SYNTHESIS OF ARITHMETIC CIRCUITS

FPGA, ASIC, and Embedded Systems

JEAN-PIERRE DESCHAMPS
University Rovira i Virgili

GÉRY JEAN ANTOINE BIOUL
National University of the Center of the Province of Buenos Aires

GUSTAVO D. SUTTER
University Autonoma of Madrid

A JOHN WILEY & SONS, INC., PUBLICATION

SYNTHESIS OF ARITHMETIC CIRCUITS

SYNTHESIS OF ARITHMETIC CIRCUITS

FPGA, ASIC, and Embedded Systems

JEAN-PIERRE DESCHAMPS
University Rovira i Virgili

GÉRY JEAN ANTOINE BIOUL
National University of the Center of the Province of Buenos Aires

GUSTAVO D. SUTTER
University Autonoma of Madrid

A JOHN WILEY & SONS, INC., PUBLICATION

Published by John Wiley & Sons, Inc., Hoboken, New Jersey. Published simultaneously in Canada.

Library of Congress Cataloging-in-Publication Data:

Deschamps, Jean-Pierre, 1945-
Synthesis of arithmetic circuits: FPGA, ASIC and embedded systems/Jean-Pierre Deschamps, Gery Jean Antoine Bioul, Gustavo D. Sutter.
p. cm.
ISBN-13 978-0471-68783-2 (cloth)
ISBN-10 0-471-68783-9 (cloth)
1. Computer arithmetic and logic units. 2. Digital electronics. 3. Embedded computer systems. I. Bioul, Gery Jean Antoine. II. Sutter, Gustavo D. III. Title.

TK7895.A65D47 2006
621.39'5 - - dc22 2005003237

Printed in the United States of America

10 9 8 7 6 5 4 3 2 1

To Marc

CONTENTS

PREFACE

From the beginnings of digital electronic science, the synthesis of circuits carrying out arithmetic operations has been a central topic. As a matter of fact, it is an activity directly related to computer development. From then on, a well-known technical discipline was born: computer arithmetic. Traditionally, the study of arithmetic circuits has been oriented toward applications to general-purpose computers, which provide the most important applications of digital circuits. However, the electronic market share corresponding to specific systems (embedded systems) is significant. It is important to point out that the huge business volume that corresponds to general-purpose computers (personal computers, servers, main frames) is distributed among a relatively reduced number of different models. Therefore the number of designers involved in general-purpose computer development is not as big as it might seem and is much less than the number of engineers dedicated to production and sales. The case of embedded systems is different. Embedded systems are circuits designed for specific applications (special-purpose devices), so a great diversity of products exist in the market, and the design effort per fabricated unit can be a lot bigger than in the case of general-purpose computers. In consequence, the design of specific computers is an activity in which numerous engineers are involved, in all type of companies—even small ones—within numerous countries.

In this book methods and examples for synthesis of arithmetic circuits are described with an emphasis somewhat different from the classic texts on computer arithmetic.

- It is not limited to the description of the arithmetic units of computers.
- Descriptions of computation algorithms are presented in a section apart from the one dedicated to their materialization or implementation by digital circuits. The development of an embedded system is an operation of hardware–software codesign for which it is not known beforehand what tasks will be executed by a microprocessor and what other tasks by specific coprocessors. For this reason, it

appeared useful to describe the algorithms in an independent manner, without any assumption on subsequent executions by an existent processor (software) or by a new customized circuit (hardware).

- A special, although not exclusive, importance has been given to user programmable devices (field programmable devices such as FPGAs), especially to the families Spartan II and Virtex. Those devices are very commonly used for the realization of specific systems, mainly in the case of small series and prototypes. The particular architecture of those components leads the designer to use synthesis techniques somewhat different from the ones applied for ASICs (application-specific integrated circuits) for which standard cell libraries exist.
- In what concern circuits description, logic schemes are presented, sometimes with some VHDL models, in such a way that the corresponding circuits can easily be simulated and synthesized.

After an introductory chapter, the book is divided in two parts. The first one is dedicated to mathematical aspects and algorithms: mathematical background (Chapter 2), number representation (Chapter 3), addition and subtraction (Chapter 4), multiplication (Chapter 5), division (Chapter 6), other arithmetic operations (Chapter 7), and operations in finite fields (Chapter 8). The second part is dedicated to the central topic—the synthesis of arithmetic circuits: hardware platforms (Chapter 9), general principles of synthesis (Chapter 10), adders and subtractors (Chapter 11), multipliers (Chapter 12), dividers (Chapter 13), other arithmetic primitives (Chapter 14), operators for finite fields (Chapter 15), and floating-point unit.

Numerous VHDL models, and other source files, can be downloaded from http://www.ii.uam.es/~gsutter/arithmetic/. This will be indicated in the text (e.g., *complete VHDL source code available*). As regards the VHDL models, they are of two types: some of them have been developed for simulation purposes only, so the working of the corresponding circuit can be observed; others are synthesizable models that have been implemented within commercial programmable components (FPGA's).

The authors thank the people who have helped them in developing this book, especially Dr. Tim Bratten, for correcting the text, and Paula Mirón, for the cover design. They are grateful to the following universities for providing them the means for carrying this work through to a successful conclusion: University Rovira i Virgili (Tarragona, Spain), University Rey Juan Carlos (Madrid, Spain), State University UNCPBA (Tandil, Argentina), University FASTA (Mar del Plata, Argentina), and Autonomous University of Madrid (Spain).

JEAN-PIERRE DESCHAMPS
University Rovira i Virgili

GÉRY JEAN ANTOINE BIOUL
National University of the Center of the Province of Buenos Aires

GUSTAVO D. SUTTER
University Autonoma of Madrid

ABOUT THE AUTHORS

Jean-Pierre Deschamps received a MS degree in electrical engineering from the University of Louvain, Belgium, in 1967, a PhD in computer science from the Autonomous University of Barcelona, Spain, in 1982, and a PhD degree in electrical engineering from the Polytechnic School of Lausanne, Switzerland, in 1983. He has worked in several companies and universities. He is currently a professor at the University Rovira i Virgili, Tarragona, Spain. His research interests include ASIC and FPGA design, digital arithmetic, and cryptography. He is the author of six books and about a hundred international papers.

Géry Jean Antoine Bioul received a MS degree in physical aerospace engineering from the University of Liège, Belgium. He worked in digital systems design with PHILIPS Belgium and in computer-aided industrial logistics with several Fortune-100 U.S. companies in the United States, and Africa. He has been a professor of computer architecture in several universities mainly in Africa and South America. He is currently a professor at the State University UNCPBA of Tandil (Buenos Aires), Argentina, and a professor consultant at the Saint Thomas University FASTA of Mar del Plata (Buenos Aires), Argentina. His research interests include logic design and computer arithmetic algorithms and implementations. He is the author of about 50 international papers and patents on fast arithmetic units.

Gustavo D. Sutter received a MS degree in Computer Science from the State University UNCPBA of Tandil (Buenos Aires), Argentina, and a PhD degree from the Autonomous University of Madrid, Spain. He has been a professor at the UNCPBA, Argentina and is currently a professor at the University Autonoma of Madrid, Spain. His research interests include ASIC and FPGA design, digital arithmetic, and development of embedded systems. He is the author of about 30 international papers and communications.

1

INTRODUCTION

The design of embedded systems, that is, circuits designed for specific applications, is based on a series of decisions as well as on the use of several types of development techniques. For example:

- Selection of the data representation
- Generation or selection of algorithms
- Selection of hardware platforms
- Hardware–software partitioning
- Program generation
- New hardware synthesis
- Cosimulation, coemulation, and prototyping

Some of these activities have a close relationship with the study of arithmetic algorithms and circuits, especially in the case of systems including a great amount of data processing (e.g., ciphering and deciphering, image processing, digital signature, biometry).

1.1 NUMBER REPRESENTATION

When using general-purpose equipment, the designer has few possible choices concerning the internal representation of data. He must conform to some fixed

Synthesis of Arithmetic Circuits: FPGA, ASIC, and Embedded Systems
By Jean-Pierre Deschamps, Géry J. A. Bioul, and Gustavo D. Sutter

and predefined data types such as *integer*, *floating-point*, *double precision*, and *character*. On the contrary, if a specific system is under development, the designer can choose, for each data, the most convenient type of representation. It is no longer necessary to choose some standard fixed-point or floating-point numeration system. Nonstandard specific formats can be used. In Chapter 3 the main number representation methods will be defined.

1.2 ALGORITHMS

Every complex data processing operation must be decomposed into simpler operations — the computation primitives — executable either by the main processor or by some specific coprocessor. The way the computation primitives are used in order to perform the complex operation is what is meant by *algorithm*. Obviously, knowledge of algorithms is of fundamental importance for developing arithmetic procedures (software) and circuits (hardware). It is the topic of Chapters 4–8.

1.3 HARDWARE PLATFORMS

The selection of a hardware platform is based on the answer to the following question. How do we get the desired behavior at the lowest cost, while fulfilling some additional constraints? As a matter of fact, the concept of cost must be carefully defined in each particular case. It can cover several aspects: for example, the unit production cost, the nonrecurring engineering costs, and the implicit cost for a late introduction of the product to the market. Some examples of additional technical constraints are the size of the system, its power consumption, and its reliability and maintainability.

For systems requiring little data processing capability, *microcontrollers* and low-range *microprocessors* can be the best choice. If the computation needs are greater, more powerful microprocessors, or even *digital signal processors* (DSPs), should be considered. This type of solution (microprocessors and DSPs) is very flexible as the development work mainly consists in generating programs.

For getting higher performances, it may be necessary to develop specific circuits. A first option is to use a programmable device, for example, a *field-programmable gate array* (FPGA). It could be an interesting option for prototypes and small series. For greater series, an *application-specific integrated circuit* (ASIC) should be developed. ASIC vendors offer several types of products: for example, *gate arrays*, with relatively small prototyping costs, or *standard cell libraries*, integrating a complete *system-on-chip* (SOC) including processors, program memories, data memories, logic, macrocells, and analog interfaces.

A brief presentation of the most common hardware platforms is given in Chapter 9.

1.4 HARDWARE–SOFTWARE PARTITIONING

The hardware–software partitioning consists of deciding which operations will be executed by the central processing unit (the software) and which ones by specific coprocessors (the hardware). As a matter of fact, the platform selection and the hardware–software partitioning are tightly related operations. For systems requiring little data processing capability, the whole system is implemented in software. If higher performances are necessary, the noncritical operations, as well as control of the operation sequence, are executed by the central processing unit, while the critical ones are implemented within specific coprocessors.

1.5 SOFTWARE GENERATION

The operations belonging to the software block of the chosen partition must be programmed. In Chapters 4–8 the algorithms are presented in an Ada-like language that can easily be translated to C or even to the assembly language of the chosen microprocessor.

1.6 SYNTHESIS

Once the hardware–software partition has been defined, all the tasks assigned to the specific hardware (FPGA, ASIC) must be translated into circuit descriptions. Some important synthesis principles and methods are described in Chapter 10. The synthesis of arithmetic circuits, based on the algorithms of Chapters 4–8, is the topic of Chapters 11–15, and an additional chapter (16) is dedicated to the implementation of floating-point arithmetic.

1.7 A FIRST EXAMPLE

Common examples of application fields resorting to embedded solutions are cryptography, access control, smart cards, automotive, avionics, space, entertainment, and electronic sales outlets. In order to illustrate the main steps of the design process, a small digital signature system will now be developed (complete assembly language and VHDL code available).

1.7.1 Specification

The system under development (Figure 1.1) has three inputs,

- `character` is an 8-bit vector.
- `new_character` is a signal used for synchronizing the input of successive characters.
- `sign` is a control signal ordering the computation of a *digital signature*.

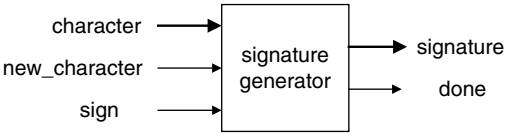

Figure 1.1 System under development.

and two outputs,

- `done` is a status variable indicating that the signature computation has been completed,
- `signature` is a 32-bit vector, namely, the signature of the message.

The working of the system is shown in Figure 1.2: a sequence $c_1, c_2, \ldots, c_n$ of any number n of characters (the message), synchronized by the signal `new_character`, is inputted. When the `sign` control signal goes high, the `done` flag is lowered and the signature of the message is computed. The `done` flag will be raised as soon as the signature s is available.

In order to sign the message two functions must be defined:

- a `hash` function associating a 32-bit vector (the `summary`) to every message, whatever its length;
- an `encode` function computing the signature corresponding to the summary.

The following (naive) hash function is used:

Algorithm 1.1 Hash Function

```
summary:=0;
while not(end_of_message) loop
   get(character);
   a:=(summary(7 downto 0)+character) mod 256;
   summary(23 downto 16):=summary(31 downto 24);
   summary(15 downto 8):=summary(23 downto 16);
```

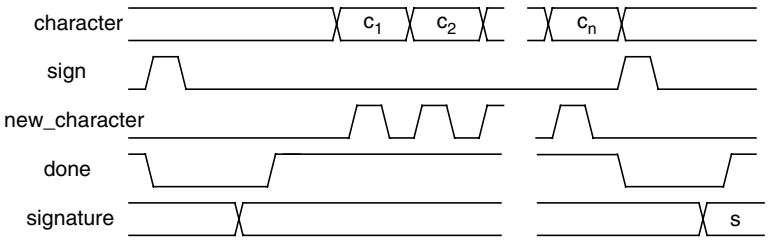

Figure 1.2 Input and output signals.