14

Conclusions

14.1 Introduction

The aim of this book has been to cover many of the techniques needed to create FPGA solutions for signal and data processing systems. Interest in FPGAs for such systems has grown since the first edition as vendors have targeted this market and introduced innovations into their technology offerings to make them more attractive. This has included dedicated DSP blocks typically comprising 18-bit MAC blocks and increased memory units but also increased parallelism in terms of resources and the ability to employ pipelining to improve speed of operation. To support this activity, FPGA companies have created a range of IP cores (for customers to use in their designs) and design tools to create such implementations.

The main attraction of using FPGAs is that the resulting designs provide a very high quality of performance, particularly if MSPS/mm^2/W is considered. This is possible as the designer is able to create an architecture which is a good match to the algorithmic needs, rather than striving to map the requirements onto a fixed architecture as is the case in microcontrollers, DSP microprocessors or GPUs, even though vendors increasingly offer multicore platforms. The main design challenge is to create a suitable architecture for the algorithmic requirements.

The creation of this suitable FPGA architecture comes from employing the right level of parallelism and pipelining to match initially the throughput rate and then area and power consumption requirements. The key approach taken in this book has been to derive an efficient circuit architecture which successfully utilizes the underlying resources of the FPGA to best match the computational and communication requirements of the applications. This was demonstrated using simple design examples such as FIR, IIR and lattice filer structures in Chapter 8 as well as more complex examples such as the fixed beamformer in Chapter 10 and the adaptive beamformer in Chapter 11.

The purpose of this chapter is to give some attention to emerging issues and provide some insight into future challenges for FPGAs. In Section 14.2, attention is given to the changes in design methods as FPGA architectures have emerged. The rest of the chapter considers a range of issues likely to be important in the future. Firstly, more consideration is given in Section 14.3 to the use of FPGAs in Big Data applications and

FPGA-based Implementation of Signal Processing Systems,
Second Edition. Roger Woods, John McAllister, Gaye Lightbody and Ying Yi.

the implications for FPGAs in high-performance computing. Connected to this is the need to allow FPGAs to be more effectively integrated in future computing systems, and this is considered in Section 14.4. In Sections 14.5 and 14.6, the key issues of floating-point arithmetic and memory architectures are then covered.

14.2 Evolution in FPGA Design Approaches

The emergence of a dedicated DSP block as outlined in Chapter 5 has now simplified the mapping of design functions. This is particularly relevant for fixed-point DSP systems with wordlengths from 8 to 12 bits (a core target market for FPGAs) as these designs readily map into the 18-bit DSP blocks if wordlength growth has been addressed. The use of pipelining is easily achievable through use of programmable pipeline registers in the DSP blocks and the plethora of scalable resisters in the main programmable logic fabric.

Moreover, since the first edition of this book, there have been a number of innovations in FPGA design, primarily focused around design tools. The ability to trade off levels of parallelism and pipelining has been encapsulated to some extent within FPGA vendor synthesis tools such as the Xilinx Vivado, where the starting position is a C description; this is clearly aimed at engineering companies as C is key design language. The Altera perspective has been to start with an OpenCL description and then use the conventional FPGA place and route implementation tools to produce the final bit files for programming the FPGA.

There has been a major growth in the availability of soft IP cores such as parameterized HDL cores for a range of telecommunications, signal and image processing applications, as well as dedicated memory interface circuitry and soft processor cores. The availability of commercial cores through the Design & Reuse website (http://www.design-reuse.com/), which has 16,000 IP from 450 vendors, and open source cores from the OpenCores website (opencores.org) provides a strong "plug-and-play" design ethos to system design; this will be an increasingly important aspect as system complexities grow.

The evolution of FPGAs to SoC platforms has transformed the design problem from one only concerned with HDL-based implementation to a highly parallel, programmable logic fabric to a hardware/software system design challenge involving the incorporation of IP cores. Of course, the shift toward hardware/software FPGAs is not new, as in the early 2000s the Xilinx Virtex-II incorporated a PowerPC processor into the FPGA die, but this was not truly supported as a hardware/software environment. This has now meant an explosion in requirements for design teams with skills in embedded system programming, memory partitioning and incorporation of system components and accompanying system timing issues.

14.3 Big Data and the Shift toward Computing

An interesting development highlighted in Chapters 1 and 12 is the interest taken by major computing companies such as Microsoft, Intel and IBM in FPGAs. Whilst FPGAs have been around for many decades, it has only been in the last couple of years that these major companies have shown an interest in employing this technology. This has been

driven by energy concerns and this computing infrastructure driven by the emergence of data science and accompanying data centers to provide the infrastructure for undertaking such data inquiries.

The new computing infrastructures are challenging as they need to access large and varied sources of data distributed across websites and then create in-memory databases in a dynamic fashion to create microservers that will perform both transactional and analytical processing. The former is being undertaken on general- purpose computers using dedicated hardware to undertake the latter form of processing. One example is the Nanostreams project (www.nanostreams.eu) which is looking to create an application-specific heterogeneous analytics-on-chip (AoC) engine to perform such processing. It includes an AoC accelerator being developed by Analytics Engines Ltd. that is based on FPGAs in the form of a programmable, customizable processing core called Nanocore which acts to give improvements in performance and energy-efficiency over GPUs.

Microsoft has undertaken research into novel, FPGA-based data center architectures and created a reconfigurable fabric called Catapult. They argue that "datacenter providers are faced with a conundrum: they need continued improvements in performance and efficiency, but cannot obtain those improvements from general-purpose systems" (Putnam *et al.* 2014). The system comprises a bed of 1632 servers equipped with FPGAs giving gains in search throughput and latency for Bing, giving 95% greater ranking throughput in a production search infrastructure at comparable latency to a software-only solution (Putnam *et al.* 2014).

The IBM work with Xilinx has focused on accelerating Memcache2, a general-purpose distributed memory caching system used to speed up dynamic database-driven searches (Blott and Vissers 2014). Intel's purchase of Altera (Clark 2015) clearly indicates a clear aim of employing FPGAs in heterogeneous computing for data centers.

Unlike processors, FPGAs offer the capability of turning off and on resources, thus allowing scaling. This provides a more direct relationship between power consumed and the processing employed. Therefore, we can create an implementation where the dynamic power can be adjusted to match the processing granularity needed by the user. This can be further refined by adjusting the voltage and employing clock gating to reduce the overall static power consumed. As outlined in Chapter 13, this further improvement may be possible but at the cost of increased design effort.

14.4 Programming Flow for FPGAs

A topic closely associated with the further adoption of FPGAs in computing applications is the design flow. Whilst progress has been made in increasing the level of abstraction, thus removing the need to get software designers to learn specialized HDLs for programming FPGAs and allowing them to employ C descriptions in Vivado and even OpenCL, the compile times will still seem strangely long to programmers.

These tools now allow developers to write their programs in high-level languages (well, high-level for an engineer!) and then use aspects of the tools to gauge speed requirements against FPGA resources. However, compile times of typically hours will seem alien to programmers and will be prohibitive to further adoption of the technology. Considerable efforts are being made to overcome this limitation (including efforts by

this book's authors) by building multi-processors which can then be programmed in software.

This seems counterproductive for several reasons. Firstly, the FPGA vendors have already introduced the MicroBlaze (Xilinx) and NIOS (Altera) processors. Secondly, the advantage of the FPGA is in creating application- specific implementations, thus overcoming the fixed multi-processor architecture which is exactly what is being proposed here. So this is the conundrum: how to gain the advantages of FPGA performance without having to undertake much of the work highlighted in this book.

14.5 Support for Floating-Point Arithmetic

A conscious decision to first introduce scalable adder structures in early FPGAs and then dedicated multiplicative complexity in latter versions, such as the Stratix® III family from Altera and the Virtex™-5 FPGA family from Xilinx, has greatly influenced the use of FPGAs for DSP systems. Along with the availability of distributed memory, this has driven further interest in using FPGAs for computing due to the extremely high computation rates required.

If FPGAs are to be applied to computing or even supercomputing applications, then support for floating-point is needed. Chapter 5 outlined the progress that FPGA vendors have made in incorporating floating-point arithmetic in FPGAs, particularly in the Altera Arria® 10 family. This FPGA contains multiple IEEE 754 single-precision multipliers and IEEE 754 single-precision adders in each DSP block. This provides support for a variety of addition, multiplication and MAC floating-point operations, useful for a variety of vector operations.

The Arria® 10 FPGA family gives a peak performance of 3340 GMACs and 1366 GFLOPS. The technology has been incorporated into the new 510T from Nallatech which is termed an "extreme compute acceleration" technology targeted at data centers. It is an FPGA PCIe Gen3 card comprising two Altera Arria 10 1150 GX FPGAs providing up to 3 TFLOPS with 4 GB DDR3 per FPGA. The card offers hybrid memory cube memory architectures using a high-speed process technology through-silicon via bonded memory die. Altera offers a single Arria FPGA card called a DK-SOC-10AS066S-ES development kit. Other FPGA-based platform vendors include Alphadata, Accelize and Picocomputing (now Micron).

14.6 Memory Architectures

The support for parallel and pipelining operations was highlighted as the major attraction of FPGAs when considered for implementing DSP systems. However, one factor that has received some attention throughout this book is the availability of a wide range of different sizes of parallel memory, whether in the form of distributed RAM blocks, simple LUTs or a single register.

As highlighted by Wulf and McKee (1995), the memory wall gives a depressing view for fixed computer architectures as the ratio of the memory access time to the processor cycle time increases. Whilst some approaches try to address this via technology and

increased use of multi-level caches (Baer and Wang 1988), FPGAs get around the problem by naturally developing a highly parallel solution with a distributed memory architecture. This happens through deliberate derivation of a distributed memory architecture or as a result of an algorithmic optimization, as for example in the application of pipelining which, in effect, results in the creation of distributed memory. This approach is particularly suited to many signal and data systems due to data independence and high computation rates.

This means that there needs to be a focus on ensuring memory utilization rather than computation. This was seen in the Imagine processor (Kapasi *et al.* 2002) where the memory architecture was developed for the class of algorithms needed, and in some of the FPGA examples in Chapters 6, where different memory, i.e. LUTs in the forms of SRLs, was selected in preference to flip-flops to provide more efficient implementation delay chains, either because of lack of flip-flop resources or more relevant selection of resources. However, this has tended to be a good design decision or optimization using the routines available in design tools, rather than a conscious need to develop memory-orientated architectures. Work by Fischaber *et al.* (2010) has suggested how design of memory can be directed from the dataflow level.

Bibliography

Baer J-L, Wang W-H. 1988. On the inclusion properties for multi-level cache hierarchies. In *Proc. 15th Ann. Int. Symp. on Computer Architecture*, 73–80.

Blott M, Vissers K 2014 Dataflow architectures for 10Gbps line-rate key-value-stores. In *Proc. IEEE Hot Chips*, Palo Alto, CA.

Clark D 2015 Intel completes acquisition of Altera. *Wall Street J.*, December 28.

Fischaber S, Woods R, McAllister J 2010 SoC memory hierarchy derivation from dataflow graphs. *Journal of Signal Processing Systems*, 60(3), 345–361.

Kapasi UJ, Dally WJ, Rixner S, Owens JD, Khailany B 2002 The Imagine stream processor. In *Proc. IEEE Int. Conf on Computer Design: VLSI in Computers and Processors*, pp. 282–288.

Putnam A, Caulfield AM, Chung ES, Chiou D, Constantinides K, Demme J, Esmaeilzadeh H, Fowers J, Gopal GP, Gray J, Haselman M, Hauck S, Heil S, Hormati A, Kim J-Y, Lanka S, Larus J, Peterson E, Pope S, Smith A, Thong J, Xiao PY, Burger D 2014 A reconfigurable fabric for accelerating large-scale datacenter services. In *Proc. IEEE Int. Symp. on Computer Architecture*, pp. 13–24.

Wulf WA, McKee SA 1995 Hitting the memory wall: implications of the obvious. In *SIGARCH Comput. Archit. News*, 23(1), 20–24.