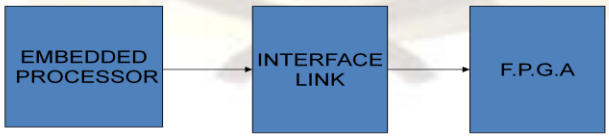
B.K.V.Prasad, Dr.Fazal Noorbasha, D.Naga Dilip Kumar, L.Veeraju / International Journal of Engineering Research and Applications (IJERA) ISSN: 2248-9622
www.ijera.com Vol. 2, Issue 3, May-Jun 2012, pp.2022-2032

ESTABLISHMENT OF SECURED COMMUNICATION

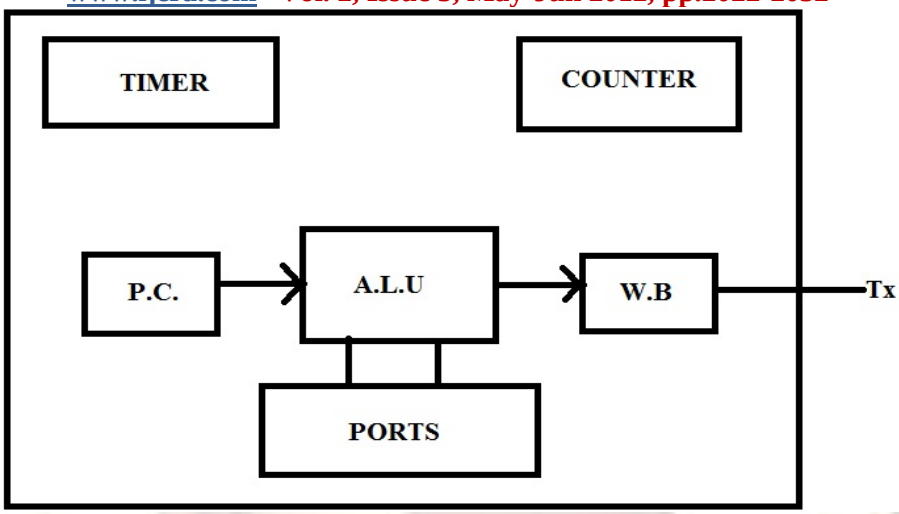
B.K.V.Prasad[1], Dr.FazalNoorbasha[2], *D. NagaDilipKumar*[3] and L.Veeraju[4]
[1,2]Dept. of ECE, K.L.University, Vijayawada, India
[3,4]M.TECH Student ,Department of ECE K.L.University,Guntur Dt.AP,India

ABSTRACT:
The main objective of this project is handling the large numbers of faults in F.P.G.A. configurable logic blocks. A key novel feature is the reuse of defective logic blocks to increase the number of effective spares and extend the mission life. To increase fault tolerance, we not only use nonfaulty parts of defective or partially faulty logic blocks, but we also use faulty parts of defective logic blocks in nonfaulty modes. By using and reusing faulty resources, our multilevel approach extends the number of tolerable faults beyond the number of currently available spare logic resources. A challenging aspect of developing applications that target the interfacing of the processor with the surrounding reconfigurable logic is main criteria in this project. O.C.P protocol interfaces the embedded processor with F.P.G.A. Testing all configurable logic blocks is under care of embedded processor. The embedded processor set the test mode for F.P.G.A, when it is needed it stops when it is in sleep mode.

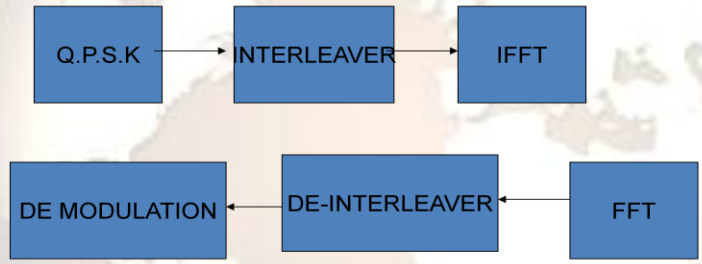
INTRODUCTION:
ADVANCES in silicon technologies continue to increase the number of transistors that can be integrated into a single device. Many chip manufacturers use this new integration potential to fabricate complete systems on a single silicon device. FPGA manufacturers are using this expanded capacity to implement embedded DSPs, multipliers, and memory blocks along with the reconfigurable fabric. At Xilinx, FPGA designers have developed new generations of FPGA architectures that contain a variety of embedded resources. the processor core is that most FPGAs contained within an embedded system require some level of interaction with an external processor. Moving this processor onto the FPGA chip eliminates bottlenecks associated with communicating through off-chip interfaces. These are replaced by on-chip interfaces that must provide efficient communication between the processor andthe reconfigurable resources. Since direct human intervention for maintenance and repair is impossible in such environments, fault-tolerant (FT) techniques resulting in graceful degradation must be used to achieve the desired mission life span even in the presence of faults.
Existing technique:
Only offline testing is available in FPGAs,there is no control over the testpatterns implies there is no accuracy while the circuit is in running position. Offline testing Deals with testing a system when it is not carrying out its normal functions (Test mode, Non-Real-Time error detection).There are two types of offline testing.
Structural : Execution based on the structure of the CUT(Explicit fault model - LFSR, ...).
Functional : Running based on functional description of CUT(Functional fault model - Diagnostic software)
PROPOSED BLOCKDIAGRAMS:



By using above block diagram we can develop a ONLINE BIST, Since there is an interaction of embedded processor.
Embedded Processor:



Interface :



OCP :

OCP PROTOCOL IS USED TO INTERFACE EMBEDED PROCESSOR AND FPGAThe **Open Core Protocol** (**OCP**) is an openly licensed, core-centric protocol intended to meet contemporary system level integration challenges. OCP defines a bus-independent, configurable and scalable interface for on-chip subsystem communications. OCP International Partnership (OCP-IP) now offers the 2.2 version specification that further extends capabilities in areas such as very high performance multithreading, synchronization primitives and single-request/multiple-data transactions. OCP data transfer models range from simple request-grant handshaking through pipelined request-response to complex out-of-order operations.

Legacy IP cores can be adapted to OCP, while new implementations may take full advantage of advanced features: designers select only those features and signals encompassing a core's specific data, control and test configuration. Core definition using OCP encapsulates a complete system integration description enabling core and test bench reuse without rework. Not only does OCP provide clear delineation of design responsibilities for core authors and System-on-Chip (SoC) integrators, but also institutes a key partitioning formalism for verification engineers and automation software.

MAPPING / MODULATION

Doble qpsk modulation is implemented for secured communication DVB-T specifies two level coding and modulation method which enables it to use hierarchical transmissions [1] [2]. In hierarchical transmission two MPEG-2 transport streams, referred to as Highand Low Priority streams, are transmitted in a combined signal. The HP stream is a low bit rate stream which can be received under poor or difficult channel conditions (low SNR) and the LP stream is a high bit rate stream which requires good communication channel (high SNR). In simulcast mode these two streams can carry the same programme in a low-bit-rate, rugged version and one high bit-rate, less rugged version. In multicast mode totally different programmes can be transmitted, one low-bit rate programme - receivable

by mobile receivers, and a high-bit rate programme - receivable by directed antennas. In hierarchical mode the receiver is only required to decode one of the two transport streams. The two modes differs by means of the channel coding chain, but they use the same inner interleaver. After the inner interleaver, the mapper is employed to convert every v-bits into one complex symbol, selected from a given signal constellation, corresponding to M-QAM digital modulation schemes, where M=2v. DVB-T specifies seven different signal constellations: QPSK (v = 2), uniform or non-uniform 16-QAM (v = 4) and 64-QAM (v = 6). The complex symbols after the mapper, are used to modulate orthogonally spaced carriers. For hierarchical transmission both uniform and non-uniform modulation can be used, and the convolution encoder can have different code rates for HP and LP streams. The non-uniform modulation has unequal distance between groups of constellation points, so that a signal carried by the QPSK constellation can be received at a lower SNR than the full constellation of 16- or 64- QAM. An important parameter, when non-uniform modulation is used, is the modulation factor □ , which can take on values 1, 2 or 4. When □□ increases the HP stream becomes more robust but the LP stream will require higher SNR. For non-hierarchical transmission uniform modulation is used with □ □ = 1. Another important parameter is the code rate of the convolutional encoder, which controls the ruggedness of the data stream. The DVB-T system can be configured to be able to cope with a variety of different channel characteristics and application requirements, by selecting the appropriate inner coding rate and signal constellation. Simulated performance of the DVB-T system for combinations of different code rates and signal constellations can be found in [1]. Measured performance can be found in [2]. In the DVB-T receiver, after the FFT and channel correction, the demapper converts every incoming complex symbol into v-bit words, [y0,y1,..., yv-1]. It uses the value □ □ and v to select the signal constellation used by the modulator. These values are carried by bits in the Transmission Parameter Signalling (TPS)[1]. In the following section, we are going to derive the MUSCOD algorithm used by the demapper.

Double Q.P.S.K perform by changing the phase of the IN PHASE carrier from 0 degrees to 180 degrees. This is used to used to indicate four states of a 2-bit binary code.

Let we take one input frame as

"0010"

After modulation:=>"1111"

INTERLEAVER :

Interleaving along with error correction coding is an effective way to deal with different types of error in digital data communication. Error burst due to multipath fading and from other sources in a digital channel may be effectively combated by interleaving technique. The error correction codes (ECC) play very important role in modern digital communication systems. Interleaving technique is traditionally used to reduce bit error rate (BER) of digital transmission over a bursty channel Interleaving is a process to rearrange code symbols so as to spread burst of errors into random like errors and thereafter ECC can be applied to correct them. Interleaving improves [2] the performance of digital transmission at the cost of increased memory requirement, system complexity, and delay. Many papers in the literature have addressed the issue of designing interleaver in order to achieve low bit error rate On the contrary, very few papers have addressed the efficient implementation issue of the interleavers Block and Convolutional are the two popular interleavers employed in digital data transmission. The former is simple and easy to implement whereas the later offers advantages like lesser end-to-end delay and reduced memory requirement . in this paper we used a comparator to find the majority data bits and adds those majority bits to the data and passed to further status blocks .For example by finding the majority value from the modulated data, sign extension is done with the help of interleaver. That extension is used to find errors at receiver side.

Interleaver input=>"1111"

Output ="11111111"

IFFT :

This a technique used to convert frequency domain to time domain,after getting data from interlever block.butterfly digrams are used for this project.

FFT :

At receiver fft butterfly diagrams are used to convert time domain to frequency domain

DEINTERLEVER :

After data converted from time domain to frequency domain the data is passed to deinterlever to recover original data.

Symbol deinterleaver

The symbol deinterleaver is a block based deinterleaver. It acts on blocks of 1512 (2k mode) or 6048 (8k mode) v-bit words. The v-bit words yin=[y0,y1,..., yv-1] from the demapper are read sequentially into a vector Yin=(yin

0,yin 1,yin 2,...,yin Nmax-1). The deinterleavedvectorYout=(yout 0,yout 1,yout 2,.....,yout Nmax-1) isdefinedby: Yout q= yin H(q) for even OFDM symbols Yout H(q)= yin q for odd OFDM symbols where q = 0,.....,Nmax-1 and Nmax=1512 in the 2k mode and Nmax=6048 in 8k mode. H(q) is a permutation function defined in [1]. Each OFDM frame consists of 68 OFDM symbols, numbered from 0 to 67. Thus, this procedure is repeated 68 times for each OFDM frame.

Bit deinterleaver

The bit deinterleavers are block based. The block size is 126 bits, which is the same for each deinterleaver. However, the deinterleaving process is different. It works as follows: The symbol deinterleaver sends 126 v-bit words, yout=[a0,a1,..., av-1], to the bit deinterleaver, each deinterleaver receives one and only one bit from a given v-bit word. The most significant bit, a0, is sent to bit deinterleaver I0 and the least significant bit, av-1, is sent to I(v-1). For each bit deinterleaver, the input bit vector is: A(e) = (ae,0, ae,1, ae,2,...., ae,125) where e ranges from 0 to v-1 The deinterleaved output vector from each bit deinterleaver: B(e)=(be,0, be,1, be,2,...., be,125) is defined by: be,He(w) = ae,w w = 0, 1, 2,..., 125 where He(w) is a permutation function which is unique for each deinterleaver and is defined as follows:

I0: H0(w) = w
I1: H1(w) = (w + 63) mod 126
I2: H2(w) = (w + 105) mod 126
I3: H3(w) = (w + 42) mod 126
I4: H4(w) = (w + 21) mod 126
I5: H5(w) = (w + 84) mod 126

This block based bit deinterleaving process is repeated 12 times per OFDM symbol in the 2k mode, and 48 times for the case of 8k mo

DEMODULATION :

At the receiver the data is demodulated and passed to the fpga. This demodulator assumes the original message data stream was split into two streams, A and B, at the transmitter, with each converted to a PSK signal. The two PSK signals were then added, their carriers being in phase quadrature. The demodulator consists of two PSK demodulators, whose outputs, after analog-to-digital (A/D) conversion, are combined in a parallel-to-serial converter. This converter performs the recombination of the two channels to the original single serial stream. It can only do this if the carriers at the demodulator are synchronous, and correctly phased, with respect to those at the transmitter.

CLBS

In this we have defined 5 clbs,each consisting of different functionalities,each clb is tested by the pattern generated by embeded processor and replaces the faulty one with the existing blocks Many methods have been proposed to test FPGAs. In some works, the circuits under consideration are programmed FPGAs, in which logic circuits have been implemented **.** Since an FPGA can be programmed in many different ways, this method is not applicable to manufacturing time testing, as we do not know the final configuration. Testing faults in general FPGAs has been proposed by many researchers . In these methods, the FPGA under test is not mapped to a specific logic function. As a result, multiple test sessions are usually required, with each session dealing with one configuration. Stroud ***et al.*** propose a built-in self-test (BIST) structure for FPGAs . This method is attractive for two reasons. First of all, it requires no extra hardware. Since FPGAs are programmable, the BIST structure is implemented only once during test time. Secondly, the requirement for automatic test equipment (ATE) is much simplified.

The ATE only needs to download the configurations that are used during test time, while the actual testing process isconducted by the circuit itself. Our method is also based on the BIST technique, which means that the testing process is conducted by the chip itself, and the requirement for external ATE support is limited. The testing time is affected by the number of faults on the chip only, and is independent of the chip size. Since the die yield of larger chips is lower than for smaller ones with the same defect density, our method is especially attractive for larger chips. Traditional chip-level testing usually deals with fault detection only, while fault diagnosis is often conducted at the system level. This is because components in the chip cannot be repaired. However, faults in FPGAs can be easily tolerated by not including faulty elements in the final circuit. Therefore, FPGA chips with faults can still be used if we can identify the fault sites. Technology provides smaller, faster, and lower energy devices which allow more powerful and compact circuitry; however, these benefits come with a cost—the nanoscale devices may be less reliable. Thermal- and shot-noise estimations alone suggest that the fault rate of an individual nanoscale device may be orders of magnitude higher than today's devices. As a result, we can expect combinational logic to be susceptible to faults. So, in order to test any circuit or device we require separate testing technique which should be done automatically. For that purpose, we are going for BIST (built in self test).

Since in built-in self-test (BIST), test patterns are generated and applied to the circuit-under-test (CUT) by on-chip hardware, minimizing hardware overhead is a major concern of BIST implementation. Unlike stored pattern BIST, which requires high hardware overhead due to memory devices required to store precomputed test patterns, pseudorandom BIST, where test patterns are generated by pseudorandom pattern generators such as linear feedback shift registers (LFSRs), requires very little hardware overhead. However, achieving high fault coverage for CUTs that contain many random pattern resistant faults (RPRFs) only with (pseudo) random patterns generated by an LFSR often require unacceptably long test sequences thereby resulting in prohibitively long test time. The random pattern test length required to achieve high fault coverage is often determined by only a few RPRFs. Though the attainment of high fault coverage with practical lengths of test sequences is still one major concern of BIST techniques, reducing switching activity has become another important objective. It has been observed that switching activity during test application is often significantly higher than that during normal operation. The correlation between consecutive random patterns generated by an LFSR is low—this is a well-known property of LFSR generated patterns. On the other hand, significant correlation exists between consecutive patterns during the normal operation of a circuit. A **field-programmable gate array** (**FPGA**) is an integrated circuit designed to be configured by the customer or designer after manufacturing—hence "field-programmable". The FPGA configuration is generally specified using a hardware description language (HDL), similar to that used for an application-specific integrated circuit (ASIC) (circuit diagrams were previously used to specify the configuration, as they were for ASICs, but this is increasingly rare). FPGAs can be used to implement any logical function that an ASIC could perform. The ability to update the functionality after shipping, partial re-configuration of the portion of the design and the low non-recurring engineering costs relative to an ASIC design (notwithstanding the generally higher unit cost), offer advantages for many applications.

FPGAs contain programmable logic components called "logic blocks", and a hierarchy of reconfigurable interconnects that allow the blocks to be "wired together"—somewhat like many (changeable) logic gates that can be inter-wired in (many) different configurations. Logic blocks can be configured to perform complex combinational functions, or merely simple logic gates like AND and XOR. In most FPGAs, the logic blocks also include memory elements, which may be simple flip-flops or more complete blocks of memory.[2]

In addition to digital functions, some FPGAs have analog features. The most common analog feature is programmable slew rate and drive strength on each output pin, allowing the engineer to set slow rates on lightly loaded pins that would otherwise ring unacceptably, and to set stronger, faster rates on heavily loaded pins on high-speed channels that would otherwise run too slow. Another relatively common analog feature is differential comparators on input pins designed to be connected to differential signaling channels.

A few "mixed signal FPGAs" have integrated peripheral Analog-to-Digital Converters (ADCs) and Digital-to-Analog Converters (DACs) with analog signal conditioning blocks allowing them to operate as a system-on-a-chip.[5] Such devices blur the line between an FPGA, which carries digital ones and zeros on its internal programmable interconnect fabric, and field-programmable analog array (FPAA), which carries analog values on its internal programmable interconnect fabric.The most common FPGA architecture consists of an array of logic blocks (called Configurable Logic Block, CLB, or Logic Array Block, LAB, depending on vendor), I/O pads, and routing channels. Generally, all the routing channels have the same width (number of wires). Multiple I/O pads may fit into the height of one row or the width of one column in the array.

An application circuit must be mapped into an FPGA with adequate resources. While the number of CLBs/LABs and I/Os required is easily determined from the design, the number of routing tracks needed may vary considerably even among designs with the same amount of logic. For example, a crossbar switch requires much more routing than a systolic array with the same gate count. Since unused routing tracks increase the cost (and decrease the performance) of the part without providing any benefit, FPGA manufacturers try to provide just enough tracks so that most designs that will fit in terms of LUTs and IOs can be routed. This is determined by estimates such as those derived from Rent's rule or by experiments with existing designs.

In general, a logic block (CLB or LAB) consists of a few logical cells (called ALM, LE, Slice etc.). A typical cell consists of a 4-input Lookup table (LUT), a Full adder (FA) and a D-type flip-flop, as shown below. The LUTs are in this figure split into two 3-input LUTs. In *normal mode* those are combined into a 4-input LUT through the left mux. In *arithmetic* mode, their outputs are fed to the FA. The selection of mode is programmed into the middle

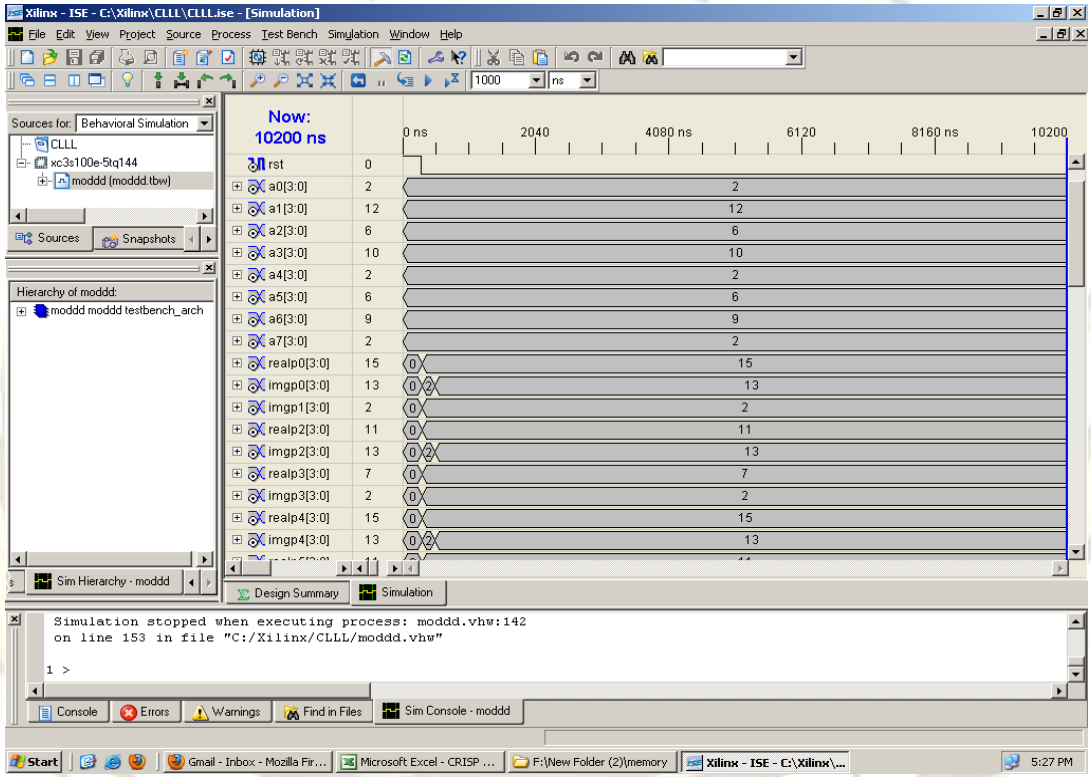
multiplexer. The output can be either synchronous or asynchronous, depending on the programming of the mux to the right, in the figure example. In practice, entire or parts of the FA are put as functions into the LUTs in order to save space

ALMs and Slices usually contains 2 or 4 structures similar to the example figure, with some shared signals.CLBs/LABs typically contains a few ALMs/LEs/Slices.

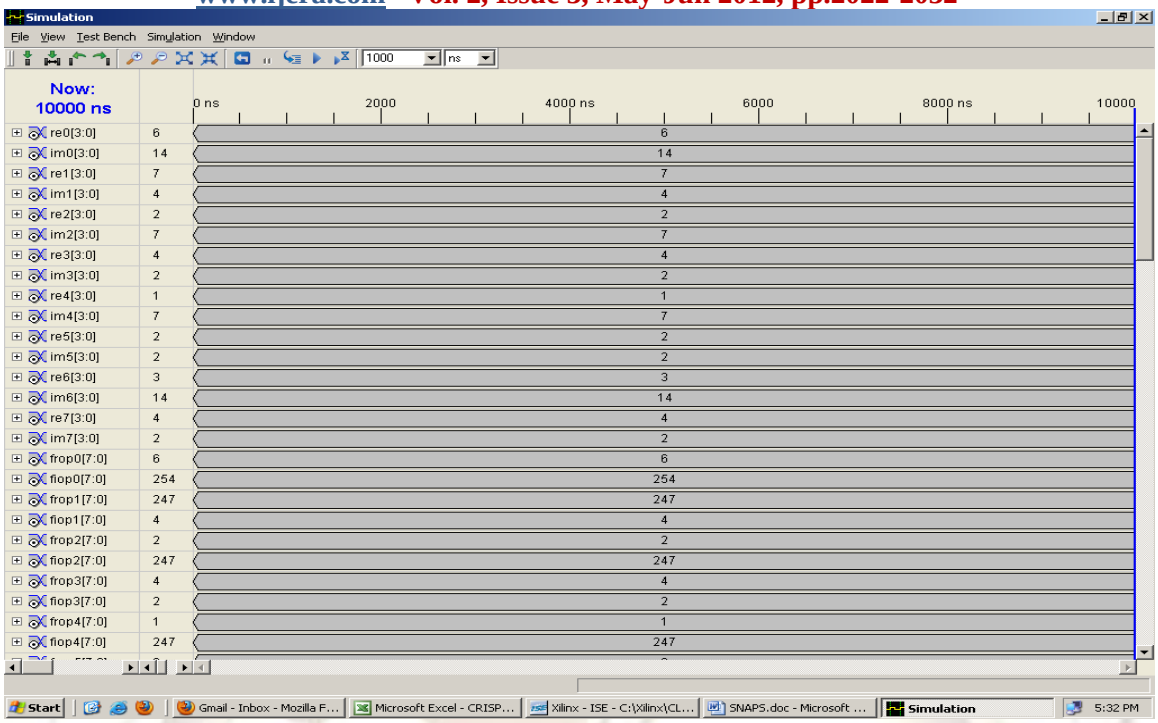
In recent years, manufacturers have started moving to 6-input LUTs in their high performance parts, claiming increased performance.Since clock signals (and often other high-fanout signals) are normally routed via special-purpose dedicated routing networks in commercial FPGAs, they and other signals are separately managed.

RESULTS:

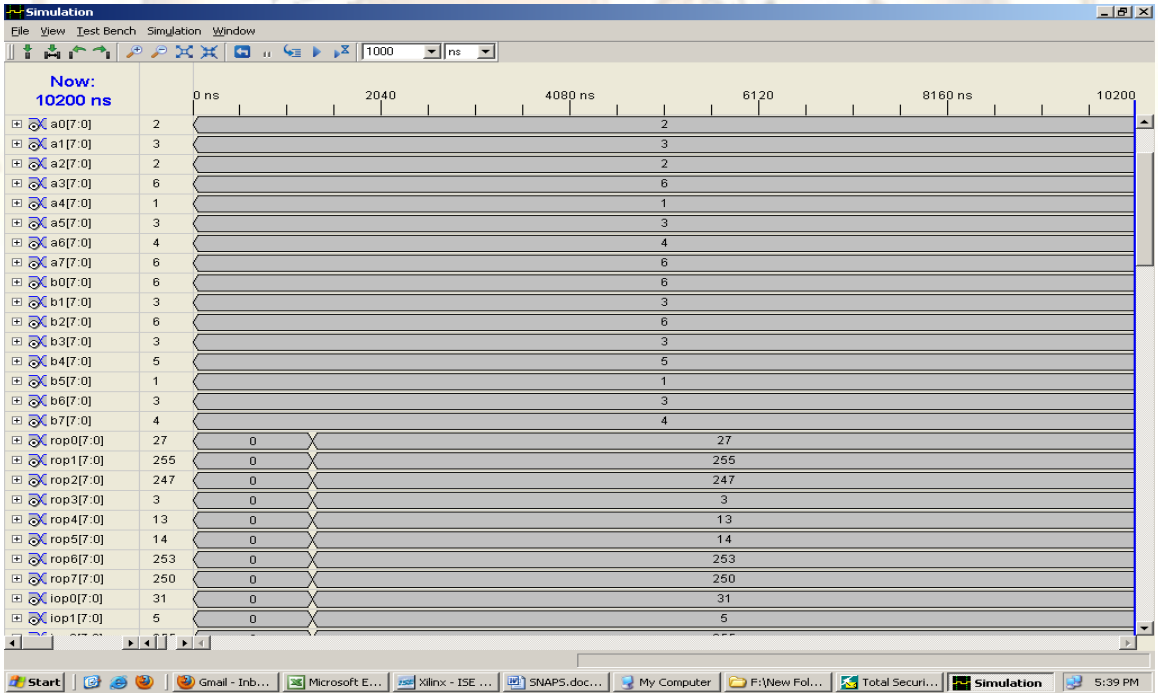
MODULATION OUTPUT



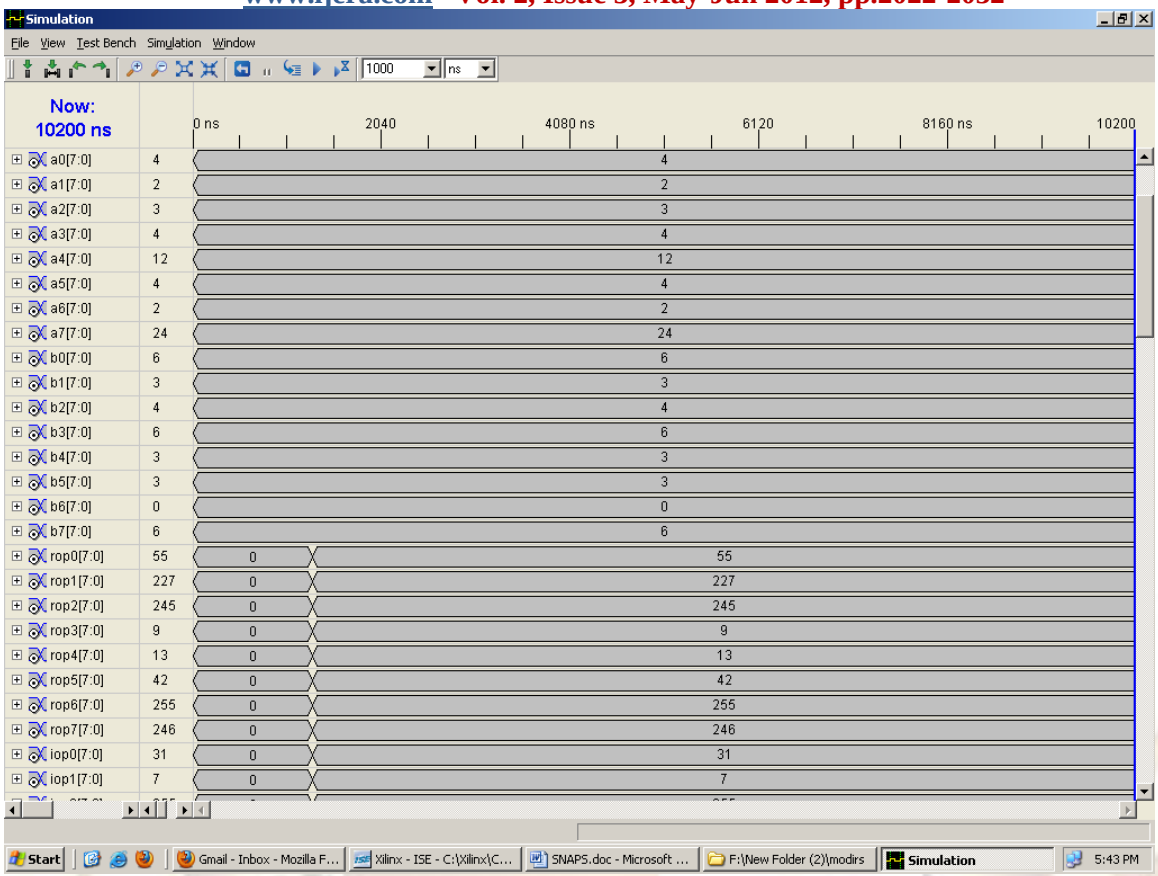
INTERLEAVER


Simulation
File View Test Bench Simulation Window
Now: 10000 ns
0 ns 2000 4000 ns 6000 8000 ns 10000
re0[3:0] 6
im0[3:0] 14
re1[3:0] 7
im1[3:0] 4
re2[3:0] 2
im2[3:0] 7
re3[3:0] 4
im3[3:0] 2
re4[3:0] 1
im4[3:0] 7
re5[3:0] 2
im5[3:0] 2
re6[3:0] 3
im6[3:0] 14
re7[3:0] 4
im7[3:0] 2
frop0[7:0] 6
flop0[7:0] 254
frop1[7:0] 247
flop1[7:0] 4
frop2[7:0] 2
flop2[7:0] 247
frop3[7:0] 4
flop3[7:0] 2
frop4[7:0] 1
flop4[7:0] 247

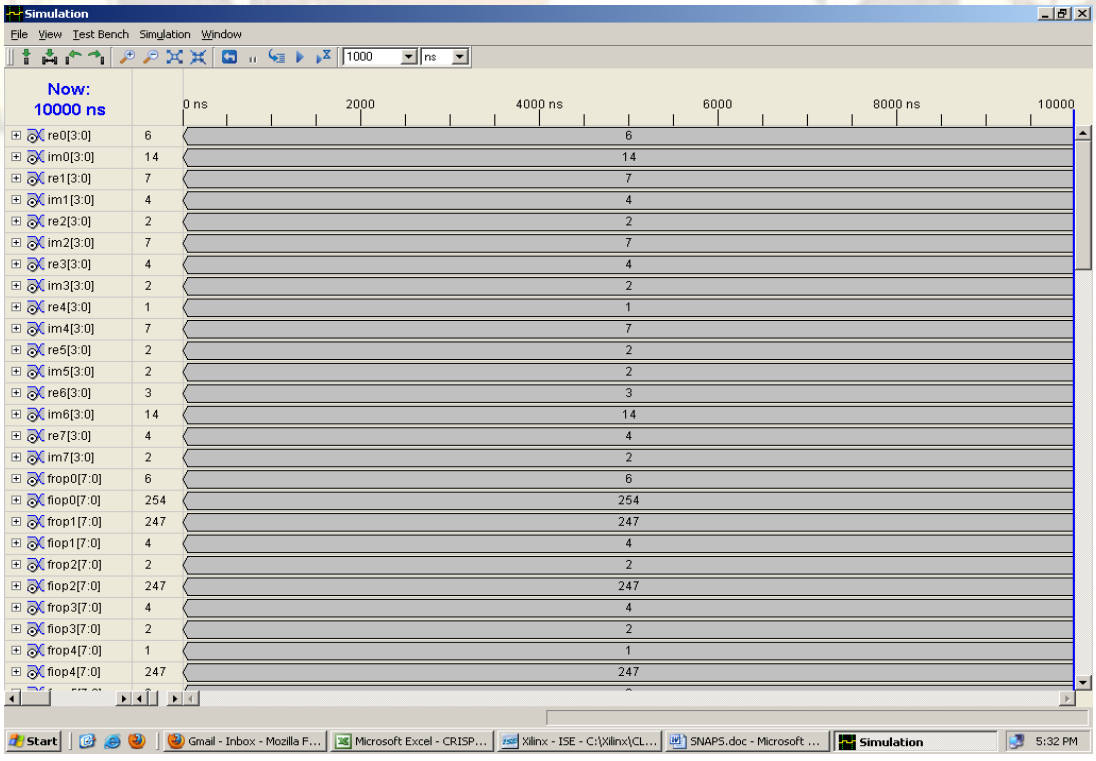
IFFT


Simulation
File View Test Bench Simulation Window
Now: 10200 ns
0 ns 2040 4080 ns 6120 8160 ns 10200
a0[7:0] 2
a1[7:0] 3
a2[7:0] 2
a3[7:0] 6
a4[7:0] 1
a5[7:0] 3
a6[7:0] 4
a7[7:0] 6
b0[7:0] 6
b1[7:0] 3
b2[7:0] 6
b3[7:0] 3
b4[7:0] 5
b5[7:0] 1
b6[7:0] 3
b7[7:0] 4
rop0[7:0] 27
rop1[7:0] 255
rop2[7:0] 247
rop3[7:0] 3
rop4[7:0] 13
rop5[7:0] 14
rop6[7:0] 253
rop7[7:0] 250
iop0[7:0] 31
iop1[7:0] 5

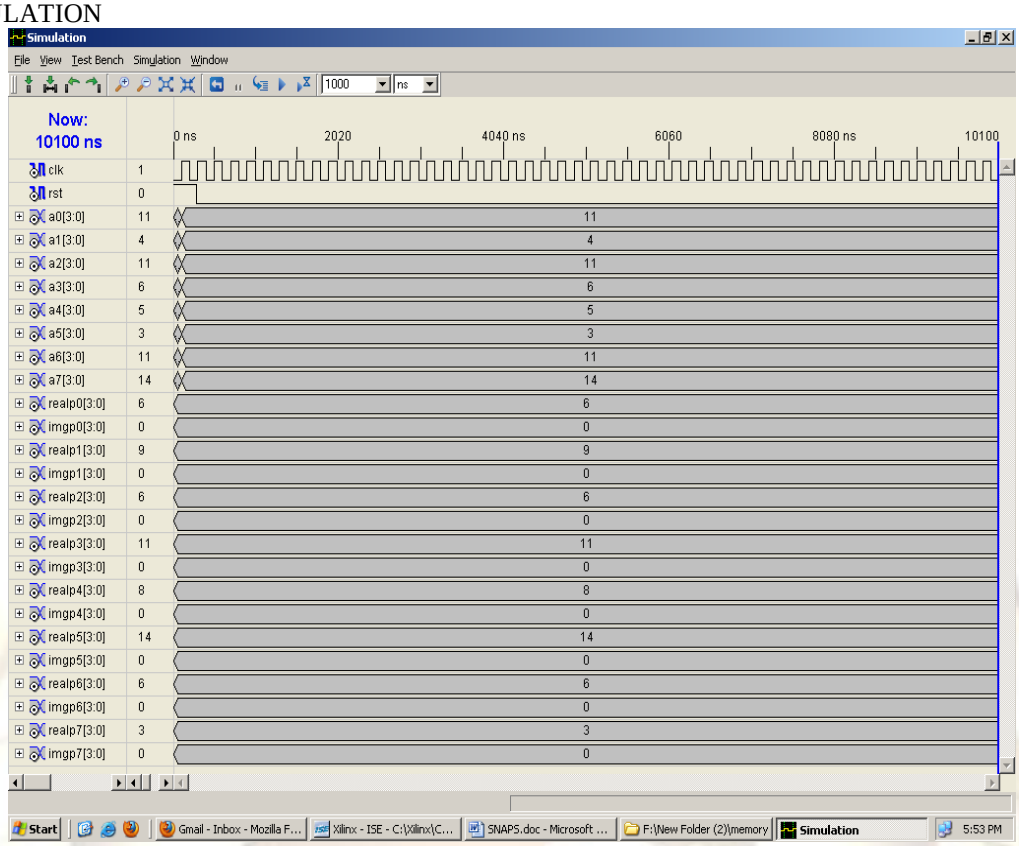
FFT



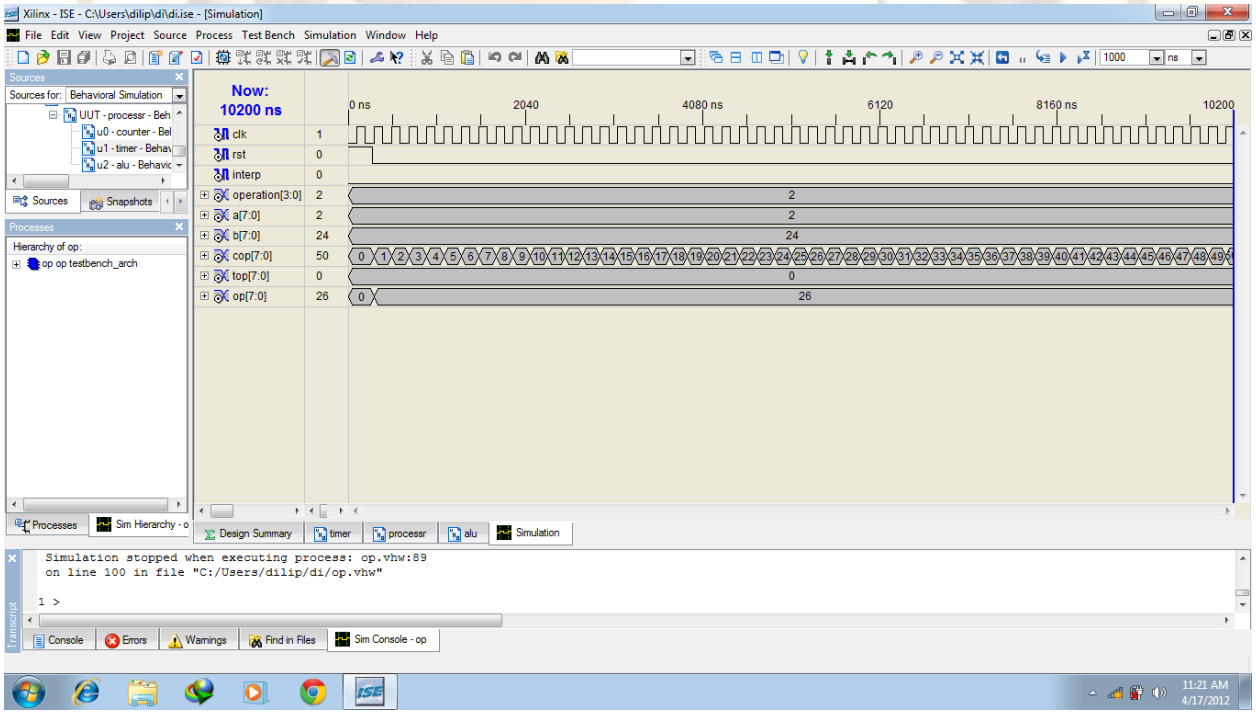
DEINTERLEAVER



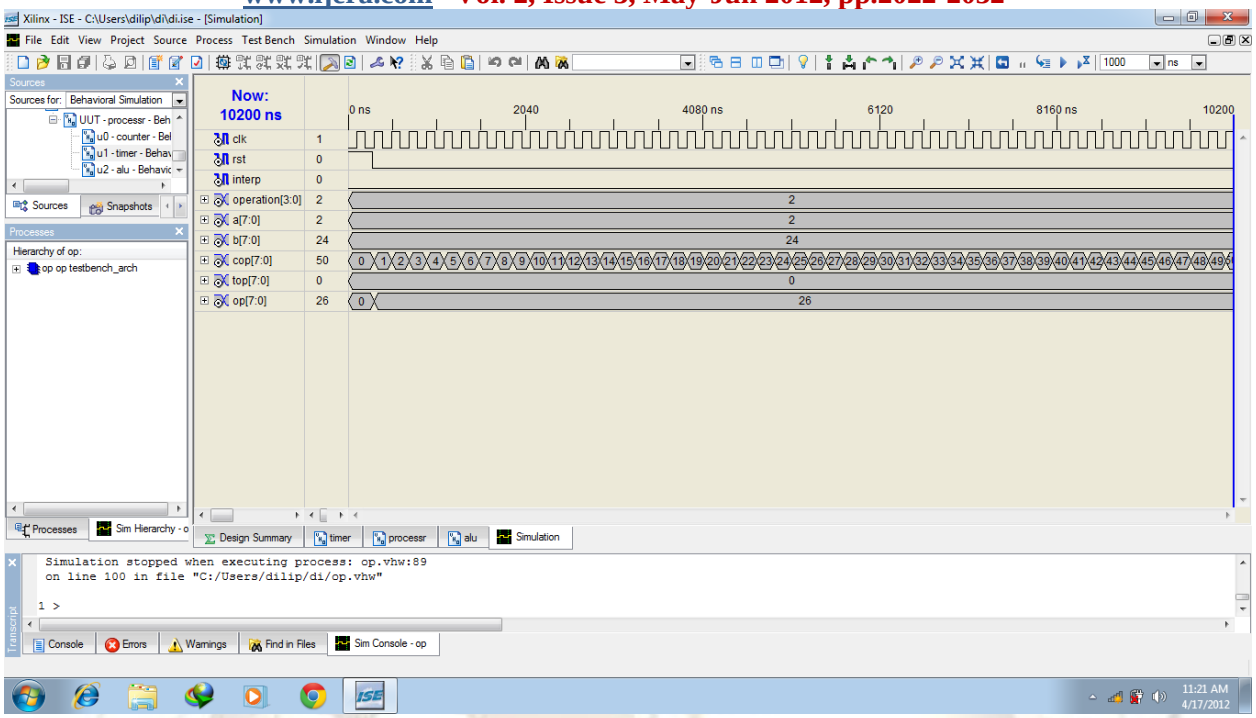
DEMODULATION



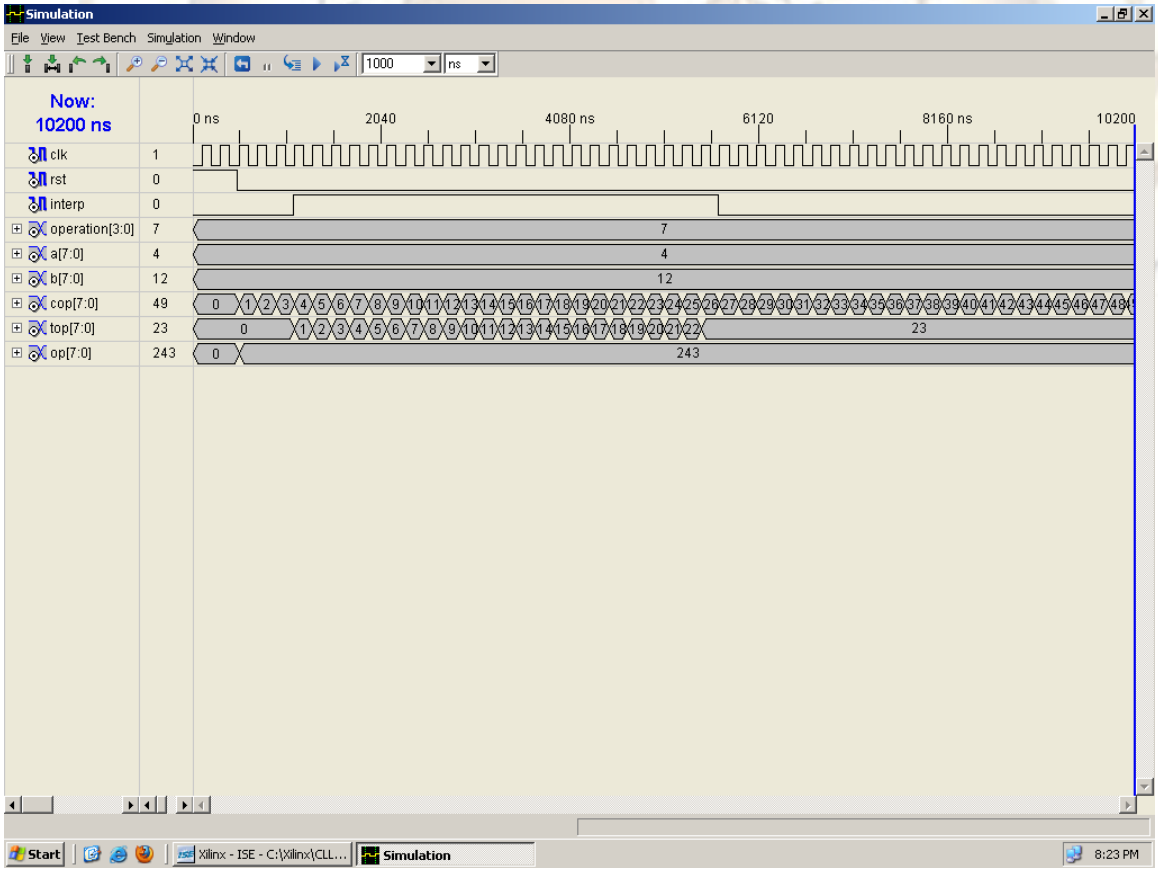
PROCESSOR OUPUT WITHOUT INTERRUPT



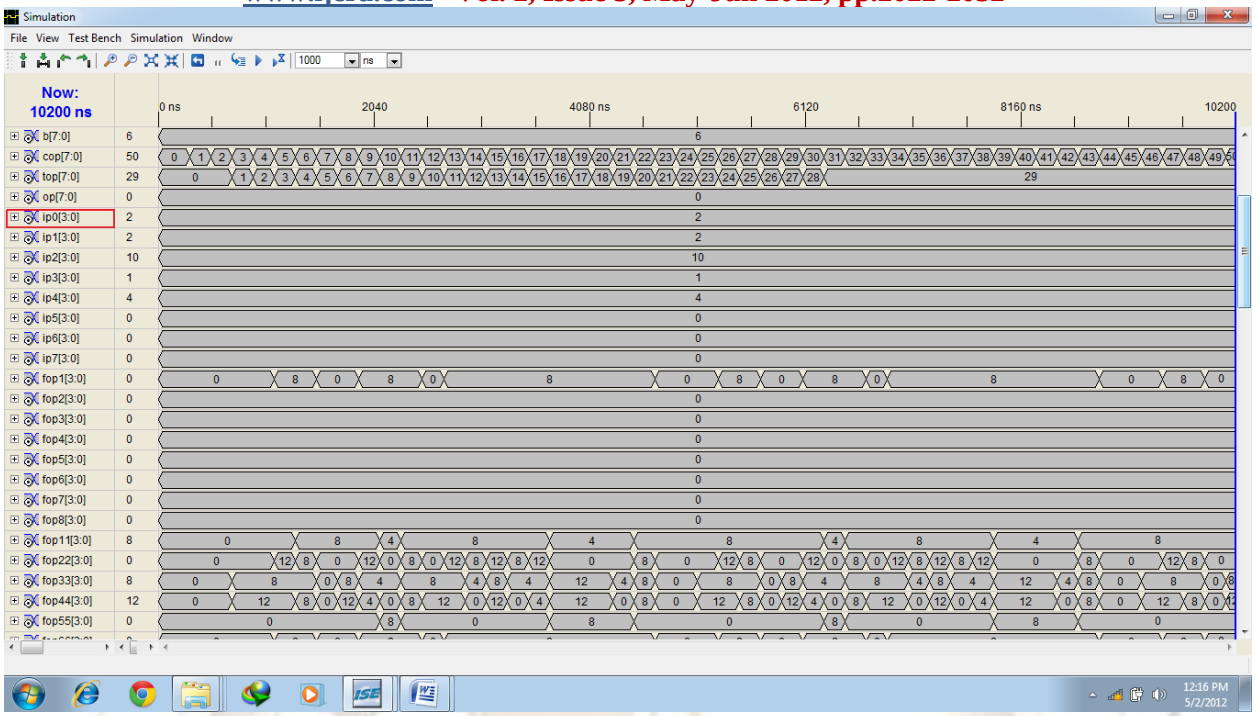
PROCESSOR OUPUT WITH INTERUPPT



PROCESSOR OUTPUT WITH INTERUPT



FINAL OUTPUT



CONCLUSION:

Hence handling the large numbers of faults in F.P.G.A. configurable logic blocks. A key novel feature is the reuse of defective logic blocks to increase the number of effective spares and extend the mission life. To increase fault tolerance, we not only use nonfaulty parts of defective or partially faulty logic blocks, but we also use faulty parts of defective logic blocks in nonfaulty modes. By using and reusing faulty resources, our multilevel approach extends the number of tolerable faults beyond the number of currently available spare logic resources. A challenging aspect of developing applications that target the interfacing of the processor with the surrounding reconfigurable logic is main criteria in this project. O.C.P protocol interfaces the embedded processor with F.P.G.A. Testing all configurable logic blocks is under care of embedded processor. The embedded processor set the test mode for F.P.G.A, when it is needed it stops when it is in sleep mode.

REFERENCES:

[1] Enhancement of Fault Injection Techniques Based on the Modification of VHDL Code Juan-Carlos Baraza, Joaquín Gracia, Sara Blanc, Daniel Gil, and Pedro-J. Gil, *Member, IEEE*

[2] C. Constantinescu, "Impact of deep submicron technology on dependability of VLSI circuits," in *Proc. DSN*, 2002, pp. 205–209.

[3] P. Shivakumar, M. Kistler, S. W. Keckler, D. Burger, and L. Alvisi, "Modeling the effect of technology trends on soft error rate of combinational logic," in *Proc. DSN*, 2002, pp. 389–398.

[4] E. Jenn, J. Arlat, M. Rimén, J. Ohlsson, and J. Karlsson, "Fault injection into VHDL models: The MEFISTO tool," in *Proc. FTCS*, 1994, pp. 356–363

[5] V. Sieh, O. Tschäche, and F. Balbach, "VERIFY: Evaluation of reliability using VHDL-models with embedded fault descriptions," in *Proc. FTCS*, 1997, pp. 32–36.

[6] J. Boué, P. Pétillon, and Y. Crouzet, "MEFISTO-L: A VHDL-based fault injection tool for the experimental assessment of fault tolerance,"in *Proc. FTCS*, 1998, pp. 168–173.

[7] A VHDL Primer by J.Bhaskar.

[8] Basic VLSI Design by Douglas A. Pucknell and Kamaran Eshrainghian.

[9] Digital Design, Principles and Practices by John F. Wakerly.