ISSN: 2456-9348
Vol (03) _Issue (07) Impact Factor: 4.520

iJETRM
International Journal of Engineering Technology Research & Management

CONCISE METHODOLGY FOR DEVELOPMENT OF COMPLEX DIGITAL ELECTRONIC SYSTEMS

Marcus Lloyde George
Ultimate Virtual Market Limited, San Fernando, Trinidad and Tobago
marcus.george99@yahoo.com

ABSTRACT
Development of simple digital electronic circuits can be easily done using combinational logic theory such as Boolean Algebra and Karnaugh Maps, along with tools such as Xilinx Schematic Editor. Even medium-sized systems can be developed this way. What about complex digital electronics systems? Utilizing the same procedure may be tedious and hence will be best done using a more effective and structured procedure, along with use of hardware descriptive languages. This paper therefore presents a concise methodology for the development of complex digital electronics systems. The paper will focus on all aspects including the specification, formal specification and modelling of the system, the choice of development platforms, hardware description, and even the formal verification and validation of such system. An example will be used as the basis of the formal specification and modelling aspect of the paper to ensure that readers can quickly absorb the material presented. The author intends for the contents of this paper to be informative and expects that developing digital design engineers find this methodology useful in development of systems they attempt to pursue.

KEYWORDS:
Hardware Design, Hardware Design Procedure, Digital Design, Formal Specification and Modelling, Formal Verification, Formal Validation

INTRODUCTION

Hardware design and implementation involves taking a specification through the steps required to represent that specification in hardware [1]. An understanding of design methodology, hardware description and verification and validation are necessary to the successful development of the required hardware [1]. This chapter presents the structured approach to be utilized in the development of the novel multi-precision floating-point multiplier architecture to be developed in the subsequent chapters of this paper. Formal specification and modelling of digital systems will be presented in this chapter. Formal verification and validation techniques for digital systems will also be presented. The concept design of the precision floating-point multiplier architecture will also be derived in this chapter.

FORMAL SPECIFICATION AND MODELLING OF DIGITAL SYSTEMS

Formal Specification and Modelling involves partitioning of system into two components: Datapath and Controlpath. The Datapath is responsible for the processing of data entering the system, including operations such as addition, multiplication, rounding, etc. The Controlpath on the other hand is responsible for the management of the data processing process, hence the management of the Datapath. It assumes the role of manager/controller in the digital system. In formal specification and modelling a structured design approach by Finite State Machine-Datapath (FSM-D) modelling is utilized. In this case the Controlpath is implemented as a finite state machine and the Datapath is managed by it. The concept of (FSM-D) modelling is shown in Figure I.

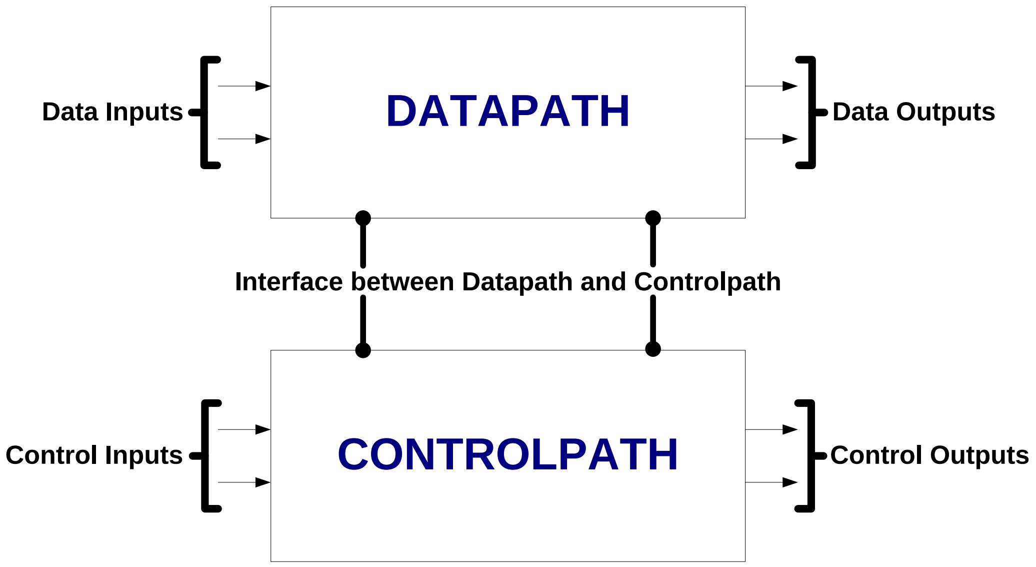

FIGURE I: CONCEPT OF THE FSM-D ARCHITECTURAL MODEL

The specification and modelling procedure consists of several important stages, all crucial in guaranteeing corrected functionality of the system to be developed. The first step involves the development of the FSM-D interface definition that can be easily derived from the system specifications. For this section assume a system *TEST_X* requires the specification and modelling of a unit which implements equation (1). Also, consider that data processed in this system will be in 32-bit floating-point format.

$$h(n) = k \cdot h(n-1) + 8 \cdot p(n) \cdot q(n) \tag{1}$$

From the specifications the system will have three 32-bit inputs *k*, *p(n)* and *q(n)* and a 32-bit output *h(n)*. The system will have a controller and as such will have clock '*clk'* as well as a reset '*RESET*' for initialization. The overall process must be initiated by an input port '*start'* and the indication of the end of the process will be done via an output port named '*done'*. Figure II gives the FSM-D interface definition for System TEST_X.

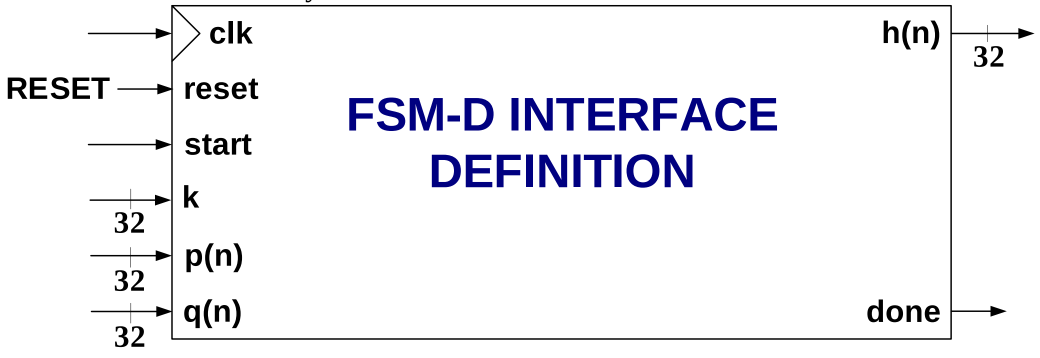

FIGURE II: FSM-D INTERFACE DEFINITION FOR SYSTEM TEST_X

The second step involves the development of the data flow diagram using the system specifications. Dataflow diagrams represent systems as a series of data manipulation and transfer operations. When developing a dataflow diagram the arithmetic operations required must be first identified. In digital signal processing each operation is represented by a symbol as shown in Figure III.

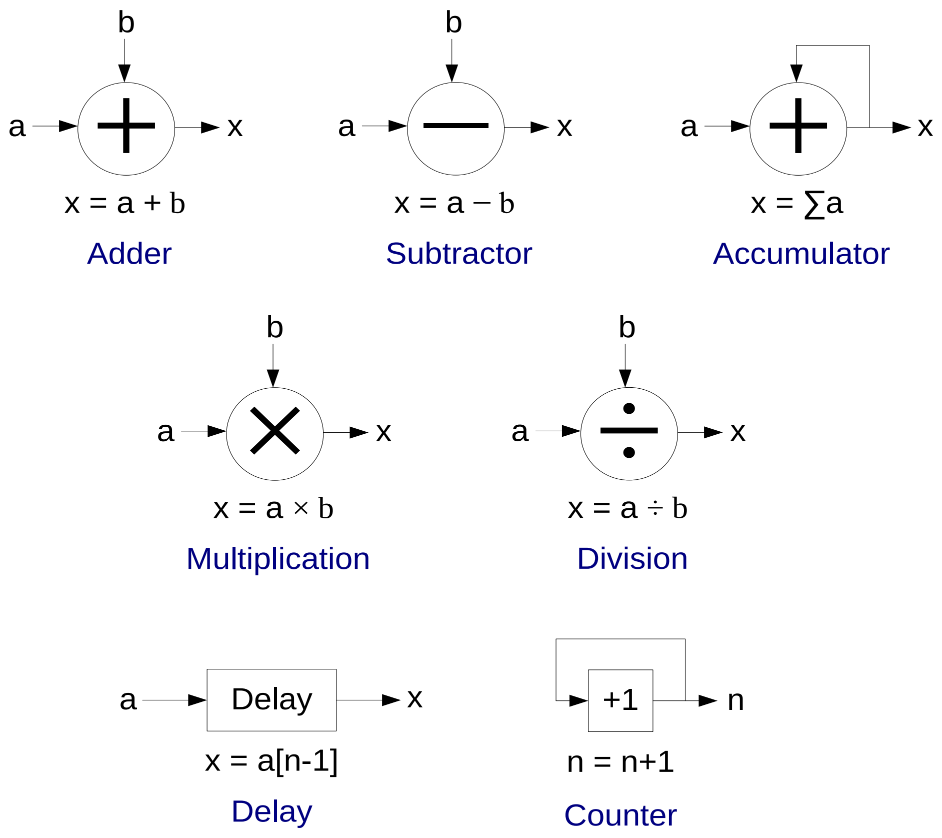

FIGURE III: SYMBOLS FOR DIGITAL SIGNAL PROCESSING OPERATION

The third step of the formal specification and modelling procedure involves the construction of the datapath design of the required system, for this process the dataflow diagram is utilized as it provides less descriptive version of the datapath design. The specification and modelling procedure may involve the application of pipelining to the dataflow diagram. In applications where the path delay of the system is substantially shorter than the time elapsed between application of inputs, there is no need to apply pipelining e.g. if the system that implements equation (1) has a path delay of 16ns but is required to perform the computation of *h(n)* once every 9 ms, then a pipelined version of the system that implements equation (1) is not required. The use of registers will be solely for stabilization of inputs of components in order to have their outputs valid for a longer period of time, eg. if a floating-point multiplier was utilized for computation of the product of *k* and *h(n-1)* but however the multiplier kept the result $k \cdot h(n-1)$ valid for half a clock cycle after the result was computed. As such a pipeline register would have been required to hold this product stable for longer than half of a clock cycle.

The forth step of the formal specification and modelling procedure involved the development of the datapath block diagram of the system using the dataflow diagram developed in the previous step. This required the inclusion of the ports and the respective port widths for all components utilized. Ports interfacing with the control path were also included, hence making the development of the FSM-D model for the system easier. The datapath block diagram of the system that implements the equation (1) is given in (Figure IV).

ISSN: 2456-9348
Vol (03) _Issue (07) Impact Factor: 4.520

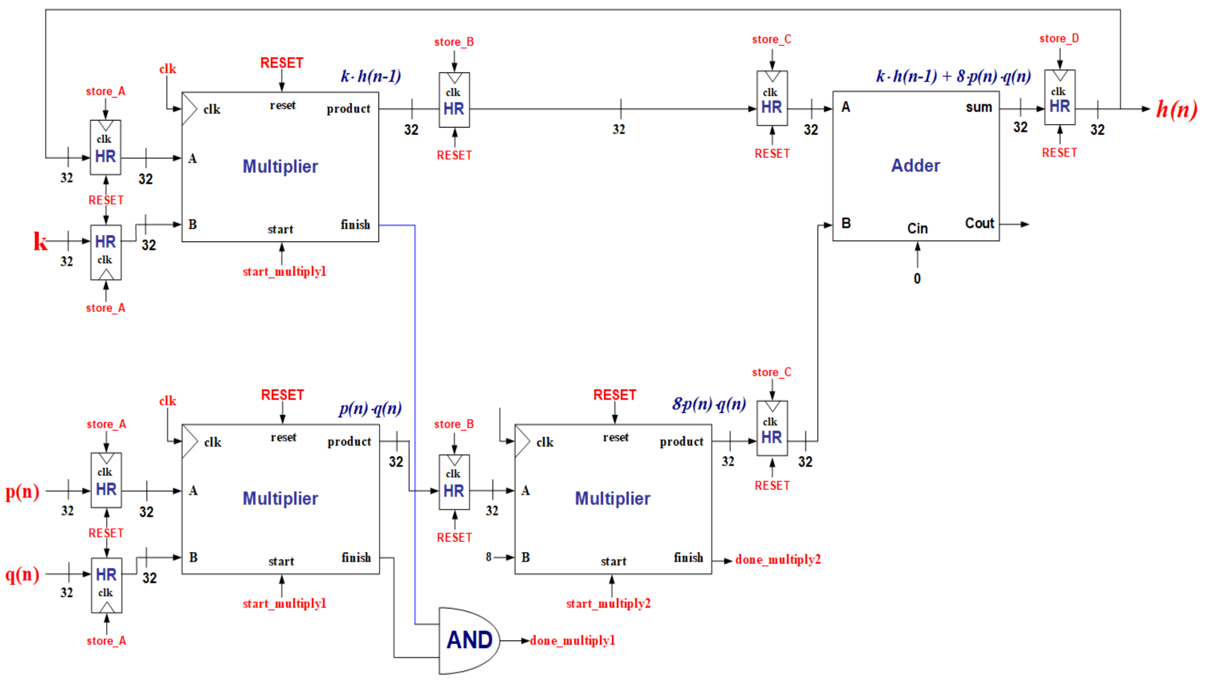

FIGURE IV: DATAPATH DESIGN OF THE UNIT IMPLEMENTING SYSTEM TEST_X

The fifth step consisted of the development of the FSM-D model for the system. First the Datapath Interface Definition was developed by analysing the datapath and deriving its inputs and outputs. The inputs and outputs making up the interface of the datapath with the controlpath were carefully derived and identified. The Controlpath Interface Definition was then derived using the datapath and the FSM-D interface definition obtained previously. Both datapath interface definition and controlpath interface definition were combined to give the FSM-D model for the system. The FSM-D model for the system that implements the equation (1) is given in Figure V.

ISSN: 2456-9348
Vol (03) _Issue (07) Impact Factor: 4.520

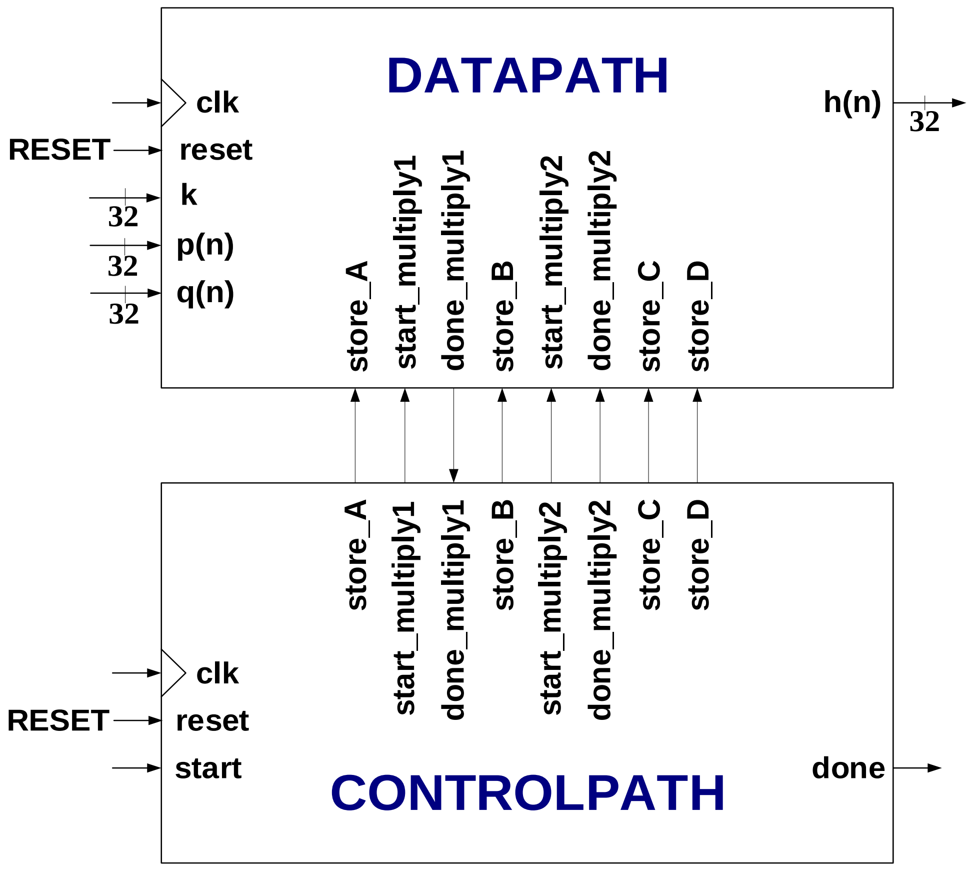

FIGURE V: THE FSM-D ARCHITECTURAL MODEL FOR SYSTEM TEST_X

The sixth and final step involves the design of the controlpath of the system by analysis of the operation of the datapath design and system specs. First the state transition table [2] is derived, after which the state diagram [2] is constructed. The state diagram of system TEST_X is seen in Figure VI.

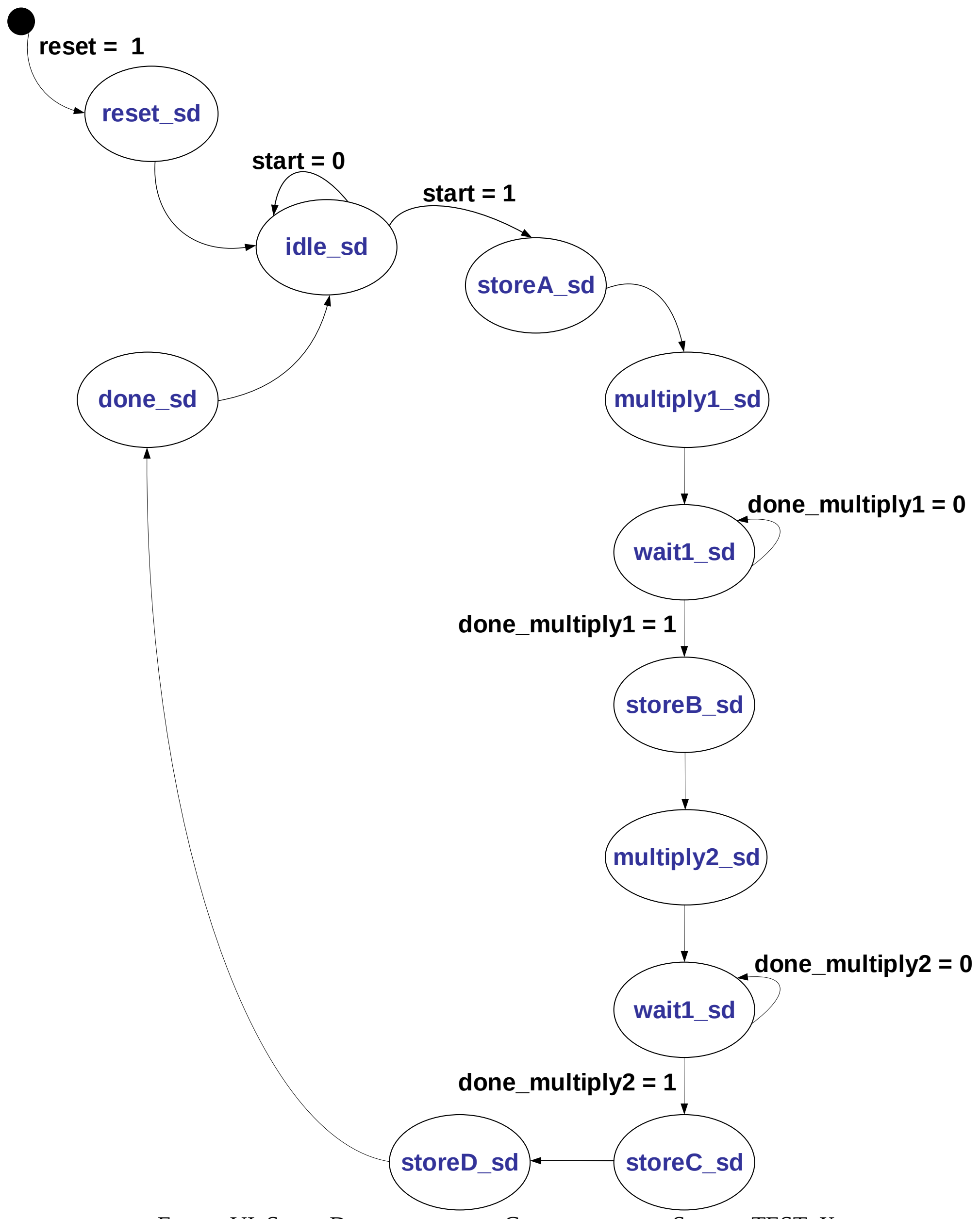

FIGURE VI: STATE DIAGRAM FOR THE CONTROLPATH OF SYSTEM TEST_X

CHOICE OF DEVELOPMENT TECHNOLOGY FOR SYSTEM IMPLEMENTATION

This section will compare three hardware development technologies: Complex Programmable Logic Devices (CPLDs), Field Programmable Gate Arrays (FPGAs) and Custom Application Specific Integrated Circuits (ASICs). This section will give the strengths and weaknesses of these technologies.

FPGAs consist of an array of gates that must be configured into specific hardware blocks. These blocks may range from registers, adders and multipliers to more complicated units that are capable of performing FIR filtering operations and FFT [5]. The main difference between FPGAs and CPLDs is that FPGAs consisted of a significantly larger number of gates than CPLDs. Therefore, much more complex digital circuits can be implemented on FPGAs than CPLDs. The high capacity of FPGAs for parallel processing makes them beneficial for tasks that require high computational power and involve a great deal of repetitive processes. FPGAs may require dedicated hardware resources for every configuration and datapath for programmes that involve extensive conditional evaluations [5].

ASICs are microchips developed for specific applications such as speech processing and frequency metering, amongst others. Unlike FPGAs and CPLDs, ASICs are not programmable and provide only a particular set of functions. ASICs provide several advantages over these technologies such as higher power efficiency and lower delay.

HARDWARE DESCRIPTION

Hardware description languages (HDLs) such as VHDL, AHDL, Verilog and ABEL provide an alternative to Boolean equations or gate-level description of digital components. HDLs also provide high-level constructs that enable designers to describe large circuits such as the development of libraries that contain components that can be reused in subsequent designs. VHDL is the standard language for hardware description [3]and enables code to be portable between both synthesis and simulation tools and device-independent designs.

After implementation using an HDL the digital circuit is synthesized. The process of synthesis consists of the conversion of a high-level description of the system design into an optimized version of the gate-level representation of the digital circuit given design constraints and a standard-cell library. Computer aided logic design tools such as Xilinx ISE and Quartus II can greatly reduce the design cycle time of digital circuits. With these synthesis tools designers are able to produce technology-independent, high-level descriptions of digital systems as well as produce technology-dependent gate-level net-lists for the required digital system.

FORMAL VERIFICATION AND VALIDATION OF DIGITAL SYSTEMS

This section presents two important aspects of digital design – verification and validation. These two processes are responsible for determining if a digital system meets required specifications and at the same time fulfils its intended purposes.

A. Verification of Digital Systems

Verification testing is used to confirm that an implemented system functions as required. Verification testing is done by creation of VHDL test-benches in Xilinx ISE for execution via ISim that is initiated from the Xilinx ISE Design Suite window. A VHDL test-bench requires the user to define input stimuli along with timing constraints such as period of the reference clock, delays after inputs changed, etc. To thoroughly verify a digital system, test cases must be selected to represent input samples belonging to each of the following categories:

(i) zero inputs

(ii) non-zero inputs
(iii) mixture of zero and non-zero values
(iv) small values
(v) medium-sized values
(vi) large values
(vii) mixed small, medium and large values

The verification process is completed by comparison of actual results of functional and timing simulation from ISim with expected results. Verification testing is considered a success once there is a 100% match between expected and actual values. Functional simulation was used for logic debugging while timing simulation was employed only for resolving actual timing issues.

B. Validation of Digital Systems

Validation testing is used to establish that a system satisfied an operational requirement [1]. Timing simulation of the implemented system can be used to determine the block latency in nanoseconds or cycles [1]. Timing simulation can be obtained using ISim from the Xilinx ISE Design Suite window. The ISim simulator provides designers with a complete, full-featured HDL simulator that is integrated within the Xilinx ISE Design Suite. Parameters such as path delay and hardware utilization can be found from synthesis reports, however a better indicator of system performance is determined from the Post Place and Route Static Timing Report from Xilinx ISE Design Suite.

B.1 Maximum Clock Frequency

The maximum clock frequency is related to the maximum delay for signal propagation in the system, and hence changing signals faster than this will result in unexpected behaviour of the system. The maximum clock frequency of a given system depends on several factors such as:

(i) gate delays - signal delays due to logic gate transitions
(ii) wire delays - delays associated with signal propagation along wires
(iii) clock skew

The predominant sources of delay in digital signals come from gate and wire delays.

Proper chip operation requires that path delay constraints must be satisfied whenever digital signal transition traverse a combinational logic path. Confirmation that every path delay constraint is satisfied is the most critical verification task that a designer will face. Performing this dynamically is infeasible because:

(i) it is computationally difficult to establish the number of possible combinations of state and input variables that can result in transitions for given combinational paths.
(ii) an exponential number of combinational paths may exist through the logic system. Designers normally statistically signoff on timing of logic circuits using a method that pessimistically assumes that the logic paths can be sensitized.

The timing closure framework is based solely on static timing analysis that is an efficient, linear-time verification procedure that identifies critical paths of logic circuits.

B.2 Static Timing Analysis - Post Place and Route Static Timing Analysis

After placement and routing of a design on Xilinx ISE Design Suite, a post-place and route static timing report is generated. This report incorporates timing delay information hence providing a comprehensive timing summary of your system. The report's contents can be customized in order to assist in determining how well your system meets timing requirements. The system can be redesigned in order to use less logic levels, allocate more time for specific paths or even for faster operation on device. Post-place and route static timing allows you two reports: an error report listing timing errors and all information on associated net and path delay, or a verbose report listing information on delay for all nets and paths. Generally, the static timing report is accurate since it uses the worst case scenarios. The Post-Place and Route Static Timing Report is used as a final analysis of whether or not the design has met all timing constraints.

Based on the timing analysis features in Xilinx ISE Design Suite, the following can be deduced about the estimated design performance:

- All performance estimates for the system are available prior to completion of system implementation.
- Generated Synthesis Reports
  - logic delays indicated are accurate
  - routing delays indicated are based on fan-out
  - the performance reported generally has a maximum error of 30%
- Post-Map Static Timing Report
  - logic delays indicated are accurate
  - routing delays indicated are based on placement and fan-out
- Post-Place and Route Static Timing Report
  - logic delays indicated are accurate
  - routing delays indicated are based on placement, routing and fan-out
  - most accurate timing report generated and uses the worst case scenarios
  - the performance reported generally has a maximum error of 3%

One very important point to note is that this simulation tools from Xilinx ISE Design Suite are adaptive to the clock frequency of the hardware of the host platform used in simulation. The performance of the post-place and route static timing report the performance reported generally has a maximum error of 3%. The structure of the Post-Place and Route Static Timing Report is:

- Timing Constraints
  - Indicates the number of paths covered and also the number of paths that failed for each constraint considered
  - Provides detailed descriptions of the longest paths for the design
- Data Sheet Report
  - Indicates the setup-times, hold-times and clock-to-out times for each I/O pin for the system
- Timing Summary
  - Indicates timing errors, clearly stating the number of failing paths for the system
  - Indicates the timing score which is the total time of all constraints that were missed
- Timing Report Description
  - Allows designer to duplicate the report

The following is the attributes of a sample post place and route static timing report:

- Constraint Summary
  - Number of paths analyzed (defaults to the three longest paths)
  - Number of timing errors
  - Length of critical path (slowest delay paths for each constraint indicated)
- Total Delay
  - Clock and data breakdown
- Clock Jitter Analysis (Slack, Required Arrival time (RAT), Actual Arrival Time (AAT))
    - Slack = AAT(v) - RAT(v)

- maximum path delay, requires that AAT at each node never exceed the RAT, hence for all nodes: AAT(v) - RAT(v) must always be negative.
- Critical paths or critical nets are signals that have negative slack, while non-critical paths or non-critical nets have positive slack.

- Detailed Path Description
  - Delay types are described in the data sheet
  - Worst-case conditions are assumed, unless pro-rated

C. Performance Verification using Post-Place & Route Simulation Model

At the end of the verification and validation processes of section A and B of this paper respectively, it is useful to evaluate the system's performance by generation of the Post-Place and Route Simulation Model. With this option you would be able to generate a simulation model after placement and routing of the design. In this process the NetGen converts the post place and route static timing results (in the form of a NCD file) to a simulation model (in the form of a structural SIMPRIM-based VHDL file) and a Standard Delay Format (SDF) file. The SDF file consists of true timing delay information for the design. The simulation model file and SDF file may be used for verification of the functionality and timing of the design.

Whereas the synthesis report may have a maximum error of approximately 30% and the post place and route static timing report may have a maximum error of approximately 3%, this post place and route simulation model report is a bit more accurate. However, the post place and route static timing report may be sufficient to verify the functionality and timing of the system.

CONCLUSION

This paper therefore presented a concise methodology for the development of complex digital electronics systems. It focused on all aspects including the specification, formal specification and modelling of the system, the choice of development platforms, hardware description, and even the formal verification and validation of such system. An example was utilized in the presentation of the methodology to ensure that the material can be better understood. It is expected that developing digital design engineers find this methodology useful in development of systems they attempt to pursue.

ACKNOWLEDGEMENT

The author would like to thank the staff at Ultimate Virtual Market Limited for their contribution to the development of this paper. We are grateful for the support provided by the company in this research.

REFERENCES

[1] George, Marcus, and Geetam Singh Tomar. 2015. "Hardware Design Procedure: Principles and Practices". *5th International Conference on Communication Systems and Network Technologies, 4-6 April, 2015,* 834 - 838. New York: IEEE. DOI: 10.1109/CSNT.2015.198.
[2] Wakerly, J. 1999. *Digital Design Principles and Practices*. 4th ed. New York: Prentice Hall.
[3] IEEE (Institute of Electrical and Electronic Engineers). 2009. *IEEE Standard VHDL Language Reference Manual.* New York: IEEE.
[4] IEEE (Institute of Electrical and Electronic Engineers). 2008. *754-2008 - IEEE Standard for Floating-Point Arithmetic. Revision of ANSI/IEEE Std 754-1985.* New York: IEEE.

[5] Dick, C. 2000. Choosing DSP or FPGA for your Application. January 03. http://www.hunteng.co.uk/info/fpga_dsp.htm (accessed 17 July, 2010).
[6] Dick, C. 1999. FPGAs: The High-End Alternative for DSP Applications. December 29. http://www.hunteng.co.uk/pdfs/tech/DSP1736FPGA.htm (accessed 17 July, 2010).
[7] Perry, D. 1998. *VHDL*. 3rd ed. New York: McGraw-Hill.
[8] Benini, L., and G. D. Micheli. 1996. "Automatic Synthesis of Low-Power Gated Clock Finite-State Machines". *IEEE Transactions on CAD* 15 (6): 630–643.
[9] Cheng, Fu-Chiung, Stephen H. Unger, Michael Theobald, and Wen-Chung Cho. 1997. "Delay-Insensitive Carry-Look Ahead Adders". *Proceedings of 10th International Proceedings VLSI Design, Conference, 4-7 January, 1997.* 37-63. New York: IEEE.
[10] Erle, Mark A, Brian J. Hickmann, and Michael J. Schulte. 2009. "Decimal Floating-Point Multiplication". *IEEE Transactions on Computers* 58 (7): 902–916.
[11] Koren, Israel. 2001. *Computer Arithmetic Algorithms*. 2nd ed. Natick, Massachusetts: A. K. Peters.