RESEARCH ARTICLE OPEN ACCESS

Design and Simulation of Mathematical Models- An DSP and Digital Communication Based Approach

Miss. Shruti R. Tambakhe*, Prof. Ajay P. Thakare**
*(PG Student (EXTC), Dept. Of EXTC, Sipna College of Engineering, Amravati, Maharashtra, India)
** (HOD, Dept. Of EXTC Sipna College of Engineering, Amravati, Maharashtra, India)

ABSTRACT
A methodology for implementing DSP based or communication applications on a field programmable gate arrays (FPGA) using Xilinx System Generator (XSG) for Matlab. The DPSK system and FFT are simulated using Matlab/ Simulink environment and System Generator, a tool from Xilinx used for FPGA design as well as implemented on Spartan 3E Starter Kit boards. We proposed the concept of simulation of mathematical model on mixed HDL-Simulink using Xilinx system generator. Many applications that are DSP based or certain communication application require mathematical modelling for their easy understanding and analysis. Due to its complexity Pure HDL is unable to simulate. Also it terms to be costly and time consuming process.
The implementation of FFT algorithms that can compute fourier transform of varied signals in real time for frequency analysis of signals on FPGAs (Spartan3E). With large demand for high dynamic range for applications, floating point implementation is used as fixed point implementation becomes increasingly expensive. The DPSK model are first board behaves as a modulator and the second as a demodulator. The modulator and demodulator algorithms have been implemented on FPGA using the VHDL language on Xilinx ISE.

Keywords- FFT, DPSK, FPGA(Field Programmable Gate Arrays), Hardware-Software Co-simulation, MATLAB, XSG(Xilinx System Generator).

I. INTRODUCTION

This paper provides a new approach towards the design and modeling of based complex mathematical model using mixed HDL platform of simulink and Xilinx form of the design architecture. The Hardware Description Languages [1], [3] (HDLs) have been developed for several years in this field. The HDLs and automated tools based on them have given new abilities to designers, however now they seem to be not sufficient. Now we are looking for tools that deal with models and descriptions on higher levels of abstraction and we would like to describe the entire system as a one piece i.e. without its initial, manual decomposition into the hardware and software. The terms of system level design, co-design and co-simulation have been well known for several years. The complexity of the systems has a strong impact on the models that are expected to reflect the functionality and timings of the real devices.

The programming of the FPGA is done using a logic circuit diagram or a source code using a Hardware Description Language (HDL) to specify how the chip should work. FPGA or Field Programmable Gate Arrays can be programmed or configured by the user or designer after manufacturing and during implementation. Hence they are otherwise known as On-Site programmable. Unlike a Programmable Array Logic (PAL) or other programmable device, their structure is similar to that of a gate-array or an ASIC. The programmable logic blocks are called configurable logic blocks and reconfigurable interconnects are called switch boxes.
Matlab is a high-level technical computing language and interactive environment for algorithm development, data visualization, data analysis, and numeric computation. In addition to the intellectual property functions provided in Matlab, the software packet is uniquely adept with vector and array based waveform data at the core of algorithms, which is suitable for applications such as image and video processing. Matlab-Simulink is an environment for multidomain simulation and Model-Based Design for dynamic and embedded systems. Matlab-Simulink is used in this application as the high level development tool in the design process. Xilinx System Generator, is a system-level modeling tool from Xilinx that facilitates FPGA hardware design. System Generator the use of simulates automatically launching an HDL simulator, generating additional HDL as needed (analogous to an HDL testbench), compiling HDL, scheduling simulation events, and handling the exchange of data between the Simulink and the HDL simulator. This is called HDL co-simulation. System Generator provides a generic interface that uses JTAG and a Xilinx programming cable (e.g., Parallel

Cable IV or Platform Cable USB) to communicate with FPGA hardware. The model with the JTAG-based hardware co-simulation block implemented on Spartan 3E platform.

In this paper two application are used one is FFT and another is DPSK system. In FFT is increased demand and advancements in product design in the field of communication, multimedia, security and safety equipment and other industrial and scientific products have created the need for high volume, low cost, multifunction, DSP based frequency analyzers that can use Fast Fourier Transform (FFTs)for their signal processing or data manipulation. The paper deals with implementation of FFT algorithms that can compute fourier transform of varied signals in real time for frequency analysis of signals on FPGAs (Spartan3E). With large demand for high dynamic range for applications, floating point implementation is used as fixed point implementation becomes increasingly expensive. The inherent massive parallelism of FPGAs allows these solutions to be competitive to software equivalent. In communication another is DPSK system, the DPSK modulation and demodulation represents an important modulation technique in terms of signal power. Both, the modulator and demodulator, have been designed and simulated have been implemented on FPGA using the VHDL language on Xilinx ISE.

II. SYSTEM ARCHITECTURE

Matlab-Simulink is an environment for multidomain simulation and Model-Based Design for dynamic and embedded systems. Matlab-Simulink is used in this application as the high level development tool in the design process. Xilinx System Generator, is a system-level modeling tool from Xilinx that facilitates FPGA hardware design. It extends Simulink in many ways to provide a modelling environment well suited for hardware design. The software automatically converts the high level system DSP block diagram to RTL. The result can be synthesized to Xilinx FPGA technology using ISE tools. All of the downstream FPGA implementation steps including synthesis and place and route are automatically performed to generate an FPGA programming file. It also provides a system integration platform for the design of DSP FPGAs that allows the RTL, Simulink, MATLAB and C/C++ components of a DSP system to come together in a single simulation and implementation environment. System Generator supports a box block that allows RTL to be imported into Simulink and cosimulated with either ModelSim or Xilinx ISE Simulator.

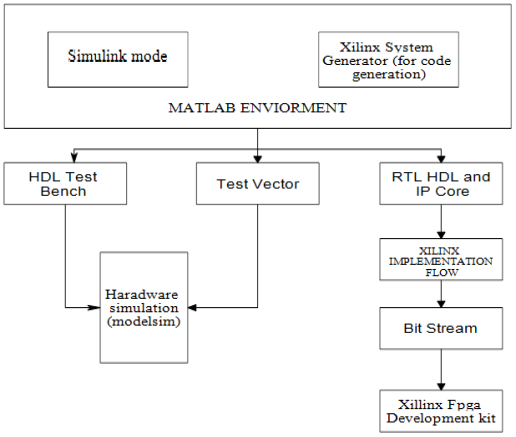

Fig 1: Block Diagram of System Architecture

System Generator the use of simulates automatically launching an HDL simulator, generating additional HDL as needed (analogous to an HDL testbench), compiling HDL, scheduling simulation events, and handling the exchange of data between the Simulink and the HDL simulator. This is called HDL co-simulation. System Generator provides hardware co-simulation, making it possible to incorporate a design running in an FPGA directly into a Simulink simulation. "Hardware Co-Simulation" compilation targets automatically create a bit-stream. When the system design is simulated in Simulink, results for the compiled portion are calculated in actual FPGA hardware, often resulting in significantly faster simulation times while verifying the functional correctness of the hardware. System Generator for DSP supports Ethernet as well as JTAG communication between a hardware platform and Simulink.

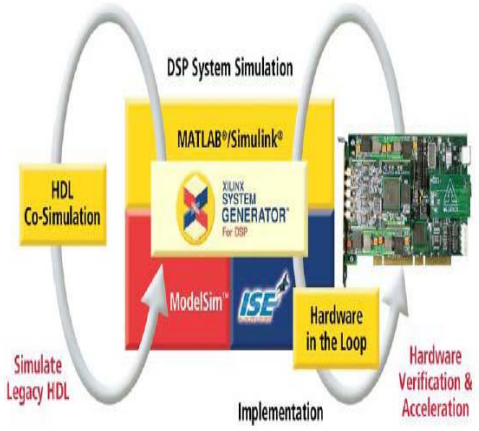

Fig 2: FPGA based Hardware-Software (HW-SW) co- simulation Environment

System Generator provides a generic interface that uses JTAG and a Xilinx programming cable (e.g., Parallel Cable IV or Platform Cable USB) to communicate with FPGA hardware. The model

with the JTAG-based hardware co-simulation block implemented on Spartan 3E platform. Point-to-point Ethernet co-simulation provides a straightforward high-performance co-simulation environment using a direct, point-to-point Ethernet connection between a PC and FPGA platform. The target FPGA chip is Xilinx Spartan 3E DSP platform. The optimization setting is for maximum clock speed. Xilinx system generator has a unique hardware in the loop co-simulation feature that allows designers to greatly accelerate simulation while simultaneously verifying the design in hardware.

Sometimes it is important to add one or more existing HDL modules to a System Generator design. The System Generator allows VHDL, Verilog, to be brought into a design. When System Generator compiles, it automatically wires the imported module and associated files into the surrounding netlist. Xilinx system generator is a very useful tool for developing computer vision algorithms. It could be described as a timely, advantageous option for developing in a much more comfortable way than that permitted by VHDL or Verilog hardware description languages (HDLs).

III. PROPOSED IMPLEMENTATION

This paper two applications that are DSP based and communication application require mathematical modelling for their easy understanding and analysis namely as

1) FFT (Fast Fourier Transform)
2) DPSK (Differential Phase Shift Keying)

1) FFT

FFT (Fast Fourier Transform). The FFT is a faster version of the Discrete Fourier Transform (DFT). The FFT utilizes some clever algorithms to do the same thing as the DTF, but in much less time.

The DFT is extremely important in the area of frequency (spectrum) analysis because it takes a discrete signal in the time domain and transforms that signal into its discrete frequency domain representation. Without a discrete-time to discrete-frequency transform we would not be able to compute the Fourier transform with a microprocessor or DSP based system.

It is the speed and discrete nature of the FFT that allows us to analyze a signal's spectrum with Matlab or in real-time on the SR770.

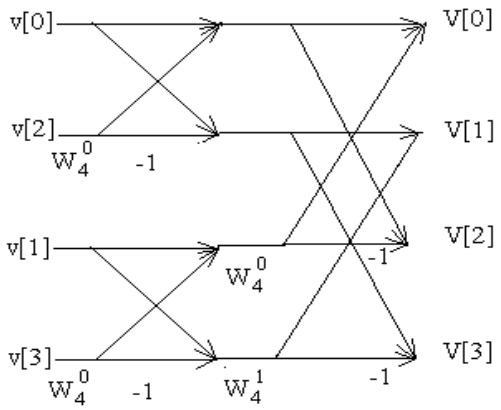

Fig 3: 4-point FFT calculation

This diagram is the essence of the FFT algorithm. The main trick is that you don't calculate each component of the Fourier transform separately. That would involve unnecessary repetition of a substantial number of calculations. Instead, you do your calculations in stages. At each stage you start with N (in general complex) numbers and "butterfly" them to obtain a new set of N complex numbers. Those numbers, in turn, become the input for the next stage. The calculation of a 4-point FFT involves two stages. The input of the first stage are the 4 original samples. The output of the second stage are the 4 components of the Fourier transform. Notice that each stage involves N/2 complex multiplications (or N real multiplications), N/2 sign inversions (multiplication by -1), and N complex additions. So each stage can be done in O(N) time. The number of stages is $\log_2 N$ (which, since N is a power of 2, is the exponent m in $N = 2^m$). Altogether, the FFT requires on the order of O(N logN) calculations.

Moreover, the calculations can be done in-place, using a single buffer of N complex numbers. The trick is to initialize this buffer with appropriately scrambled samples. For N=4, the order of samples is v[0], v[2], v[1], v[3]. In general, according to our basic identity, we first divide the samples into two groups, even ones and odd ones. So this is how the FFT algorithm works (more precisely, this is the decimation-in-time in-place FFT algorithm).

1. Select N that is a power of two. You'll be calculating an N-point FFT.
2. Gather your samples into a buffer of size N
3. Sort the samples in bit-reversed order and put them in a complex N-point buffer (set the imaginary parts to zero)
4. Apply the first stage butterfly using adjacent pairs of numbers in the buffer
5. Apply the second stage butterfly using pairs that are separated by 2

6. Apply the third stage butterfly using pairs that are separated by 4
7. Continue butterflying the numbers in your buffer until you get to separation of N/2
8. The buffer will contain the Fourier transform.

2) DPSK

Differential phase shift keying (DPSK), a common form of phase modulation conveys data by changing the phase of carrier wave. In Phase shift keying, High state contains only one cycle but DPSK contains one and half cycle.

- The differential phase shift keying can be treated as the non-coherent version of PSK.
- It combines two basic operations, namely
  1) The differential encoding and
  2) Phase Shift Keying
- Hence the name differential phase shift keying (DPSK).

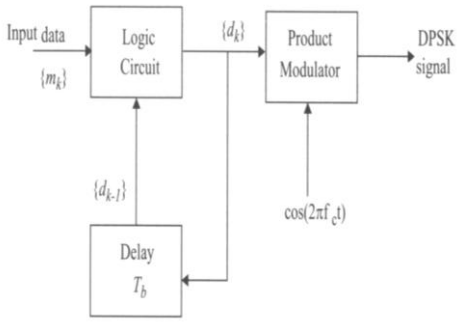

Fig 4: Block Diagram of DPSK Transmitter

The differential phase shift keying (DPSK) is a modification of BPSK. Figure shows the block diagram of the DPSK transmitter. The operation of DPSK transmitter is m_k represents the data stream which is to be transmitted. The output d_k is delayed by one bit period T_b and applied to the input data. Depending on the values of m_k and d_{k-1}, the input data produces the output sequence d_k. The multiplied with the carrier signal to produce DPSK signal.

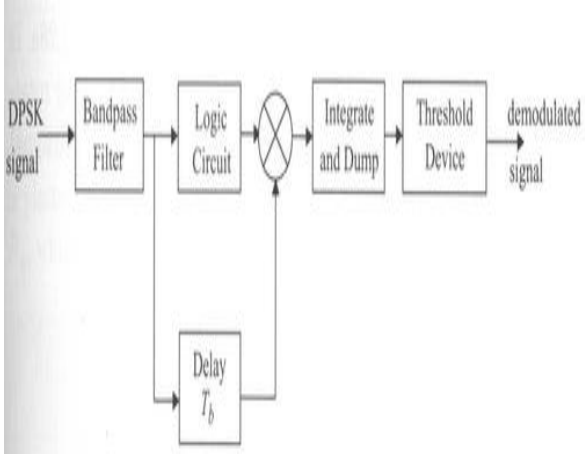

Fig 5: Block Diagram of DPSK Receiver

As shown above the block diagram of a DPSK receiver. The received DPSK signal is applied through a bandpass filter. The another input from the delay unit which introduces a delay of one bit duration (T_b). The output of demodulated signal is which compares threshold device.

IV. RESULT

In this implementation the FFT System in MATLAB simulink fig 6, and same system used in Xilinx System Generator fig 7. The FFT System implemented on the Spartan 3E Starter Kit board has the same principle as the implementation in System Generator fig 8.

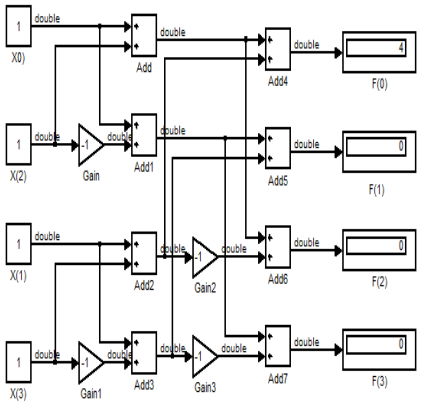

Fig 6: FFT Model of MATLAB Simulink

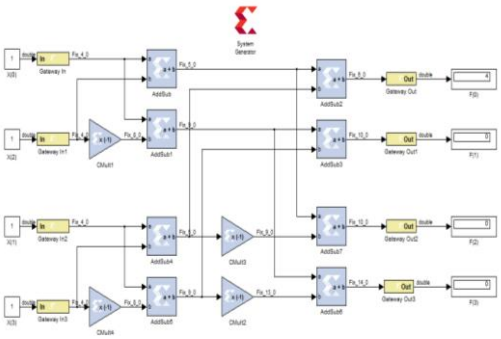

Fig 7: FFT Model of System Generator

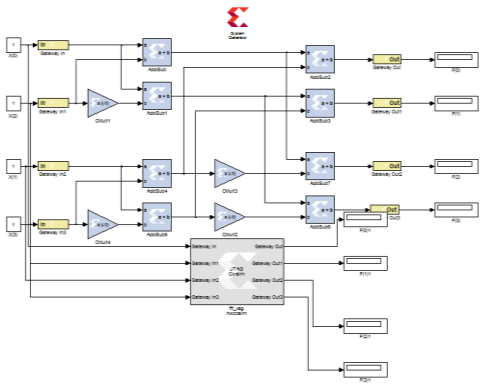

Fig 8: FFT Model of System Generator with JTAG Cosimulation

After implementing the DPSK System made of a Modulator and demodulator on the two Spartan 3E Starter Kit boards,a high performance.

Fig 11 illustrates the signal in the modulator. The first one represents the modulating signal generated by the LFSR, the second one, the carrier and the third, the modulated signal.

In fig.14, the modulated, the modulating and the demodulated signals are shown.

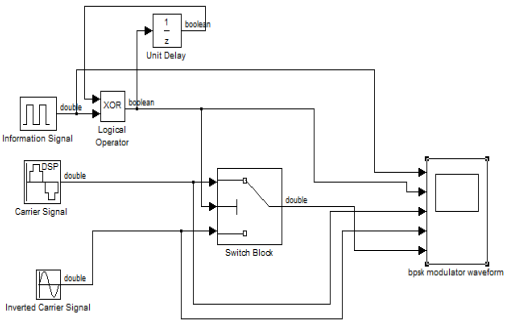

Fig 9: DPSK Modulator in MATLAB-Simulink

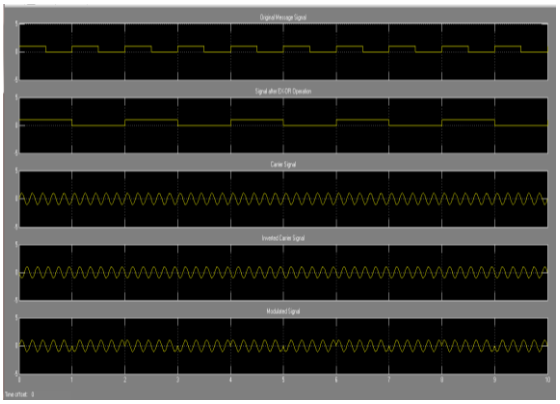

Fig 10: Waveform of DPSK Modulator

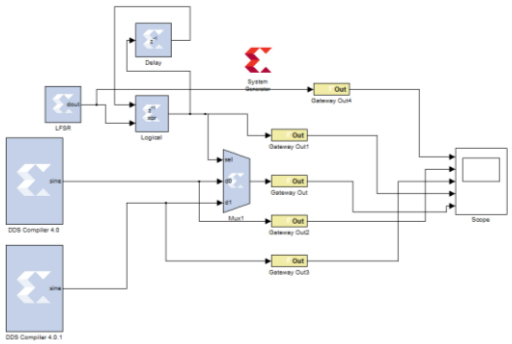

Fig 11: DPSK Modulator of System Generator

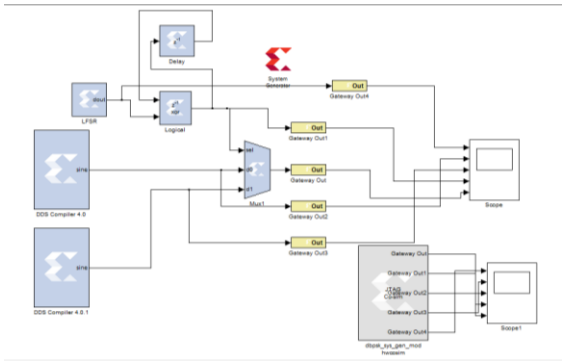

Fig 12: DPSK of system generator with JTAG

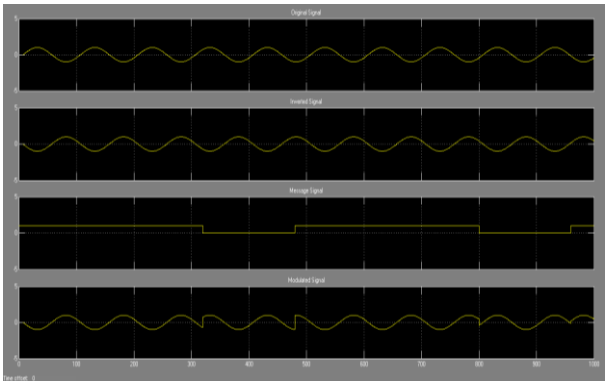

Fig 13: Waveform of DPSK Modulator system generator with JTAG Cosimulation

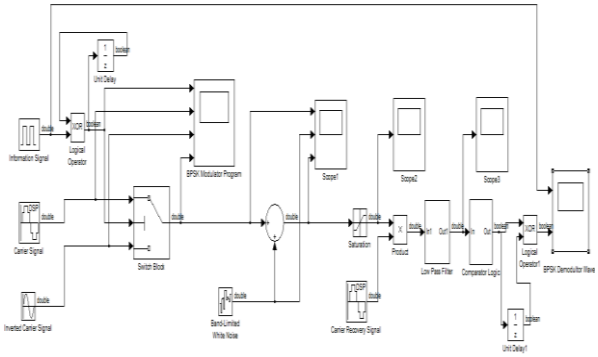

Fig 14: DPSK Mod-Demod in MATLAB-Simulink

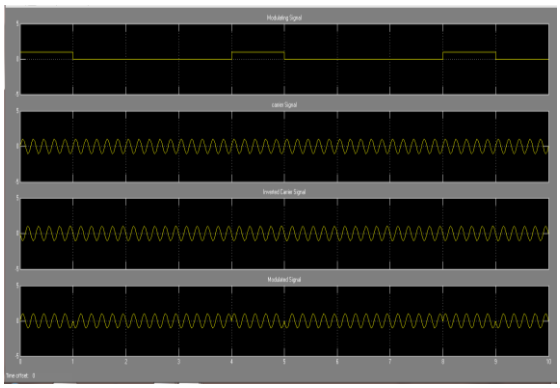

Fig 15: Waveform of DPSK Modulator

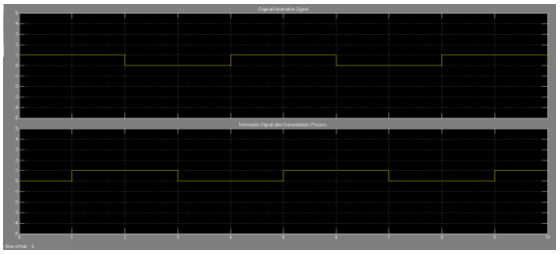

Fig 16: Waveform of DPSK Demodulator

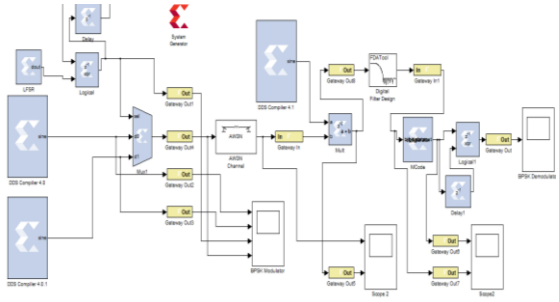

Fig 17: DPSK Mod-Demod of System Generator

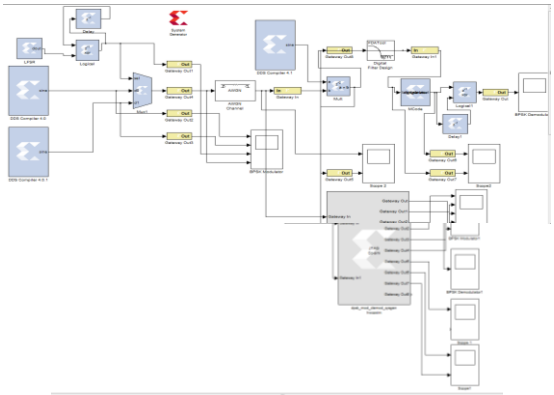

Fig 18: DPSK Mod-Demod of System Generator with JTAG Cosimulation

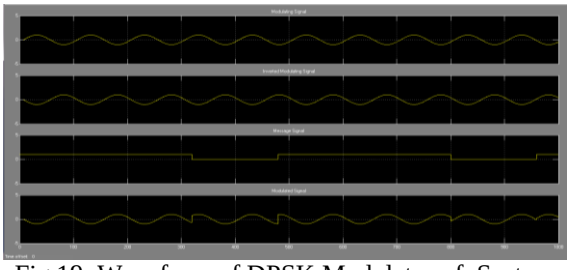

Fig 19: Waveform of DPSK Modulator of System Generator with JTAG Cosimulation

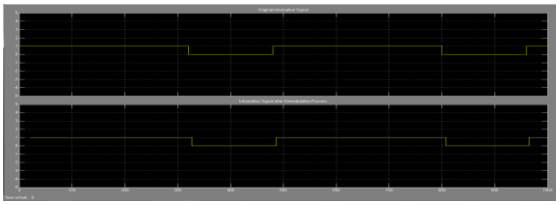

Fig 20: Waveform of DPSK Demodulator of System Generator with JTAG Cosimulation

V. CONCLUSION

In this paper we present the system to simulate complex mathematical models. The proposed methodology has been tested with Xilinx System Generator (XSG) and MATLAB environments. The basic concepts of creating a design using System Generator within the model-based design flow provided through Simulink. First of the design and simulate the model in a MATLAB Simulink. Then the design Xilinx block in system generator and we take the Xilinx executable specification through the full implementation flow. And Explore the Hardware Architectures on Spartan 3E FPGA to achieve the best performance. Finally in the program run FPGA.

In this paper have been implemented the DPSK system and FFT are simulated using Matlab/ Simulink environment and System Generator, a tool from Xilinx used for FPGA design as well as implemented on Spartan 3E Starter Kit boards.

We proposed a implementation of the DPSK System (Modulator and Demodulator) in the Matlab/Simulink environment. Then, we made a proposal of a DPSK System in System Generator. Both, the modulating signal and the carrier are generated internal, the modulating signal by a LFSR and the carrier by a DDS Compiler. The modulated signal is obtained at the output of a mux block and, then,passed through a communication channel where noise is added. In the demodulator, the carrier is recovered due to another DDS compiler and then multiplied with the modulated signal affected by noise. The obtained signal is then added with all the multiplied samples from the carrier in a period. The operation takes place in the accumulator. Once we have a result, it is compared with a decision threshold. If the compared signal is positive, the demodulator take the decision that '1' was transmitted, otherwise, '0'. The BPSK System implemented on the Spartan 3E Starter Kit board has the same principle as the implementation in System Generator. Although System Generator has an option to generate the VHDL code.

REFERENCES

[1] Adam Milik and Andrzej Pulka "*Complex Mathematical Models Simulation On Mixed HDL-Simulink Platform*", in proceeding of IEEE, Poland, 2008.

[2] S.O.Popescu, A.S.Gontean and G.Budura "*Simulation and Implementation of a BPSK Modulator on FPGA*", in Proceedings of the IEEE International Symposium on Applied Computational Intelligence and Informatics (SACI 2011), Romania, 2011, pp.459-463.

[3] S.O.Popescu, A.S.Gontean and G.Budura "*BPSK System on Spartan 3E FPGA*", in

proceedings of the IEEE International symposium on Applied Computational Intelligence and Informatics (SACI), Romania,2012.

[4] System Generator for DSP. Getting Started Guide. Xilinx, 2008.

[5] Spartan 3E FPGA Starter Kit Board. User Guide. Xilinx, 2011.

[6] Simulink: http://www.mathworks.com/products/simulink/

[7] Xilinx System Generator User's Guide, www.xilinx.com

[8] ISim Hardware Co-Simulation Tutorial: Accelerating Floating Point FFT

[9] Simulation by xilinx cooperation **UG817 (v14.1) April 24, 2012.**

[10] Wold, E. and Despain, A. 1984. Pipeline and parallel-pipeline fft processors for vlsi implementations. Computers, IEEE Transactions on C-33, 5 (may), 414 -426

[11] http://en.wikipedia.org/wiki/Fast_Fourier_transform.

[12] Winthrop W. Smith, Joanne M. Smith, *"Handbook of Real Time FFT",* IEEE Press, 1995.

[13] System Generator manuals "*System Generator for DSP: Performing Hardware-in-the- Loop with the Spartan-3E Starter Kit*".

[14] IEEE paper "*Fpga Implementation Of FFT Algorithms Using FloatingPoint Numbers*" by Hilal Kaptan, Ali Tangel, Suhap Sahin.

[15] H. J. Nussbaumer, *"Fast Fourier Transforms and Convolution Algorithms",* Springer: Berlin 1981

[16] Ramrez R. W., *"FFT fundamentals and Concepts",* Prentice Hall