Adv. Radio Sci., 4, 231–236, 2006
www.adv-radio-sci.net/4/231/2006/
© Author(s) 2006. This work is licensed
under a Creative Commons License.

Advances in Radio Science

Configurable multiplier modules for an adaptive computing system

O. A. Pfänder[1], H.-J. Pfleiderer[1], and S. W. Lachowicz[2]

[1]Microelectronics Department, University of Ulm, Germany
[2]School of Engineering and Mathematics, Edith Cowan University, Perth, Western Australia

Abstract. The importance of reconfigurable hardware is increasing steadily. For example, the primary approach of using adaptive systems based on programmable gate arrays and configurable routing resources has gone mainstream and high-performance programmable logic devices are rivaling traditional application-specific hardwired integrated circuits. Also, the idea of moving from the 2-D domain into a 3-D design which stacks several active layers above each other is gaining momentum in research and industry, to cope with the demand for smaller devices with a higher scale of integration. However, optimized arithmetic blocks in course-grain reconfigurable arrays as well as field-programmable architectures still play an important role. In countless digital systems and signal processing applications, the multiplication is one of the critical challenges, where in many cases a trade-off between area usage and data throughput has to be made. But the a priori choice of word-length and number representation can also be replaced by a dynamic choice at run-time, in order to improve flexibility, area efficiency and the level of parallelism in computation. In this contribution, we look at an adaptive computing system called 3-D-SoftChip to point out what parameters are crucial to implement flexible multiplier blocks into optimized elements for accelerated processing. The 3-D-SoftChip architecture uses a novel approach to 3-dimensional integration based on flip-chip bonding with indium bumps. The modular construction, the introduction of interfaces to realize the exchange of intermediate data, and the reconfigurable sign handling approach will be explained, as well as a beneficial way to handle and distribute the numerous required control signals.

1 Introduction

Today's ever-increasing demand for computing power in virtually all application sectors, including the emerging photonics arena, is placing heavy demands on current integrated systems. Crucial scenarios are e.g. modern wired and wireless communication over highly complex networks, processing of content-rich multimedia, and mobile applications where area and power are primary considerations. New ways to overcome today's technological barriers of silicon integrated circuits (ICs) are emerging, aiming to reduce both non-recurring engineering costs and time-to-market periods, and also considering production factors like re-usage of lithography masks. Reconfigurable computing has gone mainstream, with the many forms of reconfigurable logic gradually rivaling traditional application-specific ICs. Thanks to the progress in IC manufacturing techniques and the shrinking feature sizes, it has become possible to integrate a multitude of different functions into one single chip, while still meeting restrictions like package size or footprint. Thus, a whole system incorporating multiple special functions that had to be realized with multiple specialized separate ICs can now be implemented in a single system-on-chip (SoC). Going one step further from SoC, the idea of systems-in-a-package (SiP) emerges, when high-end concepts such as flip-chip or chip stacks are used to reach an even higher level of design integration and minimization. While the IC density continues to grow, another key feature is becoming more and more essential for many of today's widely spread application scenarios; that is, flexibility and/or configurability. There is a paradigm change in the world of microelectronics, because the term hardware does no longer stand for the total opposite of software. The notion of ICs offering options to configure them, even by the end customer, is nothing but intriguing. As feature sizes are constantly shrinking and the design complexity is continually increasing, design methodologies are forced to develop at an increasing rate and the designer faces many new obstacles. Novel 3-D integration systems such as 3-D SoC (Joyner et al., 2004) or 3-D-SoftChip (Eshraghian et al., 2003), targeted to satisfy the enormous demand for more computation throughput by effectively manipulating the functionality of hardware primitives through

Correspondence to: O. A. Pfänder
(oliver.pfaender@uni-ulm.de)

Published by Copernicus GmbH on behalf of the URSI Landesausschuss in der Bundesrepublik Deutschland e.V.

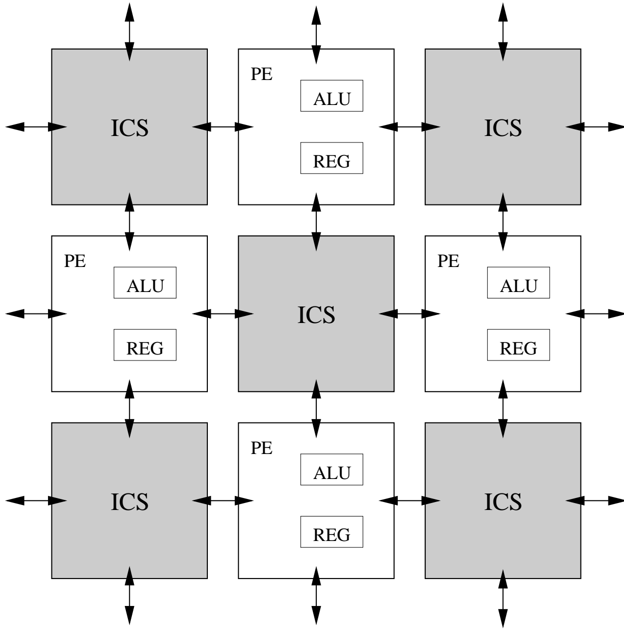

Fig. 1. The starting point is a 2-dimensional course-grain soft reconfigurable array of processing elements (PEs) and intelligent configurable switches (ICS).

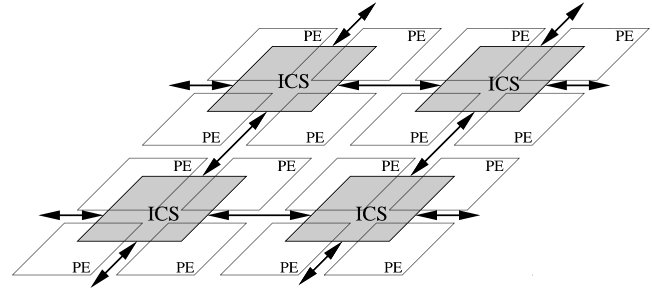

Fig. 2. Functional separation into a sea of PEs and an array of ICS blocks. Each ICS connects a group of 4 PEs using vertical wiring resources.

vertical integration of two 2-D chips, are becoming an attractive solution to combat the rising requirements for interconnect wires (Davis et al., 2005).

This paper is organized as follows: Sect. 2 reviews the 3-D-SoftChip adaptive computing system and its approach to architectural mapping and vertical integration, Sect. 3 concentrates on the design of embedded multiplier blocks into the 3-D-SoftChip's processing elements, and Sect. 4 concludes the paper and gives suggestions for further investigation.

2 3-D-SoftChip adaptive computing system

The concept of the 3-D-SoftChip adaptive computing system is located in the intersection of three major fields of research: Modern very large scale integration (VLSI) in deep sub-micron IC design, reconfigurable hardware, and 3-D integration. This section will at first highlight the basic method of transforming a 2-dimensional course-grain reconfigurable array into a sophisticated 3-dimensional integrated system, then mention the architectural mapping approach and also present the vertical integration method using indium bumps.

2.1 2-D to 3-D transformation

The starting point for the transformation into the 3-D-SoftChip system is a 2-dimensional course-grain soft reconfigurable array of processing elements as depicted in Fig. 1. Each processing element (PE) contains arithmetic and logical functional blocks combined in an arithmetic logic unit (ALU) and also buffer elements (registers, REG). The ALUs are designed for increased flexibility in order to support variable data word-lengths for various types of computation. The overall layout is very much like a checkerboard, with intelligent configuration switches (ICS) accessing the surrounding PEs through a next-neighbor interconnect scheme. Thus, each ALU is able to communicate with all adjacent ALUs through the ICS which serves as a cross point switch with embedded memory.

Looking at the checkerboard layout of Fig. 1, two functional layers are noticeable: First, the array of PEs, and second, a superordinate array of ICS blocks that covers multiple management, interface and I/O functions. However, the area usage of a complete 2-dimensional array is considerable, since there are as many ICS blocks as there are PEs and the area of the respective blocks can differ to a great extent, depending on the provided features. The area usage can be optimized if these two functional layers are separated in space. The proximate solution would be to have an arrangement in groups – one group of PEs in one part of the chip and a group of ICS blocks in another, like separating the black squares from the white squares on a checkerboard. But this solution is impracticable, since it would require an oversized amount of interconnect circuitry and wire space to connect the two groups. A better way to cope with this situation is to think 3-dimensional: By using a novel method for flip-chip bonding, as will be explained in Subsect. 2.2, the two functional layers can be realized as two separate chips that are integrated vertically. To share e.g. memory or outbound routing resources, four ICSs are merged, and the resulting blocks are placed on a second chip. On the first chip, the PEs can now move together and the area can be compacted significantly. The data exchange now happens not only internally on each of the two chips, but also between them, as will be pointed out later. Figure 2 illustrates the basic principle of transforming the planar 2-D architecture of Fig. 1 into two separate functional layers.

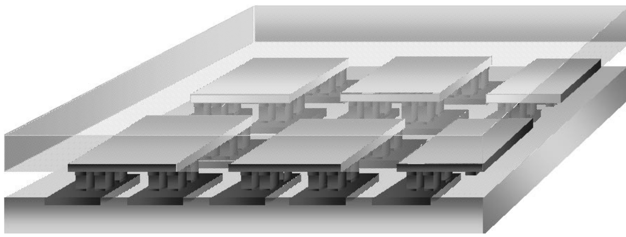

Fig. 3. To achieve a vertical 3-dimensional integration of the upper ICS chip and the lower CAP chip, indium bumps are used for flip-chip bonding.

2.2 Vertical integration

As described in the previous subsection, the sea of PEs is now separated from the array of ICS blocks. The lower chip that now contains the PEs only is called the configurable array processor (CAP). The transformation of the planar 2-D architecture through the approach depicted in Fig. 2 into two vertically integrated chips, namely the upper ICS chip and the lower CAP chip, is illustrated in Fig. 3.

Each of the two chips has an array of aluminum pads on the top metallization layer, each with a size of $10\mu\mathrm{m} \times 10\mu\mathrm{m}$. The pad pattern on the lower chip is a mirror image of the upper chip's pattern. When the chips are placed face-to-face on top of each other, this creates an interface for a vertical connection between them, and the actual connection is realized by depositing indium bumps onto the aluminum pads on the upper and lower chip, respectively. Creating the bumps comprises the following processing steps: After the silicon substrate has been oxidized and the aluminum pads have been patterned the normal way, there is an additional photoresist coating and patterning step involved to cover and protect the pads' surrounding die surface. Then, titanium (as a diffusion inhibitor), gold (as a contact layer) and indium get evaporized and thus brought onto the aluminum pads. After the lift-off, there is an extra reflow process to make use of the surface tension of the low-melting indium material, in order to transform the block-shaped deposits into bumps with an increased height-to-width ratio. This ensures a certain self-adjusting capability once the two chips get connected (Pfänder et al., 2005). The bumps have a diameter of about 7.5 μm after reflow. The upper ICS chip is flipped and bonded face-down to the lower CAP chip, then the space between is filled with a curing material to ensure a mechanical stability.

2.3 Architectural mapping

The upper chip's function (ICS) is to act as a massively parallel cross point switch as well as a parallel interconnected buffer memory, to allow for a very high-speed data manipulation within the plane. The lower chip (CAP) is a highly parallel array of soft-programmable processing elements, which is capable of carrying out complex calculation tasks directly on data stored in the CAP plane or – using the 3-D interconnect – stored in the top plane. Each of the PEs includes its own embedded register file, along with functional ALU blocks, glue logic and instruction decoding circuitry. Software-programmed instructions are forwarded globally to all processors from on-chip RAM. Even transforms and other processing tasks may be carried out according to embedded software instructions on the highly parallel sea-of-PE array. Two levels of hierarchy within the CAP architecture facilitate the configuration of the ALU's word-length: While at the first level, four processors and one ICS are utilized, this basic group can communicate with adjacent groups at the second level. The interconnection between the parallel array processors is provided by a bus architecture for rapid extraction or insertion of data. Due to the programmable nature of the CAP, the system is highly flexible, and as a result of the vertical interconnects and highly parallel configurable architecture, the efficiency is improved compared to the 2-D planar architecture that was the starting point. On the array level in the system, addressing will follow a switched bus architecture. The multi-tasking requirement of the system introduces a particular need for formulating the architecture that is driven by a variable and configurable number of bits. Therefore, the identification of generic primitives such as adders and multipliers suitable for word-length expansion becomes an important task to realize a higher system flexibility without compromising performance.

3 Configurable multiplier modules

3.1 Heterogeneous processing elements in the CAP

For generic ALU functions, the 3-D-SoftChip's standard PE is optimized for bit-level computation. However, when it comes to frequently used arithmetic functions – e.g. in signal processing applications where the data path contains many multiplication steps involving constantly changing filter coefficients – it becomes more efficient to implement fixed-wired and dedicated functional blocks instead of combining the configurable logic (Haynes et al., 1999). This approach is now common practice in modern high-performance FPGA designs, like the Xilinx Virtex-II Pro that uses 18×18 bit multiplier hard IP blocks strategically placed on the chip (Xilinx, 2005). Introducing a processing accelerator PE with embedded special blocks (Pfänder et al., 2005), such as a barrel shifter or an accumulator/subtractor unit, helps to reduce the data transfer and increases the computation efficiency. There is only a medium flexibility trade-off compared with a complete homogeneous type PE array.

The processing accelerator PE (PA-PE) thus provides increased performance for specific tasks thanks to its dedicated special-purpose functions. In the following, we will concentrate on the multiplication, since it is one of the most impor-

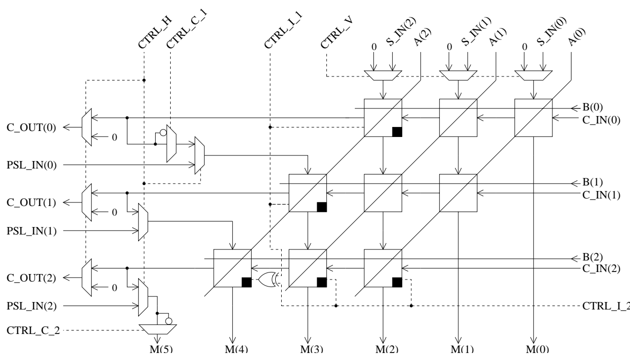

Fig. 4. A parallel-parallel modified Baugh-Wooley array multiplier with multiplexer-based interfaces

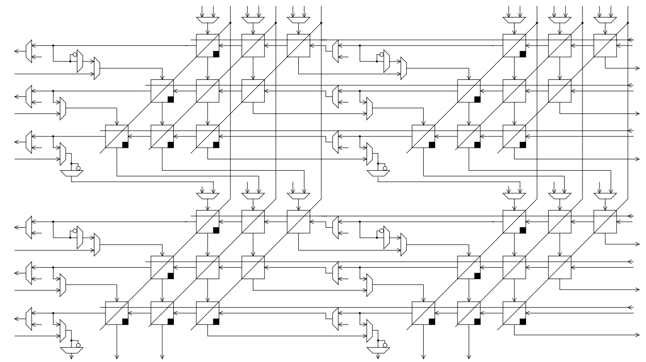

Fig. 5. A concatenation of four 3×3 bit multiplier modules creates a 6×6 bit superior multiplier. (Control signals not shown for better overview)

tant and widely used mathematical operations and requires a large amount of hardware resources when implemented in a straight-forward way. Instead of that, we have come up with a modular multiplier design that enables the system to compute at a configurable word-length, which will be explained in the following.

3.2 Basic scheme and module characteristics

With the concept of arithmetic operations accommodating word-lengths that are configurable at run-time, a digital hardware circuit can adapt to changing accuracy requirements easily and fast. With little overhead, new extensions can help to improve the embedding of multiplier architectures in the surroundings of the 3-D-SoftChip system. As was presented in Pfänder and Pfleiderer (2004), we have designed different variants of multiplier modules that rely on a multiplexer-based connectivity extension for an intermediate data exchange. The general idea is to tap the flow of carry and output bits in specific positions inside an array of basic cells. These cells compute the partial product and provide sum and carry, and with the help of multiplexers and corresponding control signals, the module's behavior can be directly influenced at run-time. The flexible word-length architecture and its dynamic reconfigurability provides either a higher throughput at low levels of precision or a higher precision by grouping multiple elements together, thus greatly increasing the design's efficiency. Also, since every module is a fully functional multiplier itself, a high parallelism can be achieved when all modules compute separately.

The basic modular building block is an $n \times n$ bit multiplier. By concatenating $m \times m$ of these uniform blocks through the use of interconnect resources, in our case also involving the vertical inter-chip interface, a superior $(m \times n) \times (m \times n)$ bit multiplier is formed that enables the computation at flexible precision in steps of n. The input operand word-lengths of the multiplier elements do not necessarily have to be equal, a scheme of $n_1 \times n_2$ bit with $n_1 \neq n_2$ is also possible.

3.3 Realization options

Starting from an unchanged parallel-parallel array multiplier core (Hwang, 1979), there are different expansion steps required in order to achieve the desired connectivity options. Here, since we are dealing with signed numbers, this demands extra sign handling circuitry. An architecture comparison in terms of hardware usage and complexity has shown that the modified Baugh-Wooley design (Baugh and Wooley, 1973) offers the least overhead of the proposed two's complement multipliers, since there is no need for a sign extension (Pfänder and Pfleiderer, 2004). The modified design incorporates the data exchange interfaces mentioned in the previous subsection and the possibility to handle different number representations, due to the modified and configurable basic cells in specific array positions. It can handle the following number systems: Unsigned (when the special cells are configured to act without negating the partial product), Signed-Magnitude (when the sign bit is calculated externally, namely utilizing an XOR gate) and Two's Complement (Pfänder et al., 2005).

Figure 4 shows an $n=3 \Rightarrow 3 \times 3$ bit parallel array multiplier as an example, and Fig. 5 represents a concatenation of four modules to form an $(m=2;\ n=3) \Rightarrow 6 \times 6$ bit multiplier. The control signals are used to control the interfaces and also the configurable partial product inversion step in the special basic cells, in order to obey the mathematical requirements given in Baugh and Wooley (1973). The signals are mapped according to the position in the concatenation and the desired number system using a control decoder in order to save I/O and interconnect resources. Thus, a part of the ICS control logic is transferred onto the CAP plane and integrated into the multiplier modules.

The identical operand scheme that builds the foundation of the parallel-parallel array multiplier can also be implemented in a serial-parallel fashion as depicted in Fig. 6. However, the control signals and also the correcting term must be provided as serial-input data words (Bermak et al., 1997).

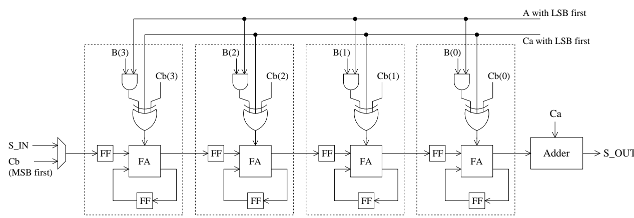

Fig. 6. The modified Baugh-Wooley multiplier can also be realized in a serial-parallel way. The first multiplicand and the control signal vector are fed in serially.

3.4 Comparison of multiplication schemes

Looking at the schemes shown in Figs. 4 and 6, it becomes evident that the number of basic cells N is a function of the word-length. For the parallel array $N \propto n^2$ and for the serial-parallel multiplier $N \propto n$. Note that the serial-parallel multiplier's alleged area advantage is compensated by the expenses for the shift registers needed to store one input operand and the control vector. However, both options have in common that considerable overhead to enable the connectivity is entailed, for example 40% at $n = 4$ for the parallel array (Pfänder and Pfleiderer, 2004). It is not sufficient to judge from the schematic hardware overhead alone; for a fair comparison, the following aspects have to be considered additionally:

- Area usage – depending on the available technology and cell library etc.
- Computation time – since the serial-parallel approach uses $2n$ clock periods
- Data throughput

In order to increase the data throughput, a fully pipelined parallel array is possible, but more than $5n^2$ extra registers would be necessary, resulting in even higher area penalty and increasing the complexity of a processing element. Thus, the decision which approach to use depends on a multitude of parameters. The range of applications plays the dominant role by asserting the claims for throughput and speed, and also other restrictions in terms of area usage and power consumption may apply. In general, the connectivity option requires a considerable extra amount of circuitry in each processing element.

4 Conclusion

In this paper, we have reviewed the 3-D-SoftChip adaptive computing system and highlighted the 2-D to 3-D transformation and its architectural mapping. To achieve a high level of integration, a novel flip-chip bonding technique based on indium bumps on aluminum pads is used to build a system-in-a-package. The indium bumps enable a 3-dimensional routing between the lower configurable array processor chip and the upper intelligent configurable switch chip. Looking at the heterogeneous processing elements located on the lower CAP chip, we have presented configurable embedded multiplier modules for accelerated processing. These modules are based on a modified Baugh-Wooley multiplier and expanded by multiplexer-based data exchange interfaces to provide a connectivity option. Both parallel-parallel and serial-parallel array multiplication schemes are possible, and three different number systems – namely unsigned, signed-magnitude and two's complement – can be handled. By connecting multiple multiplier elements together via 2-D or 3-D interconnect resources, a superior multiplier computing at an increased word-length is formed. Thus, the multiplication word-length can be chosen at run-time by dynamically realizing a concatenation of separate modules. There is a significant amount of extra hardware needed to make the dynamic array arrangement possible. But when the word-length of an optimized arithmetic unit is not fixed but can be chosen dynamically at run-time, this opens the door for multi-precision algorithms as well as a massively parallel and efficient usage of resources in the 3-D-SoftChip system.

As an outlook on future work, the impact of a certain choice of processing element architecture on the actual area usage as a function of the specific technology parameters will become necessary. Then, more detailed conclusions about hardware overhead, area efficiency and required interconnect resources can be made. A refinement of the architecture concepts as well as a design space exploration in consideration of the high-potential 3-D integration approach using indium bumps are currently in the process.

References

Baugh, C. R. and Wooley, B. A.: A Two's Complement Parallel Array Multiplication Algorithm, IEEE Trans. Computers, C-22, 1045–1047, 1973.

Bermak, A., Martinez, D., and Noullet, J.-L.: High-Density 16/8/4-bit Configurable Multiplier, IEE Proc. Circuits Devices Systems, 144, 272–276, 1997.

Davis, W. R., Wilson, J., Mick, S., Xu, J., Hua, H., Mineo, C., Sule, A. M., Steer, M., and Franzon, P. D.: Demystifying 3D ICs: The Pros and Cons of Going Vertical, IEEE Design & Test of Computers, 22, 498–509, 2005.

Eshraghian, S., Lachowicz, S., and Eshraghian, K.: Ultra High Bandwidth Image and Data Processing using 3-D Vertically Integrated Architectures, Proceedings of the SCI 2003, Orlando, FL, X, 189–195, 2003.

Haynes, S. D., Ferrari, A. B., and Cheung, P. Y. K.: Flexible Reconfigurable Multiplier Blocks suitable for enhancing the Architecture of FPGAs, Proceedings of the IEEE 1999 Custom Integrated Circuits Conference, San Diego, CA, 191–194, 1999.

Hwang, K.: Computer Arithmetic – Principles, Architecture, and Design, John Wiley and Sons, New York, 1979.

Joyner, J. W., Zarkesh-Ha, P. J., and Meindl, J. D.: Global Interconnect Design in a Three-Dimensional System-on-chip, IEEE

Transactions on VLSI systems, 2004.
Pfänder, O. A. and Pfleiderer, H.-J.: Dynamische Rekonfiguration von arithmetischen Einheiten auf Bitebene, Advances in Radio Science 2004, Miltenberg, Germany, 319–323, 2004.
Pfänder, O. A., Lachowicz, S. W., and Pfleiderer, H.-J.: Flexible Multiplier Blocks for Accelerated Processing in a 3D-SoftChip Adaptive Computing System, Proceedings of the IFIP VLSI-SoC 2005, Perth, Western Australia, 485–491, 2005.
Xilinx: Virtex™-II Platform FPGAs and Product Specification, Xilinx Document DS031 (v3.4), 2005.