operating system, and personal information management (PIM). Meanwhile the application processor is in charge of

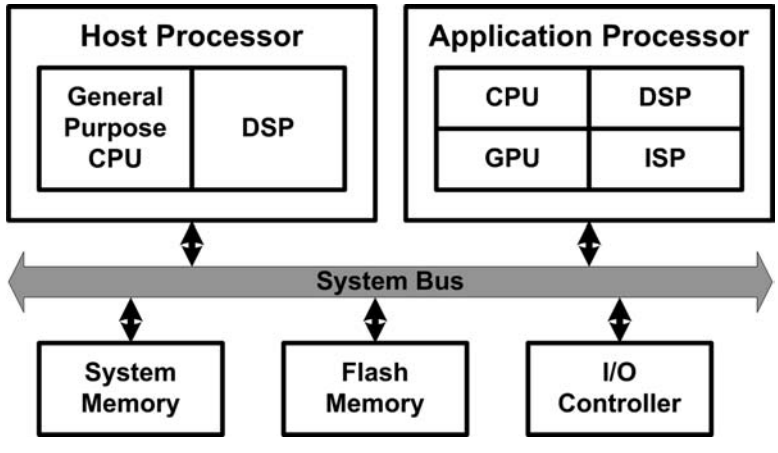


Figure 1.4 Embedded processors and system architecture

high-performance multimedia operations such as MPEG4/H.264 video encoding or real-time 3D graphics. To increase computing power, the newest processors employ multi-core architecture. Some high-performance processors contain both a general-purpose CPU and DSP together, and some application processors consist of more than four processing elements to handle various multimedia operations such as video decoding and 3D graphics processing.

1.2.2 Mobile Display Devices

It is safe to say that evolution of mobile display devices leads the revolution of mobile devices, especially the multimedia type. The first mobile devices had a tiny monotone display that could cope with several numbers or characters. Recent mobile devices support up to VGA (640×480) 24-bit true-color display. The material of the display device is also changing from liquid crystal to AMOLED (Active Mode Organic Light Emitting Diode). The notable advantages of AMPLED are fast response time (about 100 times faster than LCD), and low power consumption. Since it does not require back-lighting like the LCD, the power consumption and weight are reduced, and the thickness is roughly one-third of the LCD. Of course the functionality of the display device is improved too, so that nowadays we can use touch-screens on mobile devices.

1.2.3 Design Challenges

Although the funtionality of mobile devices is greatly improved, there are many design challenges in component design. In short, there are three major challenges.

Physical dimension – The main limitation of mobile devices is definitely their physical size. For portability the principal physical dimension is limited to about 5 inches (12.5 cm), and the latest high-end cellular phones do not exceed 4 inches. That means there is limited footprint on the system board, and components should be designed with small footprint.

Power consumption – Since the mobile device runs on a battery, the power consumption decides the available operating time. As the performance increases it consumes more power owing to the faster clock frequency or richer hardware blocks. Therefore, increasing operating time by reducing power consumption is as important as increasing computing power.

System resources – Mobile devices cannot have rich system resources owing to the physical dimension and power consumption. They cannot utilize a wide-width system bus and cannot use high-performance memory such as DDR2 or DDR3. Despite this, mobile devices provide quite high performance to satisfy user requirements.

To meet these design challenges, many mobile components are designed as SoC (System-on-a-Chip). Since the SoC includes various functional blocks such as processor, memory, and dedicated functional blocks in a single die, we can achieve high performance with low power consumption and small area.

1.3 Introduction to SoC Design

System-on-a-Chip has replaced key roles of VLSI (Very Large Scale Integration) and ULSI (Ultra Large Scale Integration) in mobile devices. The change of the name is a reflection of the shift of the main point from "chip" to "system." You may wonder what "system" means and what the difference is compared with "chip."

Before SoC, the hardware developer considered how to enhance the performance of the components. At that time, the hardware developer, the system developer and the software developer were separated and made their own domains. In the SoC era, those domains are merging. Engineers, be they a hardware engineer or a software engineer, have to consider both hardware issues and software issues and provide a system solution to the target problem with the end application in mind.

Of course, there are many different definitions of SoC according to the viewpoint, but in this book the *system* means "a set of components connected together to achieve a goal as a whole for the satisfaction of the user." To satisfy end-user requirements, the engineer should cover various domains. With regard to the software aspect, the engineer should consider the software interface such as API or device driver, specific algorithms, and compatibility. With regard to the hardware aspect, the engineer should consider functional blocks, communication architecture to supply enough bandwidth to each functional block, memory architecture, and interface logics. Moreover, since such a complicated entity can be handled only by CAD (Computer Aided Design) tools, the engineer should have knowledge of CAD, which covers automatic synthesis of the physical layouts.

Therefore, the discipline of SoC design is intrinsically complicated and covers a variety of areas such as marketing, software, computing system and semiconductor IC design as described in Figure 1.5. SoC development requires expertise in IC technology, CAD, software, and algorithms, as well as management of extended teams and project and customer research.

Initially, the concept of SoC came from the PC bus system. By adopting the same bus architectures as those used in the PC, the processing of embedded applications was to be implemented on a single chip by assembling dedicated hard-wired logic and existing general-purpose processors. As the scale of integration and design complexity increased, the concepts of "design reuse" and "platform-based design" were born. The well-designed functional blocks could be reused in the later SoC.

However, such pre-designed functional blocks, called Intellectual Property (IP), are difficult to reuse with SoC because they were optimally developed for specific purposes, not for general-purpose utilization. In addition, since conventional buses were not suitable for the on-chip environment, there was a need to develop new

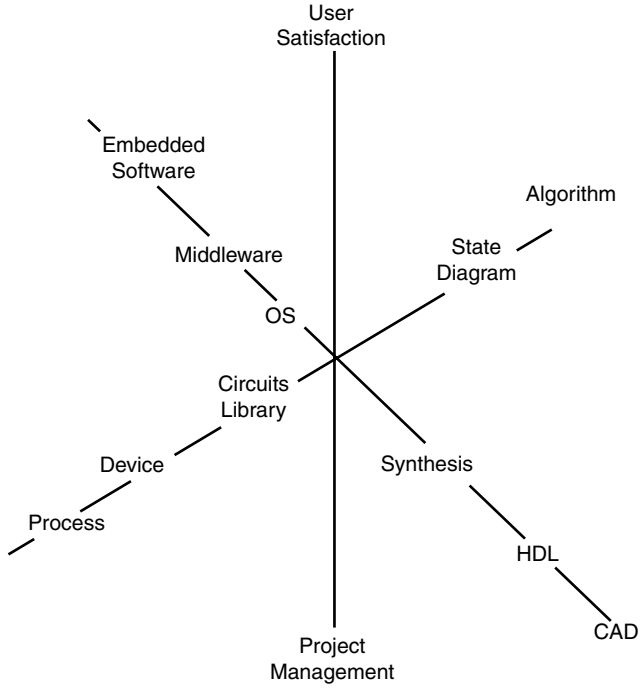


Figure 1.5 Disciplines required for the design of SoC

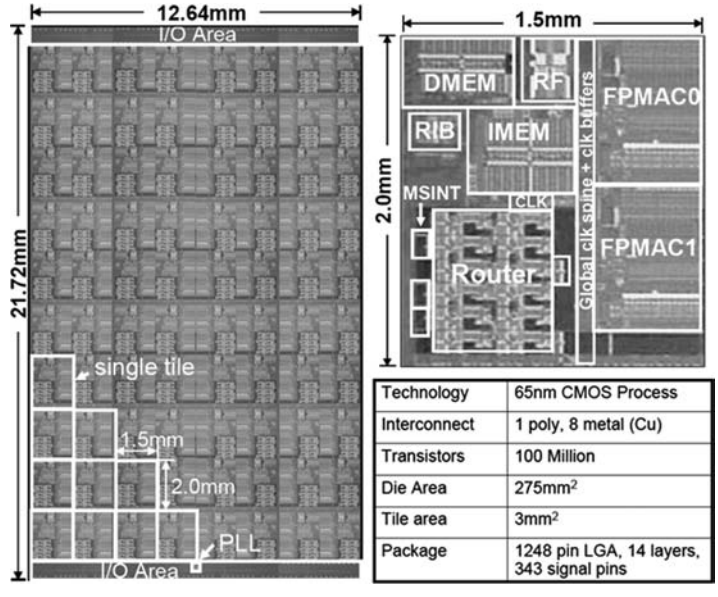
communication architecture with specific characteristics – such as wide bit width, low power, higher clock frequency, and a tailored interface. The details of design reuse and platform-based SoC design are discussed in Chapter 2.

Figure 1.6 shows an example of SoC. Intel's research chip [3] has 80 CPUs inside.

1.4 About this Book

This book describes design issues in mobile 3D graphics hardware. PC graphics hardware architecture with its shortcomings in the mobile environment is described, and several low-power techniques for mobile GPU and its real implementation are discussed.

Chapter 1 introduces the current mobile devices and mobile 3D graphics compared with desktop or arcade-type solutions. Chapter 2 discusses the general chip implementation issue, such as how to design the SoC, and includes an explanation of SoC platforms. The SoC design paradigm, system architecture, and low-power SoC design are addressed in detail. Chapter 3 deals with basic 3D graphics, the fixed-function 3D graphics pipeline, the application-geometry rendering procedure, and the programmable 3D graphics pipeline. In Chapter 4 we articulate the differences between conventional and mobile 3D graphics, and introduce the principles of mobile 3D graphics and standard mobile 3D graphics APIs.



Technology	65nm CMOS Process
Interconnect	1 poly, 8 metal (Cu)
Transistors	100 Million
Die Area	275mm^2
Tile area	3mm^2
Package	1248 pin LGA, 14 layers, 343 signal pins

Figure 1.6 Example of an SoC implementation: Intel's 80-core processor and its unit CPU. The Intel logo is a registered trademark of Intel Corporation

The design of 3D graphic processors is discussed in Chapters 5–7. Chapter 5 explains the hardware design techniques for mobile 3D graphics, such as low-power rasterizer, low-power texture unit, and several hardware schemes for low-power shaders. Chapter 6 covers the real chip implementation of mobile 3D graphics hardware. For academic architecture, KAIST RAMP architecture is introduced and the industrial architectures, SONY PSP and Imagination Technology SGX, are also described. Chapter 7 has a detailed explanation of the low-power rasterization unit with RTL code. In this chapter, readers can grasp the basic concept of how to design low-power 3D graphics processors. The future of mobile 3D graphics is very promising because people will carry more and more portable equipment in the future with high-performance displays. Finally, Chapter 8 looks at the future of mobile 3D graphics.

We also include appendices to introduce to chip design by verilog HDL The reader can run the verilog file to check the algorithms explained in the earlier chapters and get a taste of real 3D graphics chip design.

References

1 Tolga Capin, et, al., "The State of the Art in Mobile Graphics Research", IEEE Computer Graphics and Applications, Vol. 28, Issue 4, 2008, pp. 74–84.

2 Gordon E. Moore, "Cramming more components onto integrated circuits", Electronics, vol. 38, no. 8, 1965.

3 J. Held, et al, "From a Few Cores to Many: A Tera-scale Computing Research Overview," white paper, Intel Corporation, www.intel.com.

2

Application Platform

2.1 SoC Design Paradigms

2.1.1 Platform and Set-based Design

2.1.1.1 Definition of a Platform

Two steps are encountered in any design process: "planning" and "making." Certain procedures are followed when we want to perform meaningful tasks towards building a target structure. As the target structure takes on more complexity, well-established design procedures are essential. This applies in SoC design, which is strongly driven by its target applications such as multimedia and mobile communications. SoC engineers have to consider factors like quality, cost and delivery (QCD). In that sense, their design procedures naturally seek the reuse of previously developed techniques and materials at every possible design step.

In a popular English dictionary, a "system" is defined as a *set* and a *way of working* in a fixed plan with networks of components. In addition to this, SoC requires one more idea, which is the integration of components on a single semiconductor chip. So it follows that we need to focus on two concepts: the fixed plan, and integration. We can catch the concept of predetermined architecture from the fixed plan; and integration involves the network and component-based design. Considering that modern digital gadgets require not only hardware (HW) components but also software (SW) programs, we can begin to see what the "platform" means in SoC design.

The platform is a set of standalone modules that become the basis of the system. These standalone modules are pre-integrated and combine HW and SW components – we call them the "reference architectures." They are also well-verified and have well-defined external interfaces. The platform guides what designers do, and this guidance determines the design flow. The platform concept helps us to design a more complicated and less buggy system within limited QCD factors by reusing and upgrading pre-built HW and SW components.

Mobile 3D Graphics SoC: From Algorithm to Chip Jeong-Ho Woo, Ju-Ho Sohn, Byeong-Gyu Nam and Hoi-Jun Yoo
© 2010 John Wiley & Sons (Asia) Pte Ltd

In this part of the chapter we will discuss what the platform is and what it does. We will explain how the platform can be extracted from earlier design examples and how it can be used for a new design. The concept of modeling and its relationship with the platform will also be examined. We then go on to discuss the system architecture and software design in detail in the following sections.

2.1.1.2 Platformization

Sometimes, use of the word "platform" seems to be a little confused. Many engineers tend to think that a platform is a kind of restriction. However, we need to consider a platform in two respects: its philosophy and its management. By philosophy we mean how a platform is derived from the ideas, theory and history of pre-designed samples. It also encompasses how we can use the derived platform for new designs. By management we mean the directions the platform – and the designs guided from that platform – should be evolved and maintained. We should avoid trying to make a design tool such as a design wizard program while developing the platform. The platform is not intended to generate design examples automatically. Instead, it is better to approach the platform as a design methodology, which is a set-based design. That can lead us to the many benefits of design planning and our design procedures.

When developing a platform for a given design set and research area, we will try to analyze pre-designed examples and extract some common ideas in those designs. The ideas may include target design specifications with a primary feature set, external interfaces and internal architecture. After collecting these common ideas, we can make the basic standalone modules and define the platform by reusing the individual components and arranging them under categories and levels of primary features. This procedure resembles inductive reasoning, which derives general principles from particular facts and instances. In this process, it is very important to categorize the primary features and link them to each specification level (such as low-, middle-, or high-performance levels) when building the reference architectures. Actually, when we design something, we are first given the target specifications and primary feature set to be designed. The detailed architecture and design plan doesn't matter for this step. We should plan to design our target based on previous examples and theory by using previous knowledge and experience. The platform is then the collection of our design history and theories. So, categorization and arrangement of primary features are the guideline to distinguish the reference architectures in the platform.

Now we are ready for our new design. The specification and external interfaces of our new design target may contain some parts of the pre-developed platforms. Some other parts may differ. However, we can usually find one best-matched standalone module in the platforms as a reference for our next design target. The parts that are common with the reference design can be reused in the new design. Some other parts might be developed by reusing and expanding the internal architecture and interface

definitions of the reference design. This is the *set-based design approach* and resembles deductive reasoning, which generates specific facts and conclusions (our new design) from the general premises (our platform). Therefore, *platformization* can be understood as inductive and deductive reasoning, which helps us to develop a new, more complex design with very controlled and acceptable resources.

Figure 2.1 illustrates the design process. We have mentioned that the common ideas extracted from previous designs and theories contain the specification, external interface and internal architectures. In real designs, the specification and definitions of external interfaces tend to influence and decide the internal architectures to a certain extent. The definitions of internal architecture contain the following components.

Primary processing elements – What kinds of task are required and what are the related computing units?

Memory architecture – What kinds of processing result are stored for next time and how many memories are required?

Internal network – How can the processing elements and memory components be connected and interfaced with each other? And how are the internal elements connected to external interfaces?

Programmer's model – How can software developers use the HW devices to complete the functions of the target design?

In set-based design, the reference architecture is applied as a starting point for the target design. As shown in the figure, there are four options: "As-is," "Modified," "New design," and "Removed." "As-is" means the reuse of components. Since the reference architecture is not the final design output, additions and modifications are always necessary. However, the set-based design approach can help us concentrate on the updated parts and reduce design costs.

2.1.1.3 Mobile 3D Graphics Example

The operational sequence, or *pipeline*, of mobile 3D graphics consists of geometry and rendering stages, which are explained in Chapter . In this subsection, the platformization of mobile 3D graphics will be briefly described as an example of the earlier discussion.

There are many design examples in mobile 3D graphics [1–3]. Like many multimedia applications, the Advanced RISC Machines (ARM) processor family that has the reduced instruction-set computer (RISC) architecture is most widely used as a main host processor because of its good performance and low power consumption [4]. Many mobile 3D graphics designs employ the ARM architecture with appropriate hardware accelerators. These HW accelerators can be divided into fully hard-wired logic and programmable architecture. As more functionality is required,

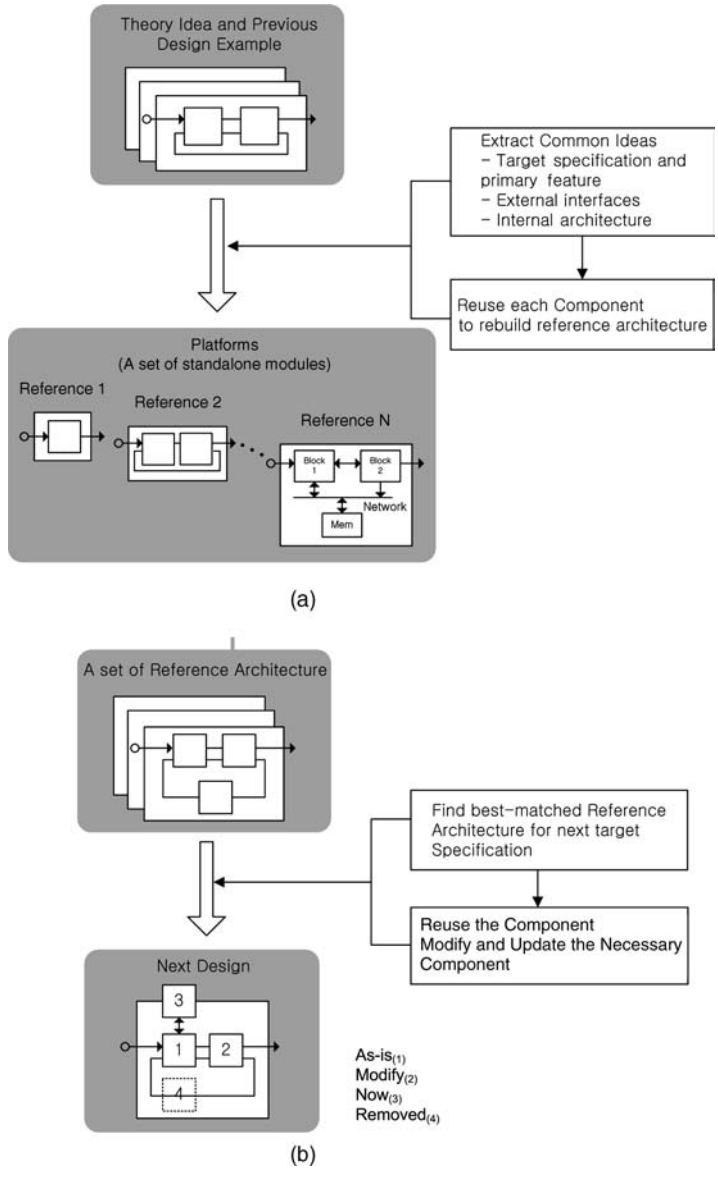


Figure 2.1 (a) Platform and (b) set-based design

more programmability is integrated. In the software part, we have the industry-standard API, OpenGL-ES, for mobile 3D graphics [5]. It is also evolving from the application of compact and efficient architecture into integrating more programmability in the next version.

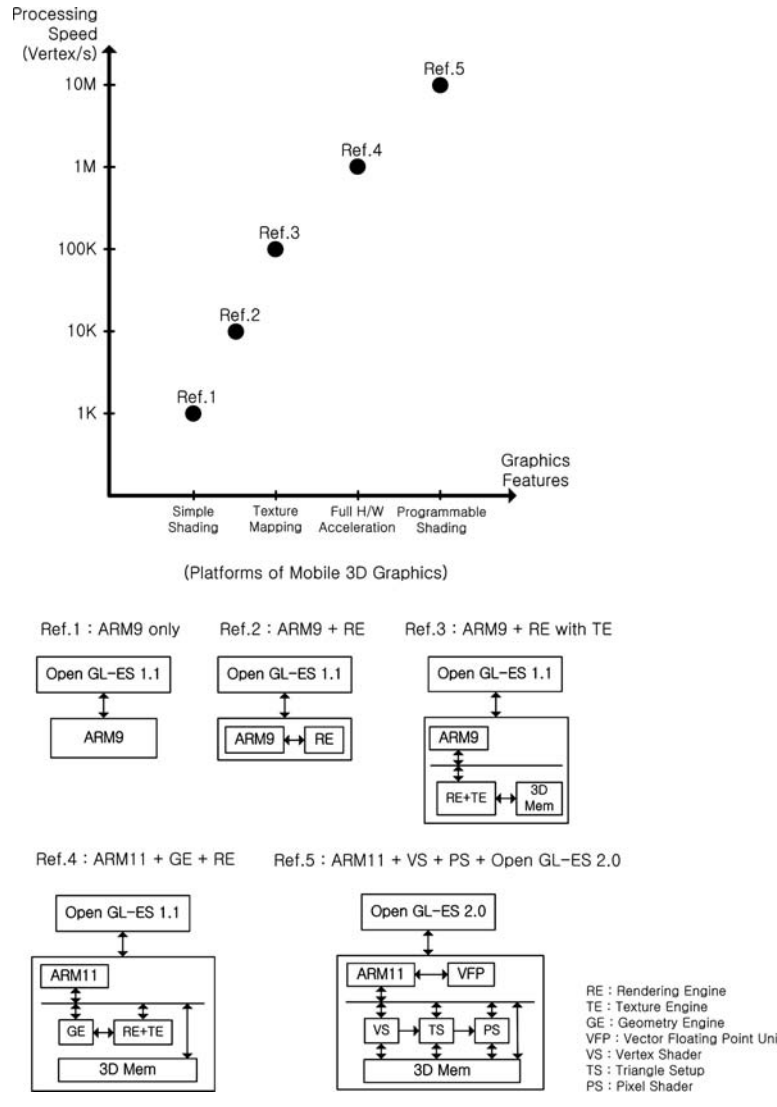


Figure 2.2 Platform of mobile 3D graphics

Figure 2.2 shows the range of graphics specifications and their reference architectures. For people wanting high-end graphics performance, programmability and full HW acceleration are necessary. In contrast, simple shading is required by some people who just want simple graphics such as the user interface of a small cellular phone. As more graphics functions are required, the processing speed must also be increased. As the target design moves towards high-end performance, more HW accelerators and programmability will be applied. The reference architecture depicted in the figure

shows typical HW building blocks and the related software OpenGL-ES library. Some HW blocks such as a rendering engine (RE) and a texture engine (TE) are reused in two more of the reference architectures. The vertex shader (VS) and pixel shader (PS) are newly introduced in the reference architecture of the highest performance range. So, when designing a new mobile 3D graphics system including HW and SW, we can decide on the reference architecture by inspecting target specifications and graphics features. Then we can complete the design by reusing, updating and optimizing the additionally necessary SW and HW components based on the chosen reference architecture.

2.1.2 Modeling: Memory and Operations

2.1.2.1 Memory and Operations

How can the platform be derived from earlier design examples? Asking this question may help us to make and use the platform for our new design more efficiently.

When we design electrical components, there are certain requirements to drive signals or store information for future use. These actions are related to memory or, in other words, state variables. If an external stimulus and internal activity do not affect any part of the internal memory contents, we can say that nothing important happened. Then, after defining memories, we can consider what types of operation can be performed on them. So, we can imagine that an electrical component actually consists of the memory and the operations. In that sense, modeling can be defined as deciding on the memory architecture and its related operations for the target components. The design of an electrical *system* can be regarded as the process that defines its memories and operations by reusing and combining sub-components that are also defined as memories and operations.

Figure 2.3 outlines the modeling and design methodology. The basic elements can be defined by their memories and operations, and the memories can be called state variables. Then we have two design methods. The first is *modular design*. This means that every element can be developed independently and reused to provide multiple functions. Many interconnections – such as serial, parallel, and feedback networks – can be implemented. So, for example, one output of element A can be fed into one input of element B. The second method is *hierarchical design*. Here, elements can be organized as parent–child or tree-like structures to permit complex functions. The state variables are newly defined and the details of internal operations are encapsulated. This process can be repeated many times, step by step. Complicated designs are made possible by combining simpler elements.

Since complex designs can be divided into sub-elements, modularly and hierarchically, we can set up reference architecture for those complex designs. The reference architectures in the platform can be built by combining or selecting necessary

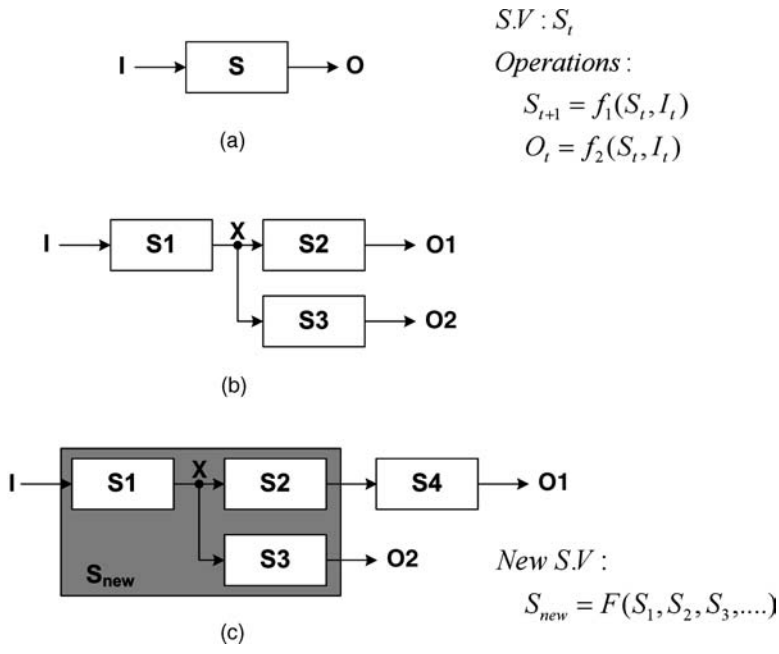


Figure 2.3 Modeling and design methodology: (a) modeling, (b) modular design, and (c) hierarchical design

sub-elements, and be reused by being combined and modified with other elements. We can update some parts of the reference design by changing the definitions of memories and operations. However, to maximize the efficiency of reuse in the reference design, we should restrict changes of input and output ports in the model of sub-elements as much as possible. This can be controlled because we know the influence of those changes by using modular and hierarchical design methods. Changing internal definitions of memories and operations does not result in changes in other parts of the whole design, and changing definitions of input and output ports can be clearly traced through the design hierarchy.

In the past, many designers have used flow charts or sequential diagrams to model their designs. However, as designs become more complex it becomes difficult to manage, update and reuse earlier designs with the flow chart method. It becomes difficult to understand the influence of changing elements. However, the sequential diagram is still useful in understating the behavior of a system when analyzing particular cases. Many design specifications are described by functional requirements, such as listening to music while viewing photographs. In that situation, the interactions of each sub-block should be clearly revealed to discover any insufficiency or bottle-neck in the whole design. These interactions are triggered in the current sub-block by events in earlier blocks. The modular and hierarchical design methods, and therefore

platform and set-based design, can help us not only to build the design but also to analyze particular cases of using the design, because the interfaces and internal architecture are clearly defined.

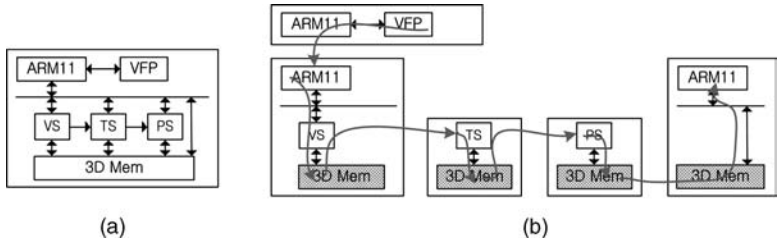


Figure 2.4 Example of use-case analysis: (a) block diagram, and (b) use-case analysis of a game

Figure 2.4 shows an example. Part (a) shows the block diagram of a mobile 3D graphics system that can perform full programmable graphics pipeline operations, including a vertex shader and a pixel shader. Part (b) illustrates a case of a game application including game logic operations and graphics operations. The game logic operations – such as game physics and artificial intelligence – are performed on the ARM11 host processor with a vector floating-point unit. Then the ARM11 commits the graphics commands into the graphics sub-system. The vertex shader is invoked first. Then a triangle setup and pixel shader follows. In this figure, note that the 3D memory block is accessed many times by multiple functional units. Finally, the ARM11 reads the final graphics results from the 3D memory. This analysis can inform us that the 3D memory block should be carefully designed for best performance.

2.1.2.2 Applications of Analog and Digital Designs

The modeling and design methodology discussed in the previous subsection can be applied to both analog and digital designs. Figure 2.5 shows examples.

In both analog and digital designs, devices manufactured with silicon materials are used – so-called semiconductor materials. The behavior of these materials is explained by physics and electromagnetic theories such as wave equations and Maxwell equations. From the viewpoint of memory and operation modeling, the electronic charges and vector fields (such as electronic and magnetic fields) are the memories. The values of those parameters represent the information carried by the materials. The governing equations define the operations performed on those memories.

In analog design, circuit elements such as field-effect transistors, resistors and capacitors are built using silicon materials. We can regard voltages and currents as new state variables, and Kirchhoff's current law (KCL) and Kirchhoff's voltage law (KVL) as new definitions of operations. Physics books describe how KCL and KVL can be deduced from Maxwell equations. Then we can build circuit blocks – such as

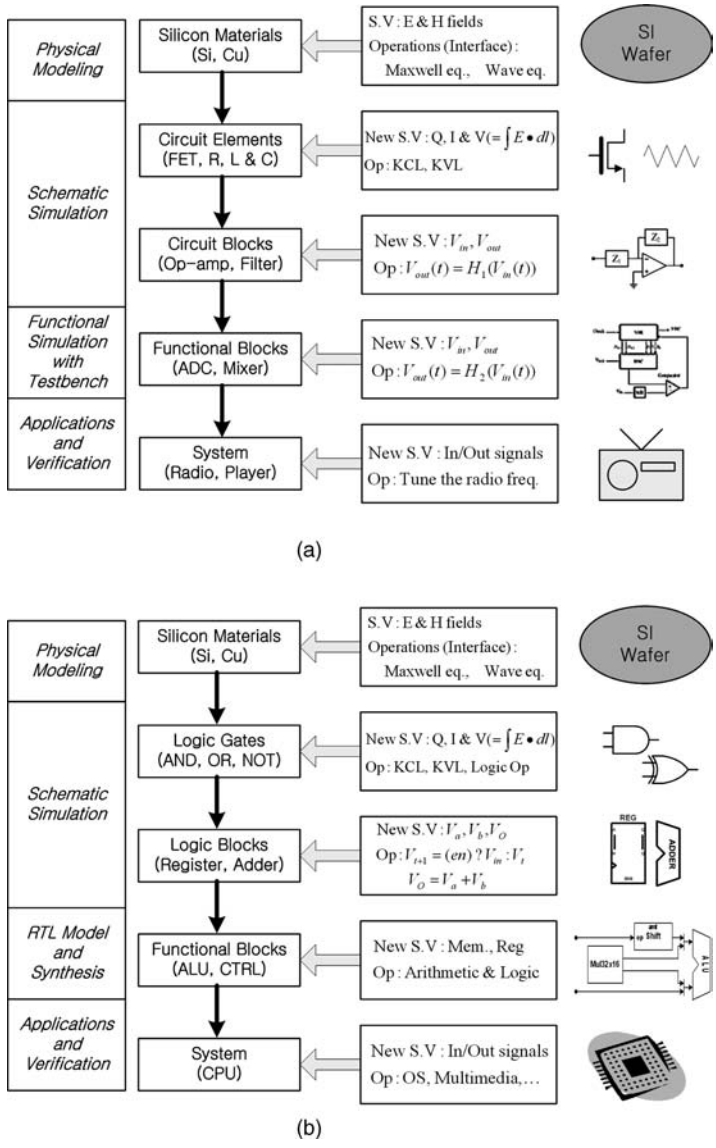


Figure 2.5 Applying (a) analog and (b) digital designs

operational amplifiers and analog filters – by using circuit elements. The voltages at important nodes and currents in important circuit paths can now be introduced as new state variables. If we repeat the same process in steps, we can build functional blocks such as analog-to-digital converters, mixers and tuner, and finally an analog radio as the product. All these processes can be understood by the modular and hierarchical design methods.

Digital design also shows the same sequences. By using silicon materials and circuit elements, we can make logic gates such as AND, OR, and NOT. Then the logic blocks such as registers and adders can be developed. Again, the voltages at important nodes can be defined as the memories. By using logic blocks, we can build functional blocks such as an arithmetic and logic unit (ALU) and a control unit, and then finally the product such as a RISC processor can be released.

The above concepts in modeling, design methodology and platform should be kept in mind during all design processes. The use of a reference architecture and set-based design results in reduced design costs and permits the development of more advanced design targets. The modular and hierarchical approach based on memory and operation modeling can make it possible to divide complex problems and keep the focus on more easily handled sub-elements.

2.2 System Architecture

2.2.1 Reference Machine and API

2.2.1.1 Definition of Reference Machine

We have described the reference architecture as the standalone module that becomes the basis of the system, and the platform as a set of those standalone modules. Now we need to step inside the reference architecture.

When deciding to implement a real system by using the reference architecture, we have to consider which parts will be mapped into software and which into hardware. The software will run on general-purpose processing elements such as RISC processors or digital signal processors (DSPs). The hardware parts can be mapped into hardware accelerators or application-specific processors with their own instruction set, such as DirectX graphics shaders. However, before beginning the separation of HW and SW parts, we have to consider how programmers or applications engineers approach the target system efficiently. Programmers require function lists that cover all possible things they can do with the system. They do not need to know how the system works internally. On the other hand, hardware or system engineers need to know how each function is actually implemented in the system. They will also want to keep the feature set within a controlled range in order to ensure design feasibility. In relation to this we can introduce two concepts concerning the reference architecture.

The first concept is the *reference machine*. It is defined as a state machine that controls a set of specific (or target) functions. So, it represents all features to be implemented. Conceptually, the reference machine is composed of datapaths, local states, global states and selectors (Figure 2.6). *Datapaths* are the computing elements that represent the operations to be performed. *Local states* are the memories storing internal information for datapaths; they are not shared with other datapaths.