

# System Level Modelling of Reconfigurable FFT Architecture for System-on-Chip Design \*

Ali Ahmadinia, Balal Ahmad, Tughrul Arslan  
The University of Edinburgh  
School of Engineering and Electronics  
Mayfield Road, EH9 3JL Edinburgh, Scotland  
{A.Ahmadinia, B.Ahmad, T.Arslan}@ed.ac.uk

## Abstract

*In the system-on-chip (SoC) era, the growing number of functionalities included on a single chip requires the development of new design methodologies to keep the design complexity under control. Intellectual property reuse has been commonly employed as a technique to address this problem, but a new system-level approach is needed to integrate IP-Reuse methodology in the design flow, in order to speed up the designer's productivity. This paper aims to produce new high level IP models in SystemC for functional verification of IP integrations, incorporating both embedded custom reconfigurable and conventional IPs, which are optimised in terms of IP Core parameters. As a case study, a novel reconfigurable FFT architecture is presented and modelled in SystemC. Power, area and performance figures are presented as well.*

## 1 Introduction

In SoC design, IP-Core based design methods are now the mainstream to accomplish an SoC for complex applications. These IP-Cores include microprocessors, memory controllers, DSPs, codecs, bus interfaces, and numerous other peripheral components.

We aim to develop an environment supporting automatic generation of new SoC architectures targeting high performance, low power applications. Therefore, we use a cycle accurate SystemC based system-level simulation for efficient functional verification of each SoC platform configuration. In this paper, we present high level reconfigurable IP models in SystemC for system-level verification.

Applications, especially those representing embedded

\*The authors would like to acknowledge the support of the Engineering and Physical Sciences Research Council (EPSRC) of the U.K. under the grant no. EP/C528328/1.

systems, differ greatly in terms of their power, performance and size requirements. Though using the same basic architecture, applications may seek to minimise power, maximise performance, minimise size, or optimise some weighted combination of those constraints. Therefore, in order to maximise the number of applications that can use a predesigned architecture, the architecture must be highly configurable.

Such configurability for a range of power, performance and size constraints requires a new focus on parameterised system design, different from the past focus of building an architecture optimised for one particular set of constraints. The past focus was often transistor-centric, focusing on minimising transistors under given power and performance constraints, or maximising performance and minimising power under given size constraints. This paper aims to overcome challenges in high level modelling of reconfigurable features of embedded IPs for fast reconfigurable IP integration in SoC design.

This paper begins with related work in Section 2, high level IP modelling and design flow of IP based SoC design are discussed in Sections 3 and 4. Section 5 explains briefly a typical basic platform for SoC design, which is followed by the description of FFT architecture as a case study in Section 6. Section 7 presents SystemC implementation of the FFT and finally in Section 8 implementation results are analysed.

## 2 Related Work

Although IP reuse is an inevitable trend, there are still great challenges ahead. It is hard to guarantee that the whole system will work after integrating all pre-verified IPs. In Savage et al. [12], the authors introduce a pace setting IP development methodology and tool flow. They show that a strict quality based design methodology is the cornerstone to IP reuse. In Coussy et al. [4], the authors present a

methodology of IP integration in an SoC design that exploits both IP designer and SoC integrator constraints. First, they extract and specify IP functional and timing constraints from the IP core. Then, they apply their modelling style of integration constraints based on IP delay model.

In Han et al. [7], reusable IP design methodology has been classified into module level and system level. Interface based IP design methodology builds standalone IP verification environment from the module level, while platform based system design methodology develops SoC based on an integration platform. A pre-verified IP should be easily modifiable for another application. Designers have to analyse the performance requirement before customisation or parameterisation. Usually, a valid IP block needs wrapper for adapting to different bus protocol [4]. The wrapper overhead could be very significant and must be taken into account.

Some EDA companies provide a set of tools that allows incorporating IP cores for high level specification and system cosimulation. Coware N2C provides a Virtual Bus [13] to connect each system block and allows the HW/SW cosimulation at the conceptual and architectural level. VCC (Virtual Component Codesign) [3] proposed by Cadence is a system-level environment for HW/SW co-design and IP reuse. This tool allows specifying the system functionality, defining the system architecture, performing the partitioning, refining communications between blocks and analyzing system performances. However, such tools require the system designer to have an efficient IP core modelling adapted for the co-simulation and system-level performance analysis steps. Furthermore, they can not manage low-level details relative to IP interface synthesis such as computing latency and I/O timing constraints.

### 3 Modelling of Reconfigurable Architectures with SystemC

As mentioned before, to design a low power and high performance design in complex SoC designs, a vast number of configurations in a rapid and efficient manner should be explored.

In order to verify the functionality of the many different configurations generated by platform transformations, there is a need for a fast simulation. For conventional hardware verification, we mainly rely on gate-level HDL simulation extensively. Unfortunately, gate-level simulation requires too much run time and memory space for million-gate SoC designs. Secondly, we need to evaluate each configuration's power/performance characteristics quickly and accurately in order to aid the selection of the best configuration. Gate-level HDL simulation can accurately evaluate power consumption, but it is too slow for SoC platforms, which may require tens of hours per configuration.

RT-level simulation is faster, but still too slow. Therefore, we are using a C-based system-level simulation in order to simulate each SoC platform configuration within a reasonable period of time, for both functional verification and power/performance evaluation.

SystemC provides a standard and well defined interface for description of the interconnections between modules (ports and signals). Moreover, among the advantages of C/C++ based hardware descriptions, there is the possibility of bridging the hardware/software description language gap [11].

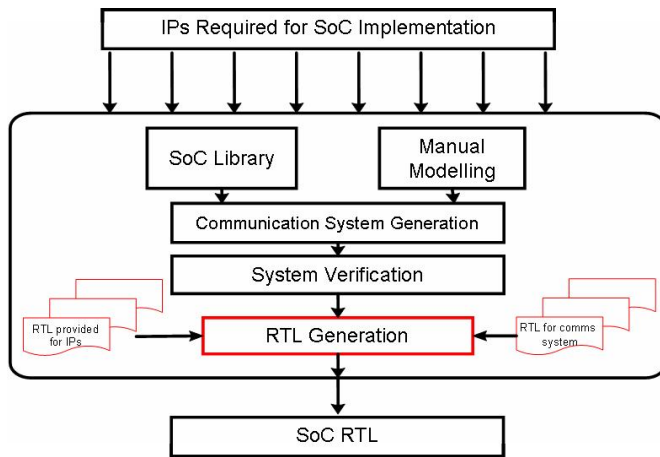
This will require the development of SystemC models for each SoC platform components as objects which then could be used to model the complete SoC platform as communicating objects. For each component, a gate-level power analysis will be performed, using a variety of parameter selection and representative input vectors. The power information obtained for each platform component will then be back-annotated to its SystemC object. It is important to notice that gate-level simulation (for power analysis) has to be performed only once for each component. Therefore, when executing the complete SystemC model of a particular configuration, fast and sufficiently accurate power estimates will be obtained.

Transaction Level Modeling (TLM) is a top-down approach to system design widely adopted in the verification phase of the design. In TLM, the system is first described at a high level of abstraction where communication details are hidden, then the description is refined with the necessary details.

The key concept of TLM is the separation of functionality definitions from communication details, to achieve this, TLM uses the concept of channel. A channel allows communication through methods calling. Then, two functional components, interconnected with a channel, communicate calling the methods exported by the channel. In this way, the communication detail is hidden in the channel definition. SystemC enables TLM modeling through `sc_interface`, `sc_channel`, and `sc_port`. A channel interface can be derived from `sc_interface`, then the `sc_interface` can be implemented in an `sc_channel`. At this point an `sc_module` can communicate through an `sc_channel`, using a `sc_port` connected to the `_sc` channel.

### 4 Design Flow

As mentioned before, to cope with the complexity of applications, a natural approach is to reuse previously designed subsystems, also called silicon intellectual properties (IP). One can obtain IP from previous generations of product development or from independent third party IP vendors or even open sources. However, integrating IP from multiple sources in various forms is nontrivial. All IP must have



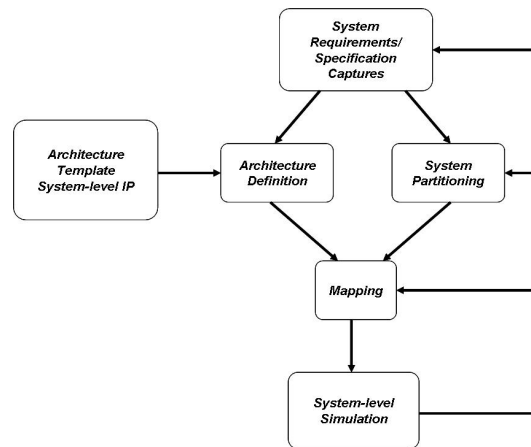
**Figure 1. Design Flow**

accurate models for various stages of design flow, including synthesis, simulation, power, test, layout etc. Our methodology is to deal with hybrid SoC systems which include custom reconfigurable IP cores, conventional parameterisable IP cores, interconnected using mixed interconnection schemes which are not necessarily based on conventional bus systems. Power, Area, and accuracy of various algorithms are captured and modelled to be integrated in a tool for automatic generation of hybrid interconnection driven System-on-Chips. Figure 1 shows the main steps of our proposed SoC design methodology.

In order to describe an IP, it is necessary to specify it initially as an entity with its high level functional description. In our work, an IP model describes an IP for its functionality and performance parameters. This means that the IP model corresponding to a given IP describes following items:

- Ports: name, direction (input, output or bidirectional), width and type (bus function, irq, or extern).
- Wrapper type that IP is attached.
- Behaviour: Master or Slave.
- Parameters include
  - width of different ports
  - operation mode is an IP-specific option which refers mostly reconfigurable IPs with ability of reconfiguration for different functionalities.
  - speed and power modes are very interdependent options, which can be determined according to the desired overall and local performance and power consumptions.

It should be noted that the main part of an IP model is its transaction-level based modelling in SystemC which represents the exact functionality of the IP core. We have chosen SystemC as the simulating language, since it provides both HW and SW components simulation and verification in a single environment. IEEE has recognised this language as standard for doing simulation and verification at system level [8]. The proposed modelling and simulation flow in SystemC is shown in Figure 2.



**Figure 2. IP modeling and simulation flow in SystemC**

Modeling different architectural choices for a given application which will be optimised in terms performance versus either given constraints or default design constraints after the application analysis. Proposing the optimised reconfigurable the architecture for a given application by exploring the different 'design space of the architecture' for reconfigurable architectures. Translating application onto a data flow graph or a hybrid architecture depending upon application requirement. Partitioning the application using hardware (HW)-software (SW) partitioning methods and algorithms; here we may use best existing HW-SW partitioning methods and algorithms for our application with two levels: one level for basic partitioning that is HW-SW tasks and other level is reconfigurable logic block and fixed HW partitioning. Design and implementation of the optimised algorithms for mapping of the design library on to the proposed reconfigurable architecture. Then systemC simulation results are compared to the original IP outputs to verify correctness of its functionality.

## 5 Basic Platform

The platform architecture is shown in Figure 3 [6], also indicating the possibility of including additional IP mod-

ules connected to the AMBA AHB bus. LEON is compliant with the 32-bit SPARC V8 architecture (the IEEE-1754 standard) [6]. The main features of the LEON microprocessor includes: hardware units for multiplication, division and Multiply-And-Accumulate (MAC) operations, interface to Floating-Point Unit (FPU) and custom Co-Processor (CP), and separate instruction and data cache (Harvard architecture). It also incorporates the AMBA 2.0 on-chip bus architecture [1]. The memory controller handles 8/16/32-bits wide buses for external memories. There is also support for extra local RAM for performance critical code.

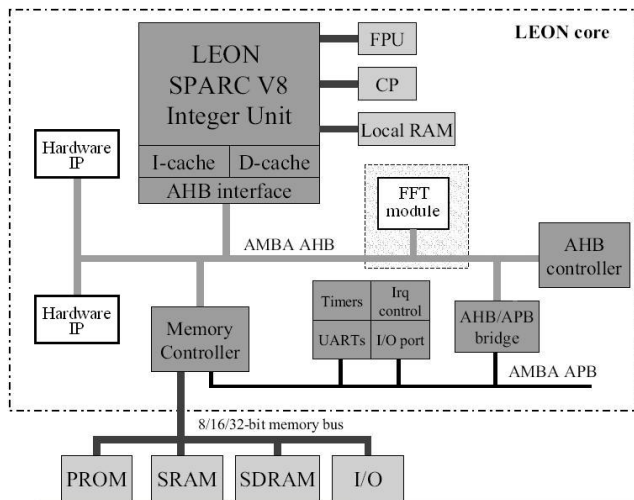


Figure 3. Platform architecture

The memory controller handles 8/16/32-bits wide buses for external memories. There is also support for extra local RAM for performance critical code. FFT is a widely used application which is considered in our work as a case study. A Reconfigurable FFT IP core as a hardware accelerator can be integrated into to the platform. The reconfigurable and low power architecture of this FFT is described in the next section.

## 6 FFT

In digital signal processing, Discrete Fourier Transform (DFT) is one of the most widely used algorithms in digital signal processing for spectral analysis and for filter implementations. It operates on an  $N$  point sequence of numbers  $x(n)$ . However, the DFT has limitation in speed for larger transform sizes. As a result of that Fast Fourier Transform (FFT) is used, which is an efficient algorithm for computation of the DFT.

Recent advances in VLSI technology have given rise to a new class of computer architectures which take advantage of application-level parallelism. These reconfigurable computers can be quickly customised at the hardware

level to perform exactly the computation required in hardware, overcoming the fixed hardware configurations found in many contemporary microprocessors. The main motivation is to make reconfigurable hardware as the main computation core. In this case, each application will be translated into hardware (Application-Specific Hardware) automatically. The processor will only be relegated to support tasks.

The overall system architecture is shown in the following (Figure 4), which is originally implemented in RTL (Register Transfer Level) [15]. The whole architecture consists of six main blocks. The control block controls the overall dataflow and generating configuration signals. The Radix-2 based butterfly block (BB) is used for butterfly calculation. Coefficients for butterfly calculation are provided by coefficient memory cluster (CMC). Data switch (DS) is used to route the data into the right data memory cluster (DMC). The inputs of DS are actually the outputs of butterfly block. Address generation block (AGB) is used to generate the address for each DMC. Address switch (AS) exchanges the addresses and sends them into the right memory module within the DMC.

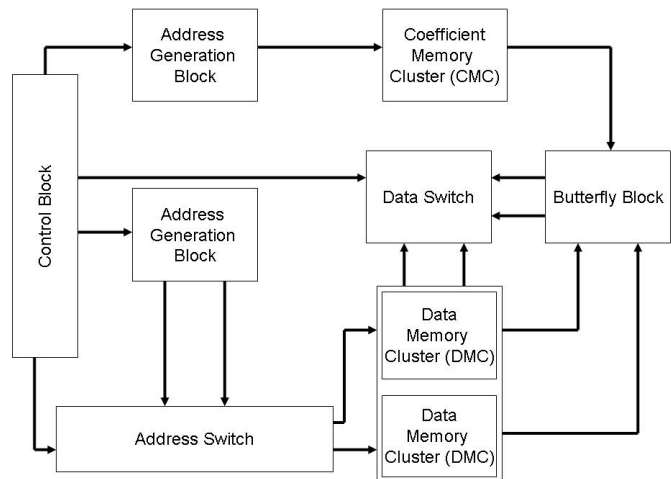


Figure 4. Block diagram of Reconfigurable FFT Processor

### 6.1 Reconfigurable FFT Architecture

The detail overviews of each module are discussed below:

- **Control Block:** This is the main control unit of FFT fabric. Two different counters are used for butterfly operations. First one is used to count the number of butterfly operations in each pass and then second one is used to count the number of passes. It generates the

inputs for Address Generation Blocks, the control signals for Data Memory Clusters and the configuration bits for Data Switch and Address Switch.

- **Butterfly Block:** Figure 5 shows the data path of butterfly block used for this architecture. This butterfly block has two parallel inputs consisting of imaginary and real parts and two parallel outputs in the same format. The complex FFT coefficients are represented by  $W$ . Each butterfly computations contains add, sub and multiplications.

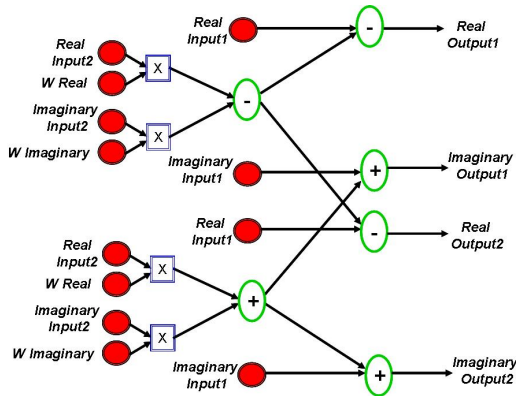


Figure 5. Data path of Butterfly Block

- **Coefficient Memory Cluster (CMC):** In this reconfigurable fabric, CMC is divided into 64 small modules, which are known as Coefficient Memory Modules. Each CMM can store 8 coefficients. So overall CMC can store 512 coefficients. All the CMMs are grouped together by using glue logic to form the CMC. The logic behind to combine a pair of CMMs is taken from Wilton [14]. Wilton [14] made a prototype for his proposed configurable memory which has 4 memory blocks. Wilton denotes this as Basic memory blocks (BMBs). Each BMB contains 1 Kbit of memory. Flexibility is provided in two ways: the aspect ratio of each BMB is configurable, and the BMBs can be combined in a flexible manner. First, consider the aspect ratio of each BMB.

Although the physical layout of the array is fixed, a programmable multiplexor connected to the data lines of each BMB allows each BMB to appear to the user as a 1024x1, a 512x2, a 256x4, or a 128x8 memory. The aspect ratio of each BMB can be configured independently. Second, a flexible interconnect matrix is provided to allow the user to program for combining the BMBs to implement larger user memories.

- **Data Memory Cluster (DMC):** As shown previously in the main block diagram, this architecture has two

DMCs in total. Each of these is divided into sixty-four 8x32-bit dual port memories. Therefore, each DMC has a capacity of 512x32-bit. The same Wilton logic [14] is used here for combination, which is explained earlier. Different from CMC, the most significant bits of addresses received by DMCs are used as configuration bits for each pair of data memory modules (DMM). Each DMM can be turned on and off separately and automatically by the configuration bits, so that the power consumption in unused memories is reduced.

- **Address Generation Block (AGB):** Two address generation blocks are designed to generate addresses for both DMCs and CMC. Different from the address blocks in conventional fixed-point FFT fabrics, these reconfigurable ones can generate addresses for various FFT sizes. In [9], an efficient coefficient address generation method was proposed by Cohen. According to his work, for a  $N = 2n$  point FFT, at the  $b^{\text{th}}$  butterfly calculation in the  $p^{\text{th}}$  pass,  $(W_0)^k = (\exp(-2\pi i/N))^k$  is the coefficient needed. The  $k$  can be obtained from  $b$  by masking out its  $(n - 1 - p)$  least significant bits.
- **Data Switch (DS) and Address Switch (AS):** The data switch receives the outputs of the butterfly module and routes them into the right DMCs. For all the FFTs ranging from 16-1024 points, the AS input width is 9 bits. Address switching works in parallel with data switching in order to allocate the right addresses for the data which are being processed through the data switch block.

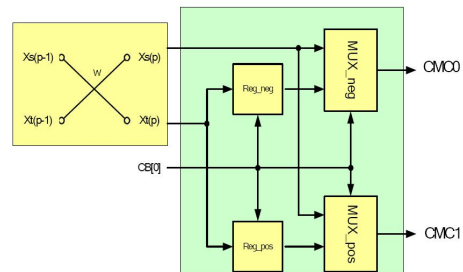


Figure 6. The structure of Switch Block

## 7 SystemC Implementation

The system FFT block computes a 16-(4096) points FFT on a sequence of complex inputs by using a radix-2 decimation in frequency algorithm. The input data is read as a signed 16-bit fixed-point number with 10 fractional bits.

Twiddle factors and output values are in the same representation. Internally in the block, computation is performed with fixed-point arithmetic. The input samples and output transformations are externally inferred as 16-bit integers.

The fixed-point functional version of the FFT was developed to be compatible with original synthesizable FFT IP core and verify the results, working at the highest level of abstraction. The FFT block initiates reading of a data sample by assertion of the *Data Request* signal. Next it waits for the *Data Valid* signal to assert. Then it de-asserts the *Data Request* signal and reads from the *Real Input* and *Imaginary Input* ports. The FFT block reads input samples of data.

After the FFT calculation is performed, the block writes the transformed values to a sink block in the testbench. It writes the real and imaginary components of the transformed value on the *Real Output* and *Imaginary Output* ports. Next, it asserts the *Data Ready* signal, indicating that the FFT is ready to read data from its ports. It waits for the *Data Acknowledge* signal to assert, then it sends the next set of values.

For verification of the FFT implementation, we have used some testbenches. The FFT testbench consists of three files: source, sink, and main\_fft.

- The source file reads in real and imaginary samples from input files, which are ASCII files containing values. The source block interacts with the FFT behavioral block using two-way handshake.
- The sink file reads the real and imaginary components of the output transform values from the FFT block. It writes the value to output files, which are ASCII format. The sink block also interacts with the FFT block by using two-way handshake.

## 8 Result Analysis

Hardware gate level simulation is carried out and Synopsys design compiler is used to calculate power and area results considered. 0.13 micron technology library is used for ASIC synthesis. Power and area results are shown in Table 1, which will be used later in our tool for obtaining an optimised overall power and area consumptions. Moreover, our modelled reconfigurable FFT architecture is compared with some existing FFT architectures in terms of power consumption. In [10], FFT consumes 545mW for a 64-point FFT at 65 MHz and 574 mW for a 2048-point FFT at 60 MHz. The power dissipation of the processor in [5] equals 300mW for a 2K FFT at 20MHz, and according to [2], it is 300mW for a 1K FFT at 20 MHz as well. As can be seen in Table 1, our FFT model gives a better power consumption results in all these cases.

Second part of the implementation results are simulation times, which should be compatible with the original IP

core, to be used for system-level verification at IP integration stage (See Figure 7). In timing analysis, the number of clock cycles is measured which are needed to compute the FFT for the corresponding FFT size. Furthermore, the timing distribution of clock cycles is shown in Figure 8. This figure demonstrates that, DMC needs nearly 50% of the overall simulation time and CMC takes about 30% of the total time. Therefore, memory clustering blocks use most of the execution time. Butterfly block and address generation blocks take next places of most time consuming parts, respectively. The results of both Figures 7 and 8 show the compatibility of our SystemC model, with our original IP core.

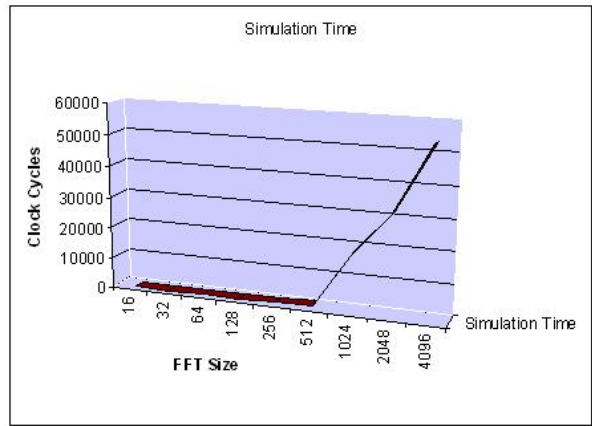


Figure 7. Simulation Times of FFT

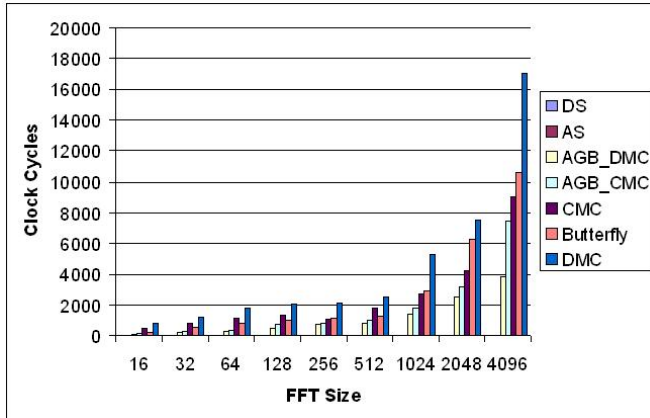
FFT Size	16	32	64	128	256	512	1024	2048	4096
Power (mW)	4.8	8.0	8.3	13.0	26.1	49.8	81.8	93.2	104.6
Energy (nJ)	9.6	38.4	93.0	332.8	1503	6374	23034	28905	35471
Area (mm <sup>2</sup> )	0.07	0.11	0.19	0.36	0.58	1.09	2.51	3.284	4.864
Clock (mHz)	20	20	20	20	20	20	20	20	20

Table 1. Power and Area Consumptions

## 9 Conclusion

In this paper, we presented a design methodology of IP integration in a SoC design that exploits both IP designer and SoC integrator constraints. The integration task is based on SystemC IP delay models that describe low-level details of the IP execution constraints.

These models can be deliverable since the internal features of the IP core are hidden. We also modelled recon-



**Figure 8. Distribution of time for reconfigurable FFT blocks in different FFT sizes**

figurability attribute of the FFT core, which can be parameterised widely for different FFT sizes with reconfiguration.

In future work, we plan to model the whole platform including the microprocessor, peripherals, and more hardware IPs for wireless communication applications to build up our IP library for automation of SoC generation.

## References

- [1] ARM. *AMBA 2.0 specification*. <http://www.arm.com/armtech/AMBA>.
- [2] B. Baas. A low-power, high-performance, 1024-point fft processor. *IEEE Journal of Solid-State Circuits*, 34(3):380–387, 1999.
- [3] Cadence. *Cadence VCC 2001*. <http://www.cadence.com/datasheets/vcc.html>.
- [4] P. Coussy, A. Baganne, and E. Martin. A Design Methodology for Integrating IP into SOC System. In *Proceedings of the IEEE 2002 Custom Integrated Circuits Conference*, pages 307–310, 2002.
- [5] E. Bidet, D. Castelain, C. Joanblanq, and P. Senn. A fast single-chip implementation of 8192 complex point fft. *IEEE Journal of Solid-State Circuits*, 30(3):300–305, 1995.
- [6] Gaisler Research. *Hardware Manual*. <http://www.gaisler.com>.
- [7] Z. J. Han, Q. and W. Jia. IP Reusable Design Methodology. In *Proceeding of 4th International Conference on Application Specified Integrated Circuit*, pages 756–759, 2001.
- [8] IEEE Standards Association. *IEEE Standard SystemC Language Reference Manual*. <http://standards.ieee.org/>.
- [9] J.W. Cooley and J. Tukey. An algorithm for the machine calculation of complex fourier series. *International Journal of Mathematics of Computation*, 19:297–301, April 1965.
- [10] J.-C. Kuo, C.-H. Wen, and A.-Y. Wu. Implementation of a programmable 64/spl sim/2048-point fft/fft processor for ofdm-based communication systems. In *ISCAS (2)*, pages 121–124, 2003.
- [11] G. D. Micheli. Hardware Synthesis from C/C++ Models. In *Proceedings of the Design Automation and Test in Europe (DATE)*, pages 382–383, 1999.
- [12] W. Savage, J. Chilton, and R. Camposano. Ip reuse in the system on a chip era. In *ISSS '00: Proceedings of the 13th international symposium on System synthesis*, pages 2–7, Washington, DC, USA, 2000. IEEE Computer Society.
- [13] K. van Rompaey, D. Verkest, I. Bolsens, and H. D. Man. Coware - a design environment for heterogeneous hardware/software systems. In *Design Automations for Embedded Systems, 1(4)*, 357-386, 1996.
- [14] S. J. Wilton. Embedded memory in fpgas: Recent research results. In *IEEE Pacific Rim Conference on Communications, Computers and Signal Processing*, 1999.
- [15] Y. Zhao, A. T. Erdogan, and T. Arslan. A low-power and domain-specific reconfigurable fft fabric for system-on-chip applications. In *19th International Parallel and Distributed Processing Symposium (IPDPS)*, 2005.