Implementation of a high Performance Electronic Circuit for Recognition, Segmentation and Verification of Digital Signals

Ashraf H. Yahia, Amr Radi, and Rehab H. Masoud

Physics Department, Faculty of Science, Ain Shams University Cairo, Egypt

ayahia@sci.asu.edu.eg., Amr.radi@cern.ch, Rehab.Hemdan@gmail.com

Abstract

A specific algorithm has been implemented using Field Programmable Gate Array FPGA devices for processing signals generated from a front-end electronic board of a specific nuclear detector. This algorithm is executed for processing (i.e. recognition, segmentation and verification) digital signals taken from the detector front-end electronic board. Such a processed signal is then analyzed by feeding it as an input to a data acquisition system DAQ. A set of VHDL hardware description language codes was also created to implement this algorithm using a regular FPGA hardware device. The simulation results for the processed signal using an ISim simulator are also presented.

Keywords: *FPGA, VHDL, CRC, ISim.*

1. Introduction

Field-Programmable-Gate-Arrays (FPGAs) are hardware in-system programmable devices whose function is not fixed. Specifically, the main advantages of algorithm implementations using FPGAs are cost efficiency, high data throughput, architecture efficiency and ability to modify and update the algorithm even dynamically. FPGA allows overcoming the main drawbacks of the traditional ASIC implementation that is lack of flexibility and high development costs [1].

FPGAs are used for a wide range of applications, e.g. network communication [2], video communication and processing [3, 4] and cryptographic applications [5]. It has been shown that FPGAs are suitable for the implementation of soft computing techniques like Neural Networks [6–8] and Genetic Algorithms [9–11].

Moreover, nowadays the Field-Programmable-Gate-Arrays are big enough to fit a whole digital system in a single device. Those System-on-a-Chips (SoCs) are designed using the core-based approach, interconnecting pre-designed hardware modules (IP cores) using standard on-chip buses. This design flow is valid for ASIC and FPGA design [1]. In this paper, we adapt an algorithm to FPGA to use in a nuclear detector system as shown in Fig. 1: a nuclear detector, a readout system and DAQ As shown in Fig. 1, the detector converts the energy deposited/induced by a particle (or photon) to an electrical signal.

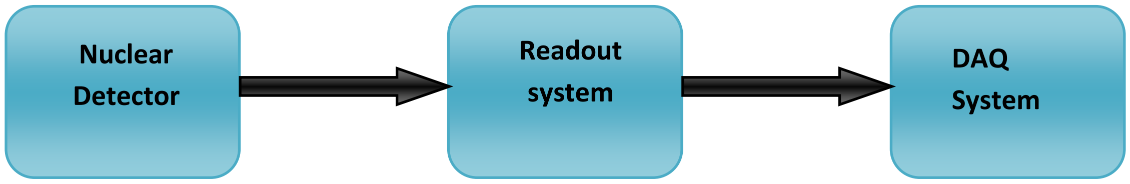

Fig. 1: The nuclear detector system

This analog signal is fed to the readout board which basically contains a chip. The chip has two main functions; the first (Trigger) is to provide programmable "fast OR" information based on the region of the sensor hit. The second function (Tracking) is for providing precise spatial hit information for a given triggered event. The digital data exit the chip serially through an output called "Data Out" with spatial format. Our goal is to recognize, segment, verify this data and convert it to parallel data, Figure 2 to become input to DAQ system using FPGA technology and VHDL as hardware description language.

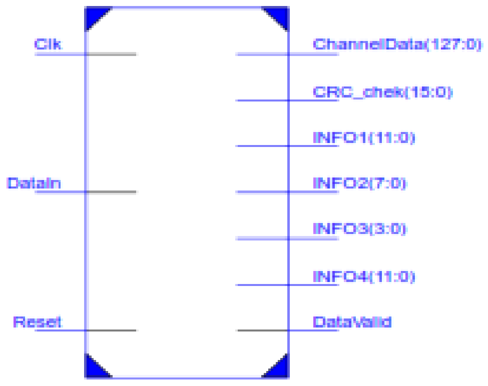

Fig. 2: The top module

2. Design

The front-end electronic board chip serially produces the data fields in the form given in Fig.3. Each data field consists of:

1- INFO1, INFO2, INFO3 and INFO4 with headers "1010", "1100" and "1110", form information about each event.

2- 128 bit event data.

3- A 16 bit cyclic redundancy check CRC. The CRC is a popular method for detecting errors during data transmission in computer-based applications. It consists of code, at the end of a data packet, a special binary word, normally with 16 or 32 bits, referred to as CRC *value*, obtained from calculations made over the whole data block to be transmitted. The receiver performs the same calculations for the received data and then compares the resulting CRC value against the received one [13].

Each data field is separated by two clock periods ,called "IDLE". From Fig.3 the total time for the readout of one package is 192 clocks for the data field ; plus 2 clocks for the IDLE period, this gives a total of 194 clock periods (4.85 μ s). The VHDL code design based on Finite State Machine (FSM) is used to control the processing on each data field of input signal.

The FSM is a special modeling technique for sequential logic circuits. Such a model is substantially helpful in the design of certain types of systems, particularly those whose tasks form a well-defined sequence [14]. State machines are usually modelled using a case statement in a process. The state information is stored in a signal. The multiple branches of the case statement contain the behaviour for each state [15].

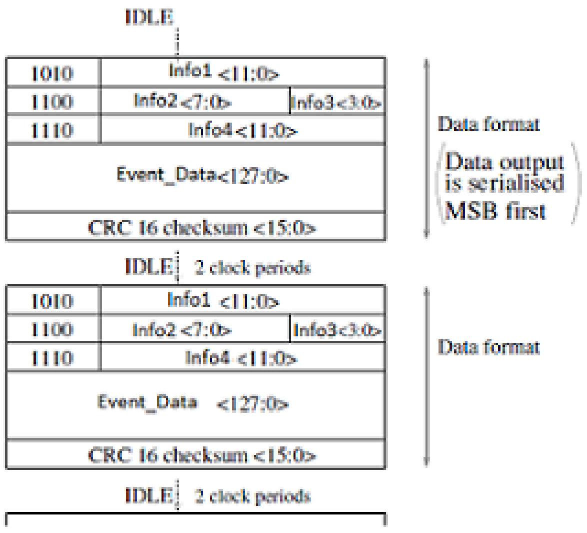

Fig. 3: Data Format

The present top-down design mainly consists of a **Top Module** , and a component, called **"CRC".** The Top Module is composed of a state machine with four states, as shown in Fig.4, namely:

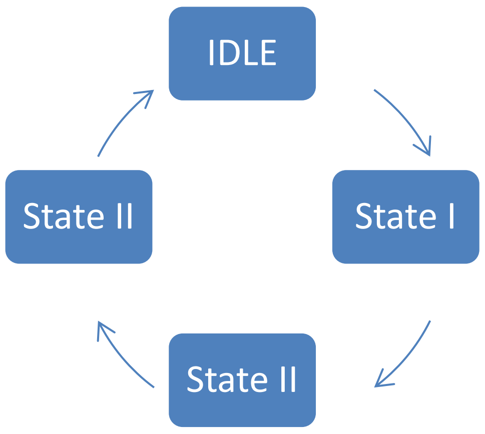

Fig.4: State machine of Top Module

Idle state is to initialize all signals and counters either when reset signal is high or at the end of detection of one data field ,Fig.5.

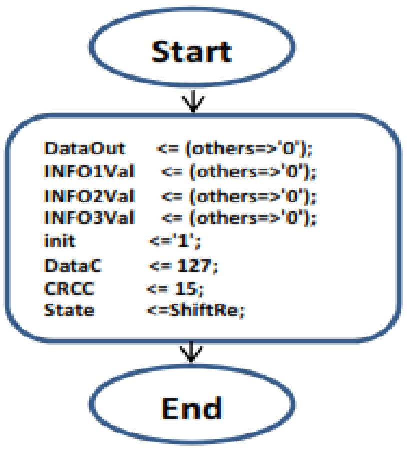
Start
DataOut <= (others=>'0');
INFO1Val <= (others=>'0');
INFO2Val <= (others=>'0');
INFO3Val <= (others=>'0');
init <='1';
DataC <= 127;
CRCC <= 15;
State <=ShiftRe;
End

Fig.5: Idle State flowchart.

1. **State I** collects INFO1, INFO2 ,INFO3 and INFO4 as well as their headers serially, one clock at time in shift register from the DataIn. To ensure the validly of the data field the **header position test** and **CRC test** are executed. According to **the header position test** the INFO1,INFO2,INFO3 and INFO4 data are only collected and registered when the headers are in their specified positions in shift register ,Fig.6.

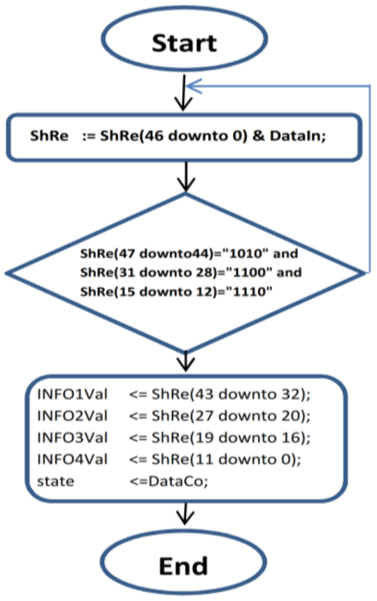
Start
ShRe := ShRe(46 downto 0) & DataIn;
ShRe(47 downto44)="1010" and
ShRe(31 downto 28)="1100" and
ShRe(15 downto 12)="1110"
INFO1Val <= ShRe(43 downto 32);
INFO2Val <= ShRe(27 downto 20);
INFO3Val <= ShRe(19 downto 16);
INFO4Val <= ShRe(11 downto 0);
state <=DataCo;
End

Fig.6: State I flowchart

2. **State II** where the event data are serially collected from the input data and saved in register. The CRC value of the stored data is calculated from the header of INFO1 up to the end of event data. This is composed of 176 bits divided into 11 signal codes, each having a 16 bit length. The codes are sent sequentially as signals to the CRC component. The output signal of this component is sorted and registered in “CRCTest” , as in Fig. 7.

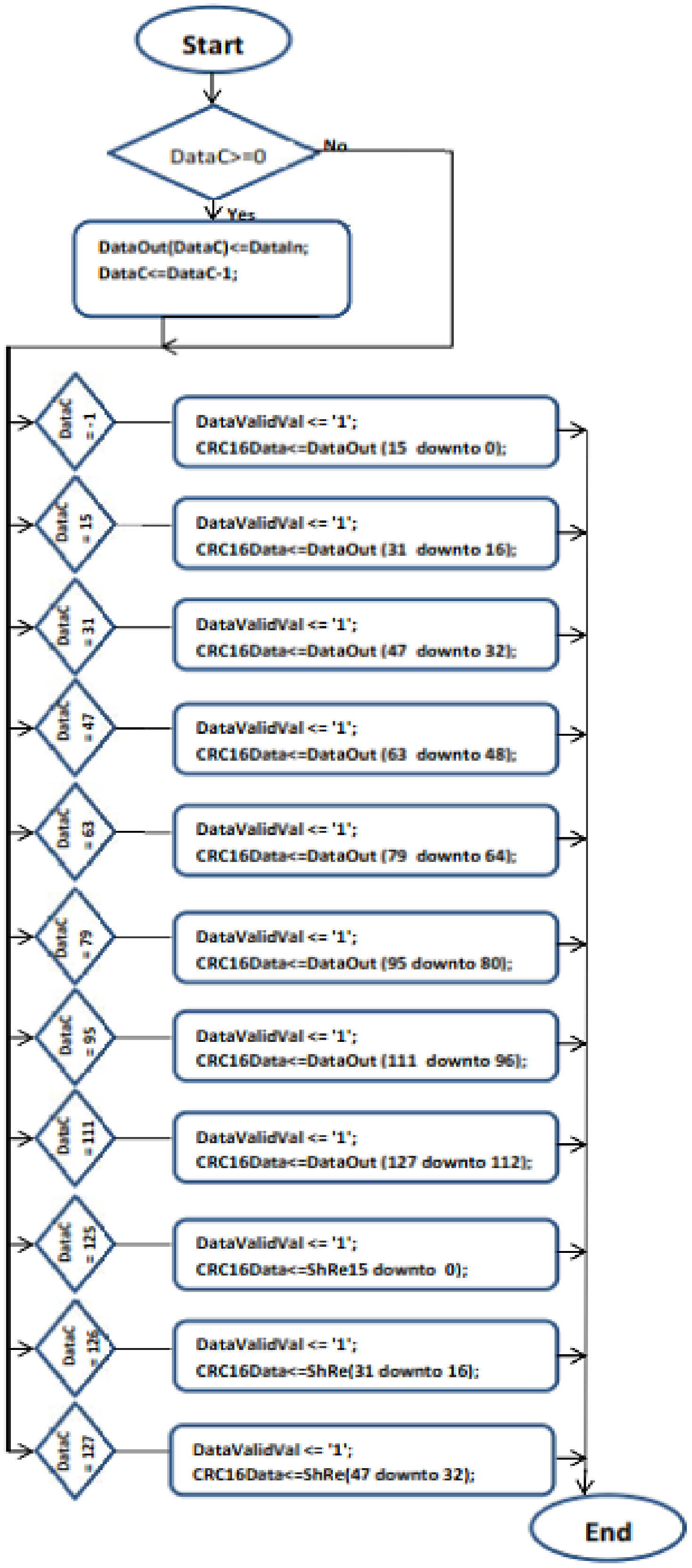

Fig.7: State II flowchart.

3. **State III** is in charge of gathering CRC of incoming data into another register, denoted as “CRCVal”. Upon completing this task, the CRCVal is compared with CRCTest (from **State II)** . The DataValid signal is “1” when the two values are identical. Finally, all saved data in registers are assigned to output ports and transferred as an input to DAQ system, Fig.8.

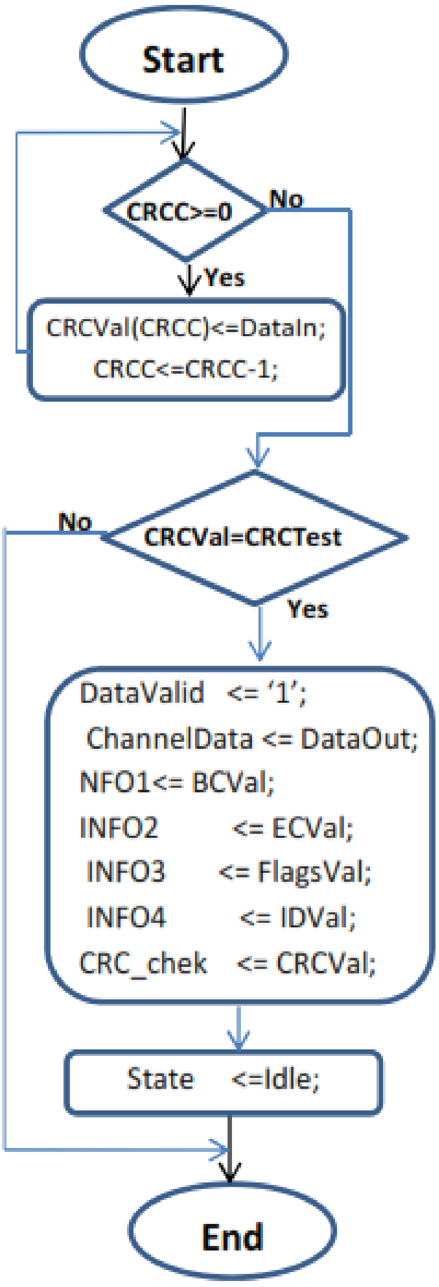

Fig.8: State II flowchart.

3. Verification and Results

After the development of our code it has been executed /simulated in a host computer to verify the correctness of the circuit operation and *synthesized* to a physical device. Simulation is usually performed within the same HDL framework. We created a special program, known as a *testbench*, to mimic a physical lab bench [16]. The sketch of the module testbench program is shown in Fig.9. The UUT block is the unit under test, the test vector generator block generates testing input patterns, and the monitor block examines the output responses. To simulate the model we used ISim which is a Hardware Description Language (HDL) simulator that lets us perform functional (behavioral) and timing simulations for VHDL, Verilog, and mixed-langue designs.The ISim environment comprises the following key elements:

1. Vhpcomp (VHDL) and vlogcomp (Verilog) parsers
2. Fuse (HDL elaborator and linker) command
3. Simulation executable
4. ISim Graphical User Interface (GUI) [17]

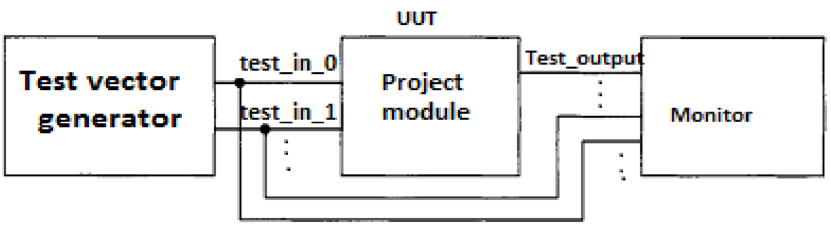

Fig. 9: Top module testbench block diagram.

In this work ISim GUI has been used to time simulate module testbench code which is written with VHDL .In this testbench virtual inputs are created (clock, reset, data_in) and the outputs are monitored Figure 10 illustrates the obtained results from **State I** where INFO1, INFO2, INFO3 and INFO4 had changed from "0" to their expected values .

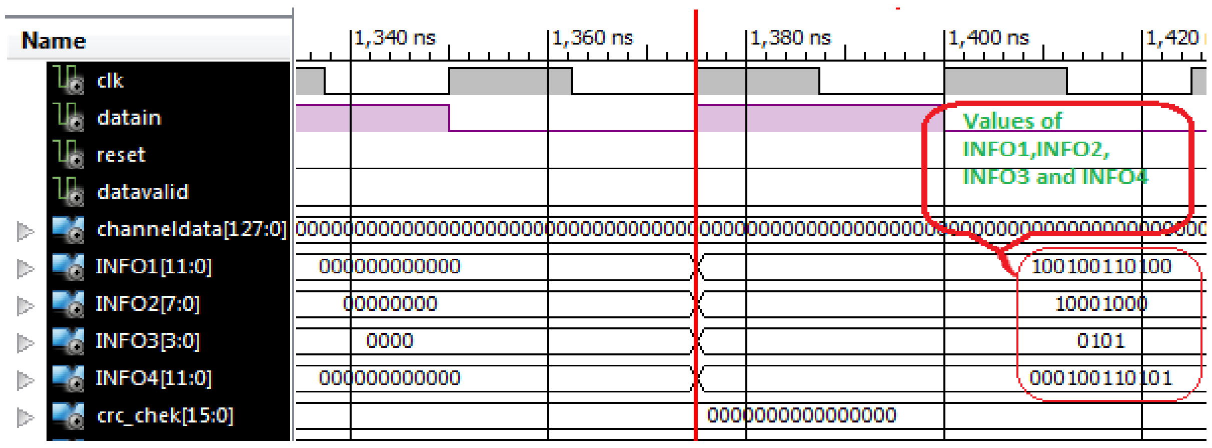

Fig.10: Simulation of State I

One has to note the collection of Data event and calculation of CRC value as expected from **State II,** Fig.11**.**

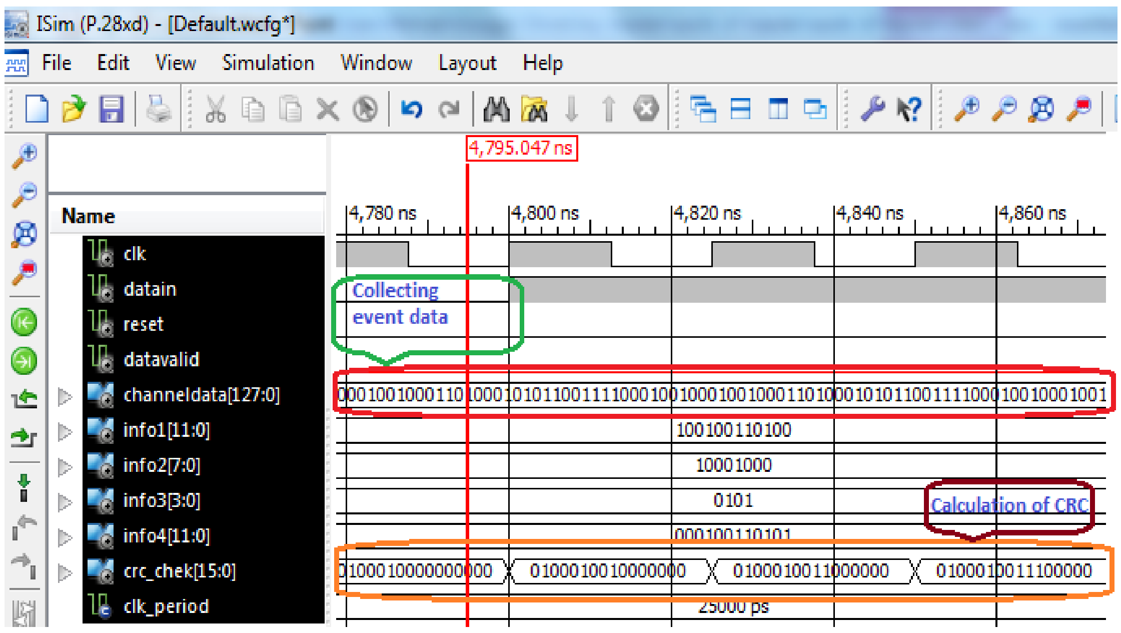

Fig.11: Simulation of State II

On comparing CRC test with that from the front-end board (CRC value), one notices the data valid signal to become "1" indicating an ON value, and INFO1, INFO2, INFO3, INFO4 and CRC to be fed to DAQ system.

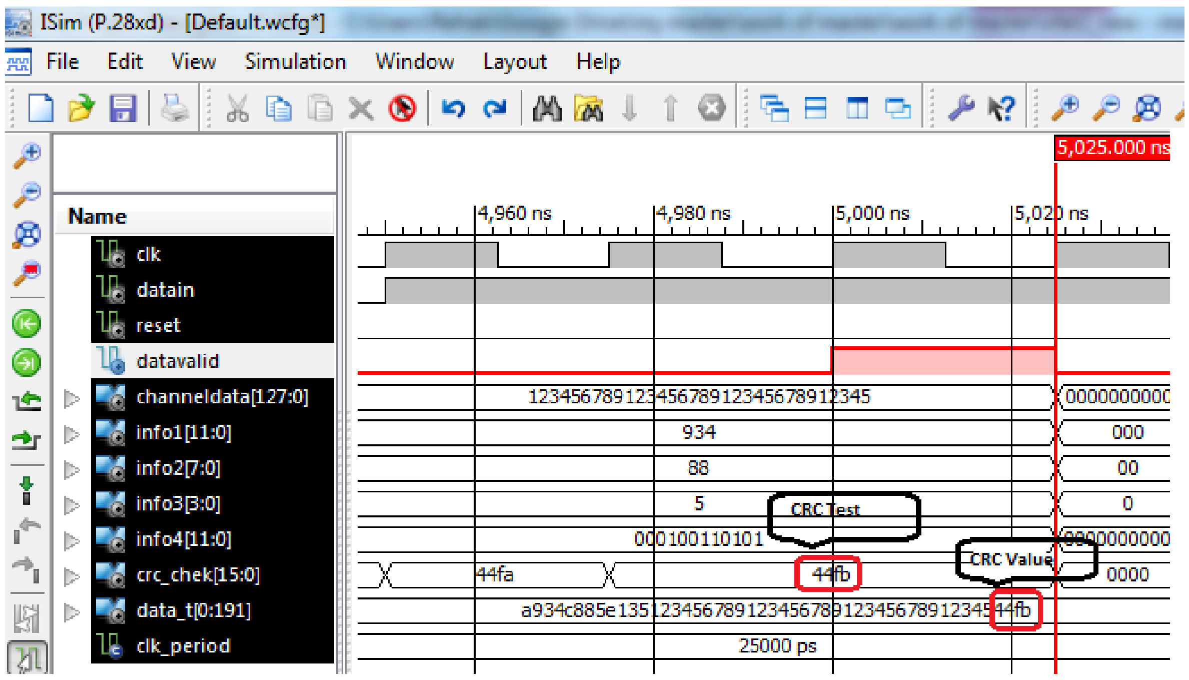

Fig.12: Simulation of State III

4. Conclusion

An FPGA chip has been designed for processing a digital signal taken from the front-end electronic board of a nuclear detector. The VHDL written algorithm /code allows signal processing, i.e. recognition, segmentation and verification. A generated digital signal is used to simulate those from detector .Reasonable results are obtained showing the feasibility of used technique.

References

[1] A. Astarloa, " FPGA technology for multi-axis control systems", Elsevier, Mechatronics. vol. 19 2009, 258–268.

[2] O. Cheung, P. Leong, "Implementation of an FPGA based accelerator for virtual private networks", in: IEEE International Conference on Field Programmable Technology (FPT), Hong Kong, 2002, pp. 34–43.

[3] J. Gause, P. Cheung, W. Luk, "Static and dynamic reconfigurable designs for a 2D shape-adaptive DCT", in: R. Hartenstein, H. Grüunbacher (Eds.), Field-Programmable Logic and Applications, FPL, Springer-Verlag , 2000,pp. 96–105.

[4] J. Villasenor, C. Jones, B. Schoner, "Video communications using rapidly reconfigurable hardware", IEEE Trans. Circuits Syst. Video Technol. (1995) 565–567.

[5] A. Hämäläinen, M. Tommiska, J. Skyttä, "6.78 gigabits per second implementation of the IDEA cryptographic algorithm, in: M. Glesner, P. Zipf, M. Renovell (Eds.), Field Programmable Logic and Applications. Reconfigurable Computing Is Going Mainstream", LNCS 2438, Springer-Verlag, 2002, pp. 760–769.

[6] S. Bade, B. Hutchings, "FPGA based stochastic neural network implementation", in: Proceedings of the IEEE Workshop on FPGAs for Custom Computing Machines, 1994, pp. 189–198.

[7] P. Lysaght, J. Stockwood, J. Law, D. Girma," Artificial neural network implementation on a fine-grained fpga", in:Field-Programmable Logic, 1994, pp. 421–431.

[8] R. Gadea, J. Cerd, F. Ballester, A. Mochol, "Artificial neural network implementation on a single fpga of a pipelined on-line back propagation", in: Proceedings of the 13th Conference on International Symposium on System Synthesis,2000, pp. 225–230.

[9] P. Graham, B. Nelson, "Genetic algorithms in software and in hardware—a performance analysis of workstation and custom computing machine implementations", in: IEEE Symposium on FPGAs for Custom Computing Machines, 1996, pp. 216–225.

[10] I.M. Bland, G.M. Megson, "The systolic array genetic algorithm, an example of systolic arrays as a reconfigurable design methodology", in: IEEE Symposium on FPGAs for Custom Computing Machines, 1998, pp. 260–261.

[11] C. Aporntewan, P. Chongstitvatana, "Hardware implementation of the compact genetic algorithm", in: Proceedings of the 2001 IEEE Congress on Evolutionary Computation, Seoul, South Korea, 2001, pp. 624–629.

[12] Chang H et al. Surviving the SOC revolution, Kluwer Academic Publishers, Massachusetts, USA; 1999.

[13] Volnei A. Pedroni,” Digital Electronics and Design with VHDL”, Morgan Kaufmann,2008.

[14] Volnei A. Pedroni, “Circuit design with VHDL”, Cambridge, Massachusetts London, England, 2004.

[15] J. Bhasker, “A VHDL Primer”, 3rd Edition, Prentice Hall PTR, (1999).

[16] Pong P. Chu,” FPGA Prototyping By Verilog Examples: Xilinx Spartan-3 Version”, John Wiley 2008.

[17] Xilinx ,“ISE Simulator (ISim) In-Depth Tutorial”, Xilinx,2009.p 1-2.