9

Complex DSP Core Design for FPGA

9.1 Introduction

It is feasible to incorporate many billions of gates on a single chip, permitting extremely complex functions to be built as a complete SoC. This offers advantages of lower power, greater reliability and reduced cost of manufacture and has enabled an expansion of FPGA capabilities with devices such as Altera's Stratix® 10 and Xilinx's UltraScale™ FPGA families. With their vast expanse of usable logic comes the problem of implementing increasingly complex systems on these devices.

This problem has been coined the "design productivity gap" (ITRS 1999) and has increasingly become of major concern within the electronics industry. Whilst Moore's law predicts that the number of available transistors will grow at a 58% annual growth rate, there will only be a 21% annual growth rate in design productivity. This highlights a divergence that will not be closed by incremental improvements in design productivity. Instead a complete shift in the methodology of designing and implementing multi-million-gate chips is needed that will allow designers to concentrate on higher levels of abstraction within the designs.

As the silicon density increases, the design complexity increases at a far greater rate since silicon systems are now composed of more facets of the full system design and may combine components from a range of technological disciplines. Working more at the system level, designers become more heavily involved with integrating the key components without the freedom to delve deeply into the design functionality. Existing design and verification methodologies have not progressed at the same pace, consequently adding to the widening gap between design productivity and silicon fabrication capacity.

Testing and verification have become a major aspect of electronic design. Verification of such complex systems has now become the bottleneck in system-level design as the difficulties scale exponentially with the chip complexity. Design teams may often spend as much as 90% of their development effort on block or system-level verification (Rowen 2002). Verification engineers now often outnumber design engineers. There are many design and test strategies being investigated to develop systems to accelerate chip testing and verification. With the increasing level of components on a single piece of silicon

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

there is an increasing risk involved in the verification of the device. Added to this is the increased difficulty in testing design components integrated from a third party. So much more is at stake, with both time and monetary implications. The industry consensus on the subject is well encapsulated by Rowen (2002): "Analysts widely view earlier and faster hardware and software validation as a critical risk-reducer for new product development projects."

This chapter will cover the evolution of reusable design processes, concentrating on FPGA-based IP core generation. Section 9.2 discusses design for reuse, and Section 9.3 goes on to to talk about reusable IP cores. Section 9.4 discusses the evolution of IP cores, and Section 9.5 goes on to talk about parameterizable IP cores. Section 9.6 describes IP core integration and Section 9.7 covers current FPGA-based IP cores. Section 9.8 presents watermarking. Concluding comments are made in Section 9.9.

9.2 Motivation for Design for Reuse

There is a need to develop design and verification methodologies that will accelerate the current design process so that the design productivity gap will be narrowed (Bricaud 2002). To enable such an achievement, a great effort is needed to research the mechanics of the design, testing and verification processes, an area that to date has so often has been neglected. *Design for reuse* is heralded to be one of the key drivers in enhancing productivity, particularly aiding system-level design.

In addition to exponentially increased transistor counts, the systems themselves have become increasingly complex due to the combination of complete systems on a single device, with component heterogeneity bringing with it a host of issues regarding logic design and, in particular, testing and verification. Involving full system design means that designers need to know how to combine all the different components building up to a full system-level design. The sheer complexity of the full system design impacts the design productivity and creates ever more demanding time-to-market deadlines. It is a multidimensional problem trying to balance productivity with design issues such as power management and manufacturability.

Design productivity can be enhanced by employing design-for-reuse strategies throughout the entire span of the project development from initial design through to functional testing and final verification. By increasing the level of abstraction, the design team can focus on pulling together the key components of the system-level design, using a hierarchical design approach.

The 2005 International Technology Roadmap for Semiconductors report covers the need for design for reuse in great depth (ITRS 2005). To increase overall productivity and keep pace with each technology generation, the amount of reuse within a system design must increase at the same rate, and the level of abstraction must rise. A summary of one of the tables is given in Table 9.1. Productivity gains by employing reuse strategies for high-level functional blocks are estimated to exceed 200% (ITRS 2005). These reusable components need to be pre-verified with their own independent test harness that can be incorporated into the higher-level test environment. This can be achieved by incorporating IP cores from legacy designs or third-party vendors. The need for such cores has driven the growth of an IP core market, with ever greater percentages of chip components coming from IP cores.

Table 9.1 SoC design productivity trends (normalized to 2005)

	2012	2014	2016	2018	2020
Design needed to be reused (%)	58	66	74	82	90
Trend SoC total logic size	5.5	8.5	13.8	20.6	34.2
Required productivity for new designs	4.6	6.7	10.2	14.3	22.1
Required productivity for reused designs	9.2	13.5	20.4	28.6	44.2

In 2006, the ITRS reported that the percentage of logic from reused blocks was at 33%, and this figure is expected to reach 90% by 2020. It is hard to determine if this ambitious target will be achieved, but in 2016 the Design & Reuse website (http://www.design-reuse.com/) boasted 16,000 IP cores from 450 vendors. Thus there seems to be an active community involved in producing IP cores but, of course, this does not translate into reuse activity.

The discussion to date regarding design for reuse has focused on ASIC design. However, some of the key concerns are becoming increasingly relevant with FPGA design. With the onset of microprocessors and other additional auxiliary components on an FPGA, the drive is now for system-level design on a single device. With this advance comes the need to drive design reuse methodologies for FPGA technologies and to close the design productivity gap.

9.3 Intellectual Property Cores

One of the most favorable solutions for enhancing productivity is the strategy of using pre-designed functional blocks known as silicon IP cores. The term "IP core" applies to a range of implementations ranging from hard cores, which are given in the form of circuit layout, through to soft cores, which can be in the form of efficient code targeted at programmable DSP or RISC processors, or dedicated cores captured in an HDL.

The flexibility inherent in DSP processor solutions has often been cited as a key reason for their widespread use within industry, despite the obvious reduction in overall performance criteria such as speed, area and power. At the other end of the spectrum application-specific hardware designs provide unrivaled performance capabilities at the cost of design flexibility.

Design-for-reuse methodologies provide the flexibility allowing designs targeted to one project to be regenerated for another; the key is how to develop the initial design so that high performance can be obtained meeting the changing needs of the project specifications. Within the realms of ASIC and FPGA implementations, IP cores are often partitioned into three categories: hard, firm and soft. Hard IP refers to designs represented as mask layouts, whereas firm IP refers to synthesized netlists for a particular technology. Soft IP refers to the HDL version of the core that will have scalability and parameterization built in. For the latter, the term that has evolved is parameterizable IP. They can be designed so that they may be synthesized in hardware for a range of specifications and processes. For DSP applications parameters such as filter tap size, DCT point size, and wordlength may be made flexible (Ding *et al.* 1999). Parameters controlling these features would be fed into the code during the synthesis, resulting in the

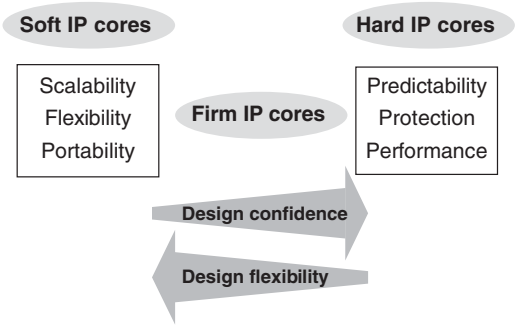

Figure 9.1 Benefits of IP types

desired hardware for the application. There are advantages and disadvantages with each type of IP core, as shown in Figure 9.1.

Some flexibility can still be included from the outset in firm and hard IP devices. In these cases, the IP parameters that define the core are termed static IP (Junchao *et al.* 2001), whereby registers internal to the final design can be set to allow a multiplexing of internal circuits so as to reconfigure the functionality of the design. Reconfiguration has been a subject of great interest for FPGAs, particularly with their increasing capabilities (see Alaraje and DeGroat 2005).

In contrast, the IP parameters within soft IP cores are termed dynamic IP parameters. They are often local or global parameters such as data widths, memory sizes and timing delays. Control circuitry may also be parameterized, allowing scalability of the design. Parameters may be set to allow the same primary code to optimize for different target technologies from ASIC libraries to different FPGA implementations.

Many companies offer IP products based around DSP solutions, that is, where the IP code is embedded onto DSP processors. This offers full flexibility, but with the obvious reduction in performance in terms of area, power and speed. Texas Instruments and particularly ARM^{TM} are two examples of successful companies supplying chipsets with supporting libraries of embedded components.

In a similar manner, there are now many companies delivering firm and soft IP cores. Several FPGA companies not only sell the chips on which the user's designs can be implemented, but can also provide many of the fundamental building blocks needed to create these designs. The availability of such varied libraries of functions and the blank canvas of the FPGA brings great power to even the smallest design team. They no longer have to rely on internal experts in certain areas, allowing them to concentrate on the overall design, with the confidence that the cores provided by the FPGA vendors have been tested through use by previous companies. The following list of current IP vendors (Davis 2006) shows the diversity of IP products:

CEVA: The CEVA families of silicon IP cores are fully programmable low-power architectures for signal processing and communications (http://www.ceva-dsp.com/).

Barco-Silex: IP cores in RTL HDL form or netlist for cryptography functions including AES, Data Encryption Standard (DES) and hashing, public key and video products including JPEG 2000, JPEG, MPEG-2 and VC-2 LD (http://www.barco-silex.com).

OpenCores: OpenCores is the world's largest site for development of hardware IP cores as open source (www.opencores.org).

Digital Blocks: VHDL and Verilog core for various network functionality including UDP, IPv4, IPv6 and Transmission Control Protocol (TCP) (http://www.digitalblocks.com/).

Within hard IP cores, components can be further defined (Chiang 2001), although arguably the definitions could be applied across all variations of IP cores, with the pre-silicon stage relating more to the soft IP core and the production stage relating to a pre-implemented fixed design hard IP core:

Pre-silicon: Given a one-star rating if design verified through simulation.
Foundry verified: Given a three-star rating if verified on a particular process.
Production: Given a five-star rating if the core is production proven.

When developing IP, vendors often offer low-cost deals so as to attract system designers to use their new product and prove its success. Once silicon proven, the product offers a market edge over competing products.

9.4 Evolution of IP cores

As technology advances, the complexity of the granularity of the cores blocks increases. This section gives a summary of the evolution of IP cores.

Within the realms of ASICs, families of libraries evolved bringing a high level of granularity to synthesis. At the lowest level the libraries define gated functions and registers. With increased granularity, qualified functional blocks were available within the libraries for functions such as UARTs, Ethernet and USBs. Meanwhile, within the domain of DSP processors, companies such as TI were successfully producing software solutions for implementation on their own devices.

The development of families of arithmetic functions is where the role of IP cores in design for reuse for ASIC and FPGA designs came to play. It was a natural progression from the basic building blocks that supported ASIC synthesis. The wealth of dedicated research into complex and efficient ways of performing some of the most fundamental arithmetic operations lent itself to the design of highly sophisticated IP cores operating with appealing performance criteria.

Figure 9.2 illustrates the evolution of IP cores and how they have increased in complexity, with lower-level blocks forming key components for the higher levels of abstraction. The arithmetic components block shows a number of key mathematical operations, such as addition, multiplication and division, solved and implemented using the techniques described in Chapter 3. The list is far from conclusive.

With greater chip complexity on the horizon, arithmetic components became the building blocks for the next level in the complexity hierarchy, for designs such as filter banks consisting of a large array of multiply and accumulate blocks. This led to the development of fundamental DSP functions such as FFT and DCT. These examples are matrix-based operations consisting of a large number of repetitive calculations that are performed poorly in software. They may be built up from a number of key building blocks based on multiply and accumulate operations.

The structured nature of the algorithms lends itself to scalability, allowing a number of parameters to control the resulting architecture for the design. Obvious examples

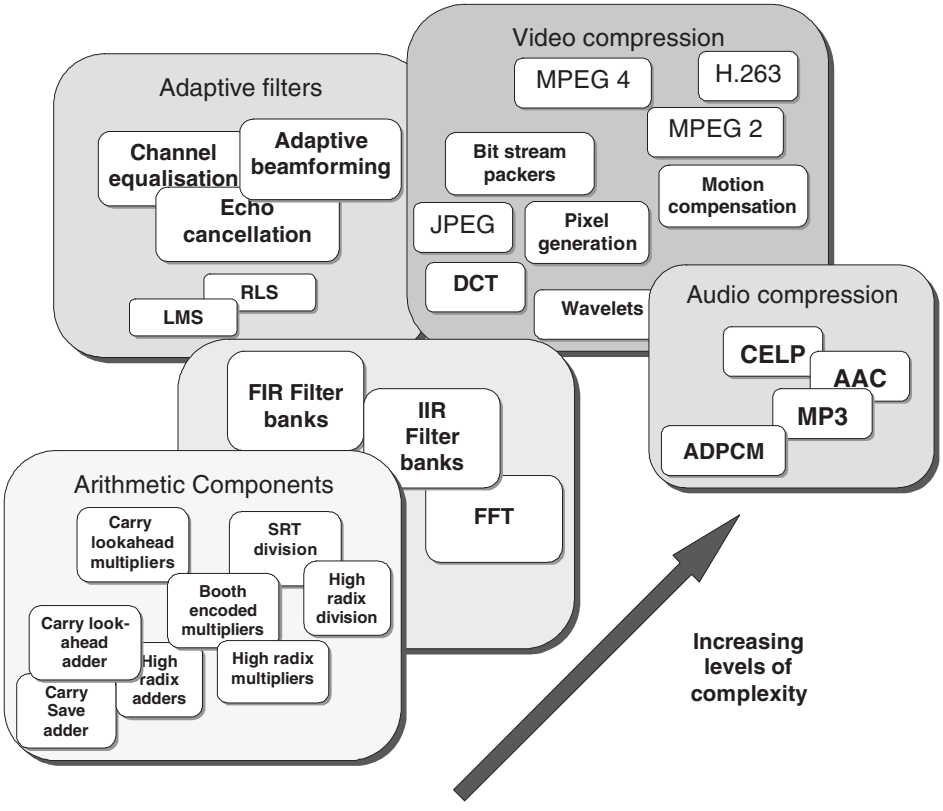

Figure 9.2 Evolution of IP cores

of parameters are wordlength and truncation. Other examples would be based on the dimensions of the matrix operations, relating, for example, to the number of taps on a filter. The work devoted to a single application could be expanded to meet the needs of a range of applications.

Other more complicated foundation blocks were developed from the basic arithmetic functions. More complicated filter-based examples followed such as adaptive filters implemented by the rudimentary LMS algorithm or the more extravagant QR-RLS algorithm (see Chapter 11). Highly mathematical operations lend themselves well to IP core design. Other examples, such as FEC chains and encryption, whereby there is a highly convoluted manipulation of values have also been immensely successful.

IP cores have now matured to the level of full functions that might previously have been implemented on independent devices. Again there is an increased level of complexity. Within image processing, the DCT is a key algorithm for JPEG and MPEG functions. Each of these will be covered in more detail below.

9.4.1 Arithmetic Libraries

Figure 9.2 lists a number of basic mathematical operations, namely addition, multiplication, division and square root. The efficient hardware implementation of even the most

basic of these, addition, has driven an area of research, breaking down the operations to their lowest bit level of abstraction and cleverly manipulating these operations to enhance the overall performance in terms of area, clock speed and output latency (Koren 1993). This subsection gives some detail on the choice of arithmetic components and how parameters could be included within the code.

Fixed-Point and Floating-Point Arithmetic

The arithmetic operations may be performed using fixed-point or floating-point arithmetic. With fixed-point arithmetic, the bit width is divided into a fixed-width magnitude component and a fixed-width fractional component. Due to the fixed bit widths, overflow and underflow detection are vital to ensuring that the resulting values are accurate. Truncation or rounding would be needed to protect against such problems.

With floating-point arithmetic, the numbers are stored in a sign–magnitude format. The most significant bit represents the sign. The next component represents an exponential value. Biasing is used to enable the exponent to represent very small and very large number. The remaining data width is the mantissa, which represents the fractional component of the number and is given the boundaries of greater than or equal to 1 but less than 2. The greater flexibility of floating-point enables a wider range of achievable values.

Although number representation within the data width differs for fixed-point and floating-point design, there is overlap in how the main functionality of the operation is performed, as illustrated for multiplication in Figure 9.3; there has been research into automating the conversion from fixed-point to floating-point.

Addition, Multiplication, Division and Square Root

There has been an extensive body of work devoted to high-performance implementations of arithmetic components as indicated in Chapter 3. At was clear from the description given in Chapter 5, dedicated hardware functionality has been included in many

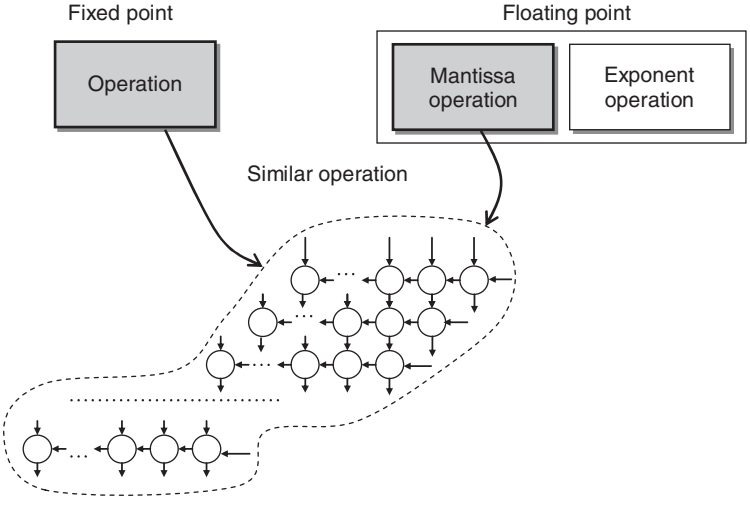

Figure 9.3 Fixed- and floating-point operations

FPGA families to support fixed-point addition/subtraction and multiplication and is supported in high- level synthesis tools. Division and square root are more complex techniques, and procedures for their implementation were described in Chapter 3. Most FPGA vendors supply IP libraries to support such functions.

9.4.2 Complex DSP Functions

More complex DSP cores can be created from lower-level arithmetic modules as illustrated in Figure 9.2, leading to the creation of systems for audio and video. For example, FFT cores can be used to create OFDMA systems and a power modulation/demodulation scheme for communications applications such as wireless (802.11a/g) or broadcasting (DVB-T/H). DCT and wavelet cores are used for a wide range of image processing cores, and LMS and RLS filtering cores applied to a range of adaptive beamformers and echo cancelation systems.

9.4.3 Future of IP Cores

As the level of abstraction within the core building blocks in designs increases, the role of the designer moves toward that of a system integrator, particularly with development using current FPGA devices enabling full system functionality on a single device. For the growth in IP core usage to continue, other aspects of the design flow will need to be addressed. This has driven developments in higher-level languages along with associated synthesis tools.

9.5 Parameterizable (Soft) IP Cores

This section covers the development of parameterizable IP cores for DSP functions. The starting point for the hardware design of a mathematical component may be the SFG representation of the algorithm. Here, a graphical depiction of the algorithm shows the components required within the design and their interdependence. The representation could be at different levels, from the bit-level arithmetic operations through to the cell-level functions.

Figure 9.4 shows the conventional design flow for a DSP-based circuit design, starting from the SFG representation of the algorithm. If a certain aspect of the specification were to be changed, such as wordlength, then the traditional full design flow would need to be repeated. The development of the IP core where the HDL is parameterized allows this flow to be dramatically altered, as shown in Figure 9.5.

The IP core design process needs to encompass the initial studies on data performance on the effects of wordlength and truncation, etc. Effort is needed to ensure that operation scheduling would still be accurate if additional, pipeline stages are included. The aim is for the parameterization of the core to lead seamlessly to a library of accurate cores targeted to a range of specifications, without the need to alter the internal workings of the code.

The system should effectively allow a number of parameters to be fed into the top level of the code. These would then be passed down through the different levels of abstraction of the code to the lowest levels. Obviously, considerable effort is needed at the architecture level to develop this parameterizable circuit architecture. This initial expense in

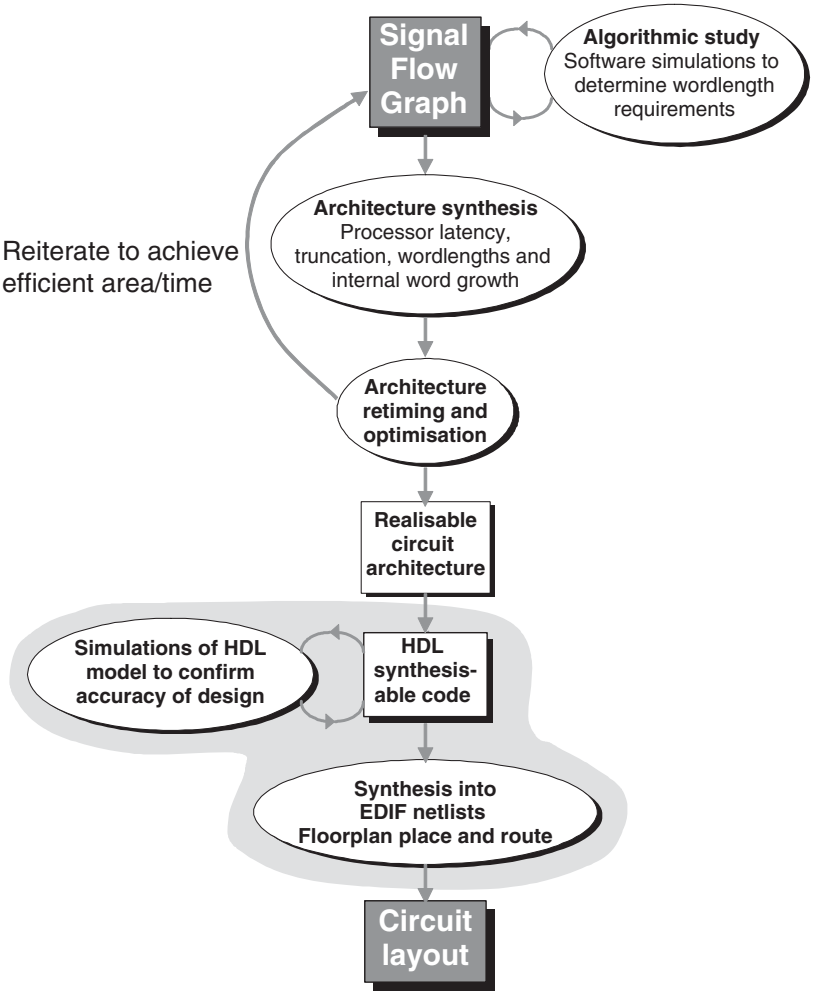

Figure 9.4 Circuit design flow

terms of time and effort undoubtedly hinders the expanded use of design-for-reuse principles. However, with this initial outlay great savings in company resources of time and money may be obtained. The choice of design components on which to base further designs and develop as IP is vitally important for this success. The initial expenditure must, in the long run, result in a saving of resources.

Future design engineers need to be taught how to encompass a full design-for-reuse methodology from the project outset to its close. The design process needs to consider issues such as wordlength effects, hardware mapping, latency and other timing issues before the HDL model of the circuit can be generated. The aspects that need to be considered create a whole new dimension to the design process, and designers need to keep in mind reusability of whatever they produce whether for development or test purposes.

If a design is developed in a parameterized fashion then initial analysis stages can be eliminated from the design flow, as illustrated in Figure 9.5, allowing additional circuits

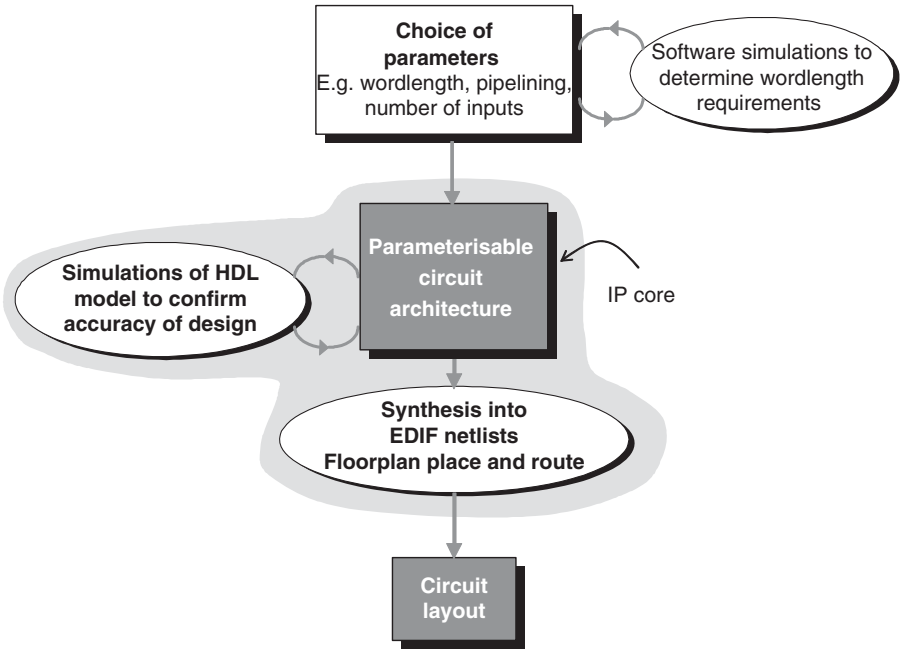

Figure 9.5 Rapid design flow

to be developed and floorplanned extremely quickly, typically in days as opposed to months. This activity represents a clear market for IP core developers (Howes 1998) as it can considerably accelerate the design flow for their customers. However, it requires a different design approach on behalf of the IP core companies to develop designs that are parameterizable and will deliver a quality solution across a range of applications.

9.5.1 Identifying Design Components Suitable for Development as IP

Within a company structure, it is vital that the roadmap is considered within the development of IP libraries as there is a greater initial overhead when introducing design-for-reuse concepts. Greater success can be achieved by taking an objective look at possible future applications so that a pipeline of developments can evolve from the initial ground work. If design for reuse is incorporated from the outset then there can be immense benefits in the development of a library of functions from the initial design.

It is often possible to develop a family of products from the same seed design by including parameterization in terms of wordlength and level of pipelining, and by allowing scalability of memory resources and inputs.

Larger designs may need to be broken down into manageable sections that will form the reusable components. This is particularly true for large design such as MPEG video compression whereby a range of different applications would require slightly different implementations and capabilities. By picking out the key components that remain unchanged throughout the different MPEG profiles and using these as the key hardware accelerators for all of the designs, vast improvements in time to market can be made. Furthermore, existing blocks from previous implementations have the advantage

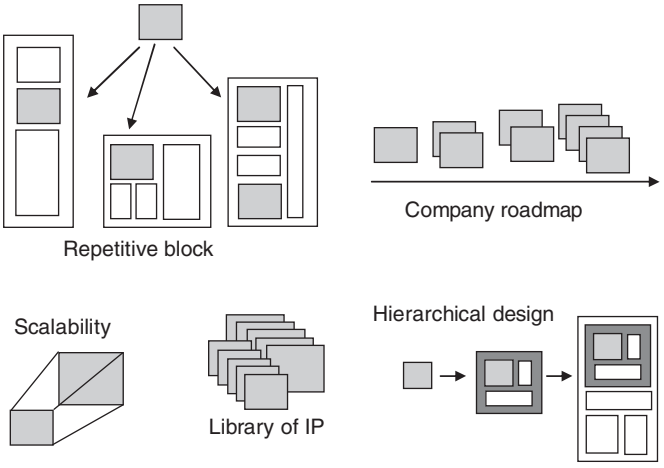

Figure 9.6 Components suitable for IP

of having been fully tested, particularly if they have gone to fabrication or deployment on FPGA. Reusing such blocks adds confidence to the overall design. Existing IP cores may also form the key building blocks for higher-level IP design, creating a hierarchical design. These key points are illustrated in Figure 9.6.

9.5.2 Identifying Parameters for IP Cores

Identifying the key parameters when developing an IP core requires a detailed understanding of the range of implementations in which the core may be used. It is important to isolate what variables exist within possible specifications. The aim is to create as much flexibility in the design as possible, but only to the extent that the additional work will be of benefit in the long run. Overparameterization of a design affects not only the development but also the verification and testing time needed to ensure that all permutations of the core have been considered. In other words, consider the impact on the design time and design performance by adding an additional variable and weigh this up with thoughts on how the added flexibility will broaden the scope of the IP core.

Figure 9.7 lists some of example parameters: modules/architecture, wordlength, memory, pipelining, control circuitry, and test environment. Aspects such as wordlength or truncation can be parameterized. Other features can be used to allow full scalability, such as scaling the number of taps in an FIR filter. The diagram highlights the flexibility of allowing different modules depending on the application, or enabling the level of pipelining to be varied. Scalable parameters such as wordlength and level of pipelining affect the timing and the operations of the IP core and therefore need to be accounted for within initial development, so that the code can rapidly be re-synthesized for a new architecture. This a key factor for the success of an IP core.

It is crucial that the resulting core has performance figures (in terms of area, power and speed) comparable to a handcrafted design. As usual, the process comes down to a balance between time and money resources and the performance criteria of the core.

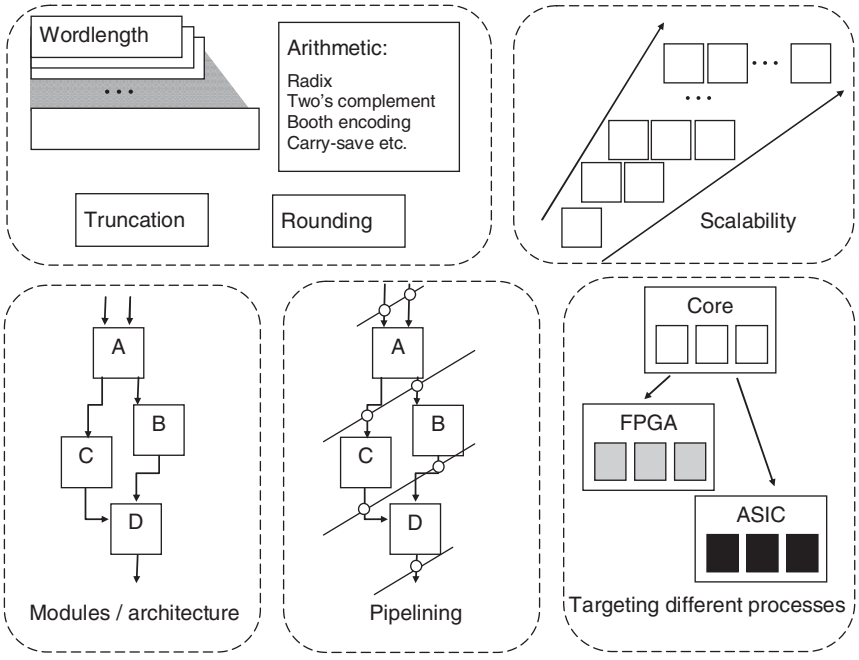

Figure 9.7 IP parameters

With time to market being a critical aspect to a product's success, the continuing use of IP components within the electronics industry has major benefits.

With further parameterization, cores can be developed to support a range of different technologies enabling the same design to be re-targeted from ASIC to FPGA, as highlighted in Figure 9.7. Allowing the same code to be altered between the two technologies has the obvious advantage of code reuse; however, it also allows for a verification framework whereby cores are prototyped on FPGA and then the same code is re-targeted to ASIC. There is obviously no guarantee that the code conversion from FPGA to ASIC implementations will not in itself incur errors. However, the ability to verify the code on a real-time FPGA platform brings great confidence to the design process and enables even the functional design to be enhanced to better meet the needs of the specification.

Consideration must be given to the level of parameterization as it makes the design more flexible and widens the market potential for the IP core. Gajski *et al.* (2000) highlight the issue of overparameterization, as increasing the number of variables complicates the task of verifying the full functionality of each permutation of the design. There is also the aspect that designs have been made so generic that they may not match the performance requirements for a specific application. Gajski *et al.* argue that increasing the number of parameters decreases the quality and characterizability of the design, that is to say, how well the design meets the needs of the user. There are also the added complications with verification and testing. These points are highlighted in Figure 9.8.

An obvious parameter is wordlength, which ultimately represents the trade-off between SNR and performance criteria such as area and critical path. Figure 9.9 gives an illustration of such analysis by plotting SNR against a range of wordlengths. It can

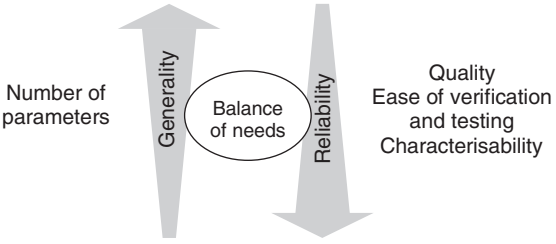

Figure 9.8 Effect of generalization on design reliability

be seen that increasing the wordlength does not significantly improve the overall performance. For addition an increase of one bit will linearly scale the area of the resulting design, whereas it has an exponential effect for multiplication and division. As with area, additional bits will affect critical path, possibly resulting in the need to introduce further pipeline stages.

Existing designs may have relied on carefully crafted libraries of arithmetic functions that were scalable in terms of bit width and level of pipelining, providing optimum performance in terms of area and speed. However, the impact of introducing processing blocks has a granular impact on area and performance when adjusting wordlengths. Obviously, there will be a need to add parameters to allow the wordlengths to be varied from the module boundary without having to manually edit the code.

Memory will also need to be scalable to account for the different wordlengths, but also for variations in the number of inputs or stored values. In addition, flexibility will need to be included within the code to allow different types of memory blocks to be employed in accordance with the choice of target technology.

In Verilog, one of two solutions can be used. Either instantiations of BRAMs for the target device can be scripted with DEFINEs at the top level of the code pointing to the memory of choice. Alternatively, the code can be written in such a way as to "imply" the application of a memory, which will be picked up during synthesis and will instantiate the memories accordingly. However, slight improvements may be still be obtained if the memory instantiations are hand-crafted but this will result in more complex code.

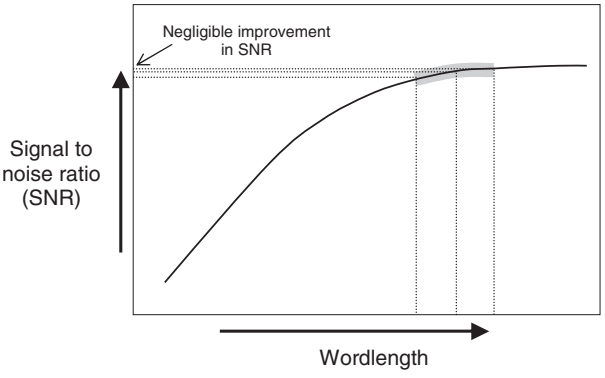

Figure 9.9 Wordlength analysis

Allowing flexibility in the data rate and associated clock rate performance for an application requires the ability to vary the level of pipelining to meet critical path requirements. This may tie in to changes in the wordlength and often a variation in the number if pipeline stages are to be part of the reusable arithmetic cores. Obviously, an increase or decrease in the number of pipeline cuts in one module will have a knock-on effect on timing for the associated modules and the higher level of hierarchy as discussed in Chapters 8 and 10.

Thus, control over the pipeline cuts within the lower-level components must be accessible from the higher level of the module design, so that lower-level code will not need to be edited manually. Control circuitry is coupled strongly with the level of pipelining and therefore must include some level of scalability to ensure that the design is totally parameterizable.

An alternative method of developing parameterizable cores can be used to develop a software code to automate the scripting of the HDL version of the module. This is particularly useful with Verilog as it does not have the same flexibility in producing scalable designs as VHDL does.

Parameterized Design and Test Environment

All associated code accompanying the IP core should be designed with scalability in mind. Bit-accurate C-models used for functional verification should have the capability to vary bit widths to match the IP core. For cycle accurate testing, the timing must also be considered. Testbenches and test data derivation are also required to be parameterizable, allowing for a fully automation generation of an IP core and it associated test hardness. The use of software such as a C-model to generate the test hardness and test data files may be advantageous in the development of the IP core. This is illustrated in Figure 9.10.

9.5.3 Development of Parameterizable Features

Many of the IP designs applied for ASIC design can be expanded for FPGA implementations. Each family of devices has its own memory components and methods for instantiating the built-in modules. The principle would be to design the code so as to allow the core to be re-targeted at the top level to the family FPGA devices of choice. This is particularly important as FPGAs are rapidly progressing, thus legacy code needs to accommodate additions for future devices and packages.

Arithmetic Block Instantiation

One example of the variations between FPGA devices is memory blocks. Each family has its own architecture for these blocks as outlined in Chapter 5. They can either be instantiated directly, or the memories can be inferred by synthesis tools. The latter allows the synthesis tool to pick up the memory blocks directly from the library and map the register values to this memory or even to ROM blocks. This has obvious benefits in that the code does not become FPGA family-specific.

There may still be a benefit in manually instantiating the memory blocks as a slight improvement in usage of the blocks can sometimes be achieved. However, the code is specified for the target device.

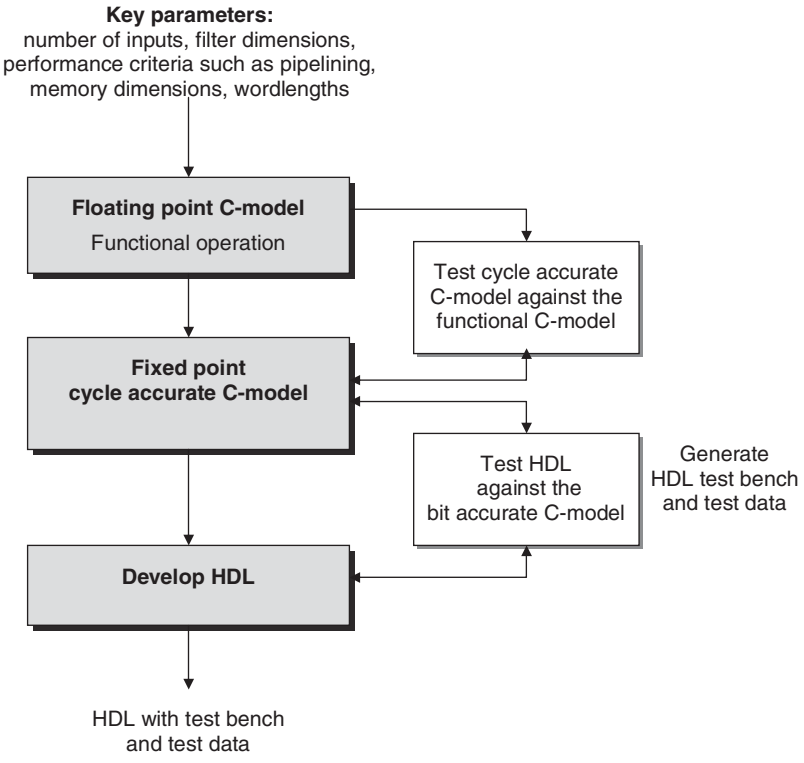

Figure 9.10 Design flow

Arithmetic Block Instantiation

Different target FPGAs may have variants of arithmetic operations available for the user. Typically the FPGA will contain a number of high-performance DSP blocks for instantiation. If the code is to be employed over a range of FPGA families and even between FPGA and ASIC, then there needs to be a facility to define the operator choice at the top level. Within Verilog, this would be done through the use of DEFINEs held in a top-level file, allowing the user to tailor the design to their current requirements.

9.5.4 Parameterizable Control Circuitry

For complex modules, there may be a need to allow for scalable control circuitry, i.e. a framework that will allow for the changes in parameters, such as the knock-on effect from additional pipelining delays. Any increase in the number of inputs or wordlength may have an effect on the scheduling and timing of the module. It may be possible to develop the control circuitry to cope with these variations.

9.5.5 Application to Simple FIR Filter

This section concludes with an example of parametric design applied to a simple FIR filter. The key parameters for the design will be highlighted and suggestions made

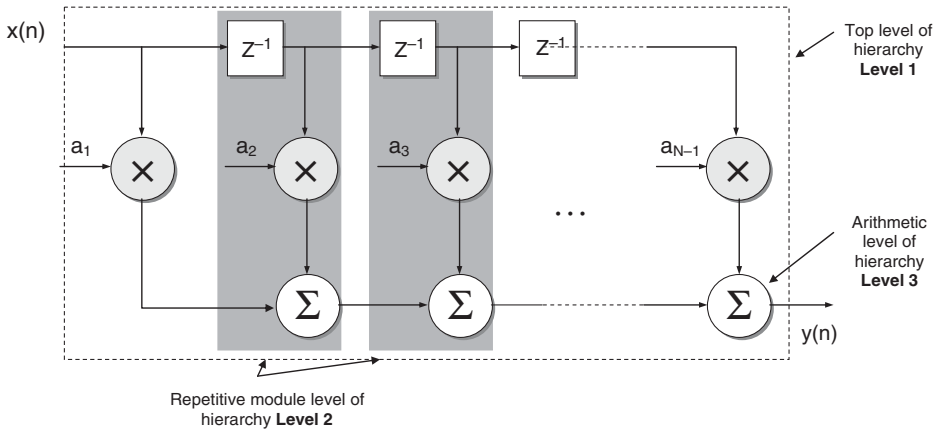

Figure 9.11 FIR filter hierarchy example

concerning how they can be incorporated into the code. Recall the difference equation for an FIR filter (equation (4.2)).

In this example, the FIR filter has three levels of hierarchy as depicted in Figure 9.11:

- Level 1: This is the top level of the filter structure with input $x()$ and output $y(n)$.
- Level 2: The top level of the FIR filter can be composed of a single DSP processing block to compute $a_0x(n)$ followed by an addition.
- Level 3: This is the arithmetic operation level, consisting of a single multiply, add and delay modules.

Another dimension of the design may be the folding of the FIR operations onto a reduced architecture as described in Chapter 8 where the hardware modules (shown as level 2 in Figure 9.11) are reused for different operations within the filter. Of course, multiplexers and programmable register blocks need to be added, and in this example all the MAC operations are performed on one set of multiplier and adder modules. The multiplexers are used to control the flow of data from the output of the MAC operations and back into the arithmetic blocks. The choice of level of hardware reduction will depend on the performance requirements for the application.

9.6 IP Core Integration

One of the key challenges of successful design reuse is with the integration of the IP cores within a user's system design. This can often be a stumbling block within a development. Investigations have been carried out to highlight these issues (Gajski *et al.* 2000), and guidelines have been set out to try to standardize this process (Birnbaum 2001; Coussy *et al.* 2001).

For the successful integration of an IP core into a current design project, certain design strategies must be employed to make the process as smooth as possible. This section highlights some of the pitfalls that might be met and provides some guidance when dealing with IP core sources externally to the design team, whether within or outside the company.

One of the considerations that may need to be addressed is whether to outsource IP components externally, or even source the IP from within a company or organization but from different departments. Successful intra-company use of IP requires adequate libraries and code management structures. Incorporating IP components from other design teams can often be the main barrier slowing down the employment of design-for-reuse strategies in system-level design.

9.6.1 Design Issues

Greater success can be obtained by taking an objective look at possible future applications so that a pipeline of developments can evolve from the initial ground work. If design for reuse is incorporated from the outset then there can be immense benefits in the development of a library of functions from the initial design.

- Need to determine the parts of the design that will be useful in future developments.
- What are possible future applications?
- Study the roadmap for the product.
- Is there a possibility of development of a family of products from the same seed design?
- How can a larger design be partitioned into manageable re-usable sections?
- Find existing level of granularity, i.e. is there any previous IP available that could provide a starting level for development?

Outsourcing IP

One of the limiting factors of using outsourced IP is the lack of confidence in the IP. The IP can be thought of as having different grades, with one star relating to a core verified by simulation, and three stars relating to a core that has been verified through simulation on the technology (i.e. a gate-level simulation). A five-star IP core provides the most confidence as it has been verified through implementation (Chiang 2001).

FPGA vendors have collaborated with the IP design houses to provide a library of functions for implementation on their devices. The backing of the FPGA vendors brings a level of confidence to the user. The aspect of core reliability is not as crucial for FPGA as it is for ASIC. However, it is still important. Time wasted on issues of IP integration into the user's product may be critical to the success of the project.

Certain questions could be answered to help determine the reliability of an IP vendor:

- Has the core been in previous implementations for other users?
- Do the company supply user guide and data book documentation?
- Does the core come supplied with its own testbench and some test data?
- Will the company supply support with the integration, and will this incur an added cost?

In-house IP

For a small company within the same location, it would be a much easier task to share and distribute internal IP. However, this task is logistically difficult for larger companies spanning a number of locations, some of which may be affected by time zones as well as physical distance.

It would be wise for the company to introduce a structure for IP core design and give guidelines on the top-level design format. Stipulating a standard format for the IP cores

could be worthwhile and create greater ease of integration. Forming a central repository for the cores once they have been verified to an acceptable level would be necessary to enable the company's full access to the IP. Most companies already employ some method of code management to protect their products.

Interface Standardization

IP integration is a key issue as it ensures IP reuse and core configurability. IP cores are becoming more complex and configurable and can have numerous ports and hundreds of different configurations. Thus the provision of standardized interfaces is vital and the IP-XACT standard provides a mechanism for standardizing IP interfaces. It is now an IEEE standard (IEEE 2014) and is a mechanism to express and exchange information about design IP and its required configuration.

IP-XACT was developed by the Spirit consortium to enable sharing of standard component descriptions from multiple component vendors (Murray and Rance 2015). It defines an XML schema that is very easy to process and has the ability to make IP more "integration-ready" through interface standardization. It is argued that it can result in a 30% improvement in the time and cost of SoC integration.

It allows the creation of an interface on the component that contains a well- known set of ports called a bus interface; it can generally have high- level transactional or dataflow characteristics and behave as master or slave and also have different variants like direction and size. Once defined, these bus definitions can be used in conjunction with IP-XACT component descriptions to describe hardware interfaces which define the physical ports and allow mapping of these ports to the standardized definition.

Once this mapping has been defined, it is a case of checking that all of the required ports in a bus definition have been mapped, all directions are correct, all port widths are consistent with the bus definition, and there is no illegal mapping of ports.

9.7 Current FPGA-based IP cores

There are a number of cores available, both open source and commercial offerings. The open source cores tend to be available from the OpenCores website (opencores.org) and commercial offerings are available from a range of sites (see www.design-reuse.com/). The FPGA vendors also have their own IP repositories.

- Xilinx through LogiCore and their partner offer IP cores for DSP and math, embedded communications, memory interfaces and controllers and video and imaging.
- Altera's MegaCore® outlines cores for some FIR filters, FFTs, DRAM and SRAM controllers and their third-party providers offer a wide range of cores for communications, interfaces, DSP and video processing.
- Microsemi CompanionCore's portfolio offers a comprehensive collection of data security, image and vision processing, communications and processors, bus interfaces and memory controller cores.
- Lattice Semiconductor's IP portfolio comprises cores for DSP, Ethernet, PCI Express and video processing and display.

The FPGA companies aim to develop IP core technology to ensure a better relationship with their customer base and may look to provide this IP to ensure FPGA sales.

Thus in many cases, the IP requirements tend to be driven by the customer base. Also, FPGA companies will sometimes provide IP through their university programmes to stimulate technical interest in this areas.

9.8 Watermarking IP

Protecting FPGA IP has become a key research area both in terms of protecting the investment that the IP vendors have made but also as a reassurance to their customers that their investment was indeed worthwhile and has not been tampered with. For these reasons, an increasing amount of attention has been paid to developing techniques for ensuring this protection (Teich and Ziener 2011).

One solution is to hide a unique signature in the core, essentially termed *watermarking*, although there are also techniques for validating the core with no additional signature (Teich and Ziener 2011). Identification methods are based on the extraction of unique characteristics of the IP core, e.g. LUT contents for FPGA IP cores allowing the core author to be identified.

The concept of digital watermarking FPGA was first proposed by Lach *et al.* (1998). The owner's digital signature is embedded into an unused LUT located in a constrained area of unused slices in the FPGA at the place and route level of the implementation. This area is then obfuscated in the design using unused interconnect and "don't care" inputs of neighboring LUTs. The approach uses additional area and may impact timing and be vulnerable to attacks that look to remove the signature.

An alternative approach in Jain *et al.* (2003) embeds the watermark at the place and route stage by modifying the non-critical path delay between non-synchronous registers. It does not need additional hardware resources but can impact the path delay, and thus the performance of the design. The DesignTag is a novel, patented, security tag by Kean *et al.* (2008) which is used to verify the authenticity of a semiconductor device. It comprises a small, digital circuit which communicates through the package with an external sensor.

9.9 Summary

This chapter began by highlighting the need for design for reuse to address the challenges of building increasingly complex SoC devices. The increasing levels of silicon technology have stressed the need to reuse good designs from previous projects.

Design for reuse has been achieved by the creation of IP cores either in the form of pre-designed functional layout such as the ARM cores which present the user with a hardware platform on which they can develop software to implement the required functionality, or parameterized HDL code which can produce highly efficient code for programmable logic implementation. The aim of the HDL code is to capture the good design practice and procedures in such a way that HDL code is provided with a series of parameters which can be set and produce efficient implementation across a range of performance needs.

The process requires the creation of a base design from which a range of implementations can be derived where the area and speed will scale with change in parameters, otherwise it is frustrating for the designer to optimize the parameters for the best design. This process is demonstrated in detail in Chapter 11 for a QR-based RLS filter.

Bibliography

Alaraje N, DeGroat JE 2005 Evolution of reconfigurable architectures to SoFPGA. In *Proc. 48th Midwest Symp. on Circuits and Systems*, 1, pp. 818–821.

Birnbaum M 2001 VSIA quality metrics for IP and SoC. In *Proc. Int. Symp. on Quality Electronic Design*, pp. 279–283.

Bricaud P 2002 *Reuse Methodology Manual for System-On-A-Chip Designs*. Springer, New York.

Chiang S 2001 Foundries and the dawn of an open IP era. *Computer*, 34(4), 43–46.

Coussy P, Casseau E, Bomel P, Baganne A, Martin E. 2006. A formal method for hardware IP design and integration under I/O and timing constraints. *ACM Trans. Embedded Computing Systems*, 5(1), 29–53.

Davis L 2006 Hardware components, semiconductor-digital-programmable logic IP cores. Available from www.interfacebus.com/IPCoreVendors.html (accessed May 11, 2016).

Ding TJ, McCanny JV, Hu Y 1999 Rapid design of application specific FFT cores. *IEEE Trans. Signal Processing*, 47(5), 1371–1381.

Gajski DD, Wu ACH, Chaiyakul V, Mori S, Nukiyama T, Bricaud P 2000 Essential issues for IP reuse. In *Proc. Design Automation Conf.*, pp. 37–42.

Howes J 1998 IP new year. *New Electronics*, 31(1), 41–42.

IEEE 2014 IEEE Standard for IP-XACT, Standard Structure for Packaging, Integrating, and Reusing IP within Tool Flows. Available from http://standards.ieee.org/ (accessed May 11, 2016).

ITRS 1999 International Technology Roadmap for Semiconductors, Semiconductor Industry Association. http://public.itrs.net (accessed May 11, 2016).

ITRS 2005 International Technology Roadmap for Semiconductors: Design. available from http://www.itrs.net/Links/2005ITRS/Design2005.pdf (accessed May 11, 2016).

Jain AK, Yuan L, Pari PR, Qu G 2003 Zero overhead watermarking technique for FPGA designs. In *Proc. 13th ACM Great Lakes Symp. on Very Large Scale Integration*, pp. 147–152.

Junchao Z, Weiliang C, Shaojun W 2001 Parameterized IP core design. In *Proc. 4th Int. Conf. on Application Specific Integrated Circuits*, pp. 744–747.

Kean T, McLaren D, Marsh C 2008 Verifying the authenticity of chip designs with the DesignTag system. In *Proc. IEEE Int. Workshop on Hardware-Oriented Security and Trust*, pp. 59–64.

Koren I 1993 *Computer Arithmetic Algorithms*. Prentice Hall, Englewood Cliffs, NJ.

Lach J, Mangione-Smith WH, Potkonjak M 1998 Signature hiding techniques for FPGA intellectual property protection. In *Proc. IEEE/ACM Int. Conf. on Computer-Aided Design*, pp. 186–189.

Murray D and Rance S 2015 Leveraging IP-XACT standardized IP interfaces for rapid IP integration. ARM White paper. Available from https://www.arm.com/ (accessed May 11, 2016).

Rowen C 2002 Reducing SoC simulation and development time. *Computer*. 35(12), 29–34.

Teich J, Ziener D 2011 Verifying the authorship of embedded IP cores: Watermarking and core identification techniques. Plenary talk. *Int. Conf. on Engineering of Reconfigurable Systems and Algorithms*. Available at http://ersaconf.org/ersa11/program/teich.php