IOSR Journal of Engineering (IOSRJEN) **www.iosrjen.org**
ISSN (e): 2250-3021, ISSN (p): 2278-8719
Vol. 15, Issue 4, April 2025, ||Series -1|| PP 59-67

Design And Implementation Of A Secure Xilinx Based Electronic Voting Machine

BORUSU VEERA RAGHAVA[1], DONKA SANTHOSH KUMAR[2], AKKINA POOJITHA[3], JAGATA SIVAMANI[4], G.VEERAPANDU[5]
[1,2,3,4]B.Tech Scholars, Department of ECE,
[5]Assistant Professor, Department of ECE, Aditya College of Engineering and Technology, Surampalem, Kakinada, AP, India.

ABSTRACT- Voting in the paper was a time consuming and erroneous process. Voting by electronic voting machine (EVM) is a simple, secure, and timesaving technology. The proposed paper consists of the parties assigned to the election can be represented in the box along with the total number of contestants participating in the election process. Biometric is also included in the machine which is a key role paid to avoid fraudulent voters and to prevent fraud. In the voting process begins when a voter votes for a particular party through the bio metric process after the buzzer will display a notification of who voted for the party displayed on the screen. After the completion of the voting process the votes are confirmed by comparing the votes cast with the contestants in their respective The proposed method is useful for college level to all election boards and digital voting system for real-time applications ranging from community college elections to university level elections, as it has the advantage of being able to be reprogrammed to perform different functions according to the user's wishes, which helps to reduce prevalence expenditures.

KEYWORDS- VHDL, FPGA LCD Display, Bio metric, Voting Machine

I. INTRODUCTION

In any vote-based system, casting a ballot is the sole criteria for citizens to choose their representatives. As a result, this entire method should be carried out with the utmost care so that only a reasonable and deserving candidate is chosen solely on the basis of popular vote. Previously, judgments were made using a poll paper structure, in which people cast their votes for their favourite challenger by simply stamping against his or her name, [1] but this technique was prone to flaws such as vote counting and unjustified results Electronic voting machines were devised to address each of these inequalities. Regardless, the concept of a simple electronic voting system with a disposable memory card was a bit random[2]. We designed an electronic voting machine in Verilog HDL using Xilinx ISE 14.7i that can be implemented on FPGA (Field Programmable Gate Array)[3] hardware since we realise how difficult it is to manage control signals.Additionally, this execution includes a secret key that is computerised and difficult to hackatory and qualitative peace engineering perspective.

India is a democratic nation wherein the people are directly involved in electing the candidates for the parliament. It is difficult to practice direct democracy in countries like India, China and several other highly populated States. Elections are a rampart of people's liberty and it is a process of putting a check on undemocratic tendencies. Elections are the backbone of a democratic system, therefore it is necessary to employ efficient methods of conducting elections.

Paper ballot employs uniform official ballots on which names of various parties are printed, voters can come and select the required party .The paper ballot was first adopted in Australian state of Victoria, it thereafter became popular as 'Australian Ballot'. The major drawback in this process is lack of efficiency in counting the votes, dependency on human resource and entertains tampering of votes.

To overcome these flaws electronic voting machine is being used. Electronic voting machine is more efficient than paper ballot process in terms of cost effectiveness since latter uses more usage of paper. EVMs are user friendly as voting process is made easy through push buttons. Votes casted in different centers using EVMs can be uploaded onto a single central unit which makes easier to announce the results.

Idea of EVM in India was emerged by the Chief Election Commissioner in 1977 and he advised E-voting to save time, expenditure on storage, transportation and security of ballot paper. In order to overcome these problems and errors in counting of ballot papers, electronic voting was recommended. Challenges was to develop a machine which would fit into the existing election procedure and appear familiar to the voter, and would be transparent and acceptable to all.

II. Related work

The demonstrated in their paper that to overcome these difficulties and build a good electoral method,[4] implementation of electronic mechanical devices in the digital domain is given in this paper. It is difficult to tamper votes in the digital domain and provides a secure and safe technique for conducting elections. May, P., Ehrlich, H.C., and Steinke,[5] In their work, Czajkowski, K., Fitzgerald, S., Foster, I., and Kesselman, C.[6] said that the conventional selection approach was a lengthy and error-prone process.[7] Electronic mechanical device (EVM) polling is a simple, safe,[9] and secure method that takes very little time.[8] Only one vote is accepted by the current Electronic Mechanical Devices (EVMs) used in LOK SABHA and an ASSEMBLY election.[10].

III. PROPOSED SYSTEM

Electronic voting machine plays major role in democratic society for voting system to avoid rigging. For giving privacy for voters and secure, integrity in elections and counting of votes and to avoid age old ballot papers in voting process electronic voting machines are highly necessary. In our work, voting process is considered with 3 contestants namely party0, party1, party2 and seg0, seg1, seg2 are registers to store the votes of party0, party1, party2 respectively. Venable is used to enable voting process and clk is used to enable the entire election process. Vswitch is a 2-bit input which can be used to activate the voting for each contestant respectively. If vswitch is 00, then voters can vote for party0, if vswitch is 01, then voters can vote for party1 and if vswitch is 10, then voters can vote for party2. Dout will hold the total votes of all contestants, invalid is the signal that gets activated when venable is not in high state to start the voting process and if voter tries to vote for any contestant. Security can be incorporated by OTP generation and verification of it from the voter's mobile before allowing the voter for voting process. A pseudo random binary sequence generator (PRBS) can be used to generate a 6-bit random number which can be used as OTP. Hence, the entire process of digital electronic voting system may help in a safe, secure and integrity process in a democratic society.

. The OTP for the above applications are generated by Linear Feedback Shift Register (LFSR). The 8 bit pattern generation circuit to generate a pattern X^7+X^5 +X^4+X^3+1 is shown in Figure 1. As the technology advances, the need for low power consumption circuit increases. Thus, the circuit is generally expected to be designed in such a way that it should consume less power, occupy minimum area with improved response time. The use of flip-flop with activated clock in the register design consumes more power which is not sufficient for high throughput, so pulsed latches are used in the place of flip-flops in this proposed work.

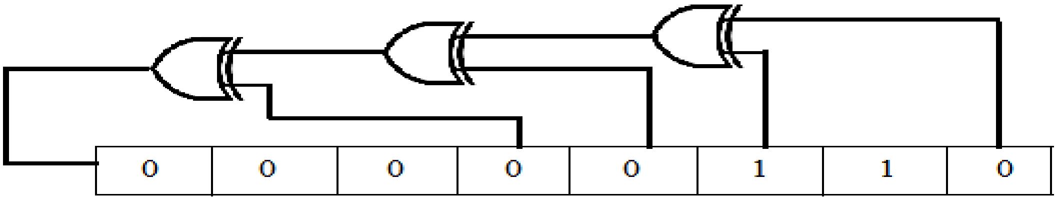


Figure. 8-bit LFSR circuit

For reducing the power consumption of the device, various methodologies are available in the literature. Dropping the number of transitions is one of the means for power optimization. Transitions are reduced by swapping the bits and applying clock to half part of the circuit. Clock gating is also employed for power optimization. Although various optimization techniques are implemented for minimizing the power consumption of the device, they are not eventually much effective by the means of reducing the response time and area. Like power optimization techniques, techniques for minimizing the area and increasing the speed are also employed in [1]-[5]. The conventional method of serial to parallel architecture and pipelining algorithms are used to increase the speed of the shift register. Also calculation of output value only by considering the past feedback value in the transposed serial architecture, increases the speed. The transformation from long LFSR sequence to several short LFSR sequence in series reduces the overhead. Though several techniques are used to reduce the power, area and speed, they are not efficient in terms of critical path delay.

LFSR is a serially connected flip-flop configuration – shift register configuration – with feedbacks from certain flip-flop outputs – taps – that are XORed together –added in modulo 2 – and connect back to first flip-flop's input. The number and position of taps determine the length and sequence of generated PRBS pattern. An exemplary 8 stage LFSR with tap connections that provide maximum possible sequence length (2n-1 patterns)

CA structure is quite similar to that of LFSR, with the inherent shift register configuration. The basic difference from the LFSR is, the interconnections of individual flip Testing and flops now always include an XOR operation and there is no global feedback. CA consist of 2 types of primary cells, namely 90 and 150 cells, and certain combination of these cells reveal maximum length sequences. The only difference between 90 and 150 cells is, 150 cells have an additional self feedback from the flip-flop output to back to its input. An exemplary 4 stage CA, with appropriate 90 and 150 cell configuration for maximum length PRBS.

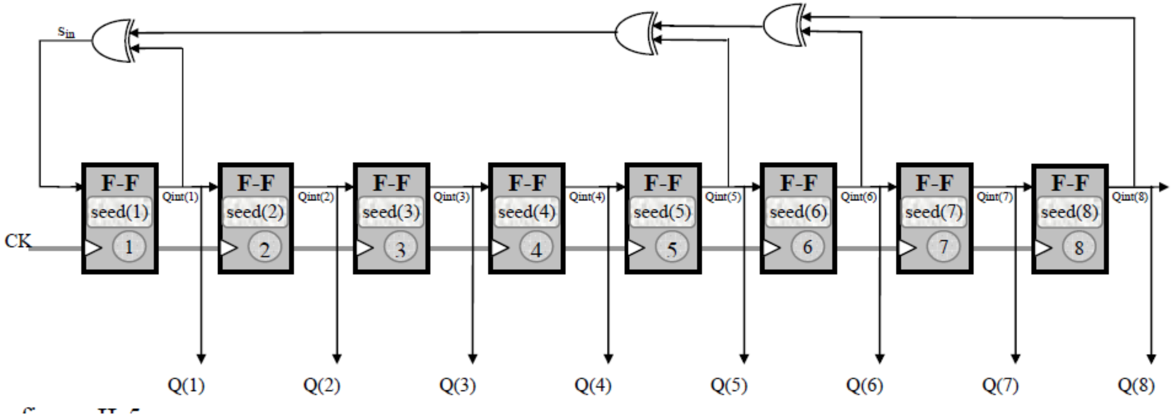


Figure An 8 stage maximum length LFSR

The pseudorandom number generator has the following disadvantages:

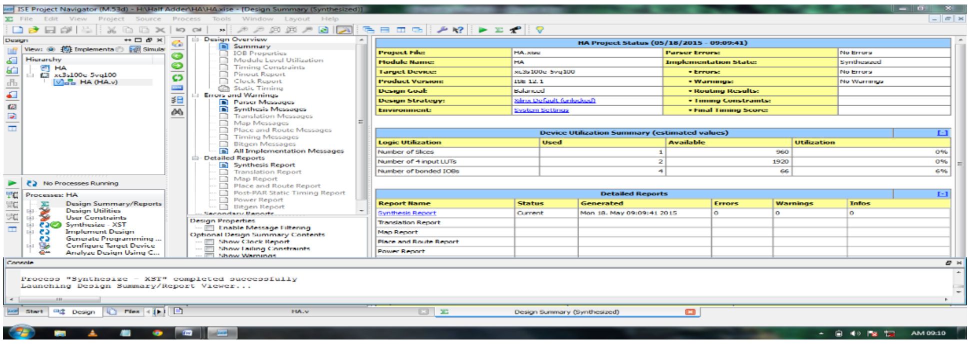
- When few bits of the plain text and their corresponding ciphertext are known, then it is easy to hack the data by creating the remaining bits in the key sequence.
- It leads to less security.

IV. Implementation:

Synthesis (XST)
-Produce a netlist file starting from an HDL description
Translate (NGDBuild)
Converts all input design netlists and then writes the results into a single merged file, that describes logic and constraints.
Mapping (MAP)
Maps the logic on device components.
Takes a netlist and groups the logical elements into CLBs and IOBs (components of FPGA).
Place And Route (PAR)
Place FPGA cells and connects cells.
Bit stream generation

XILINX Design Process:
Step 1: Design entry
HDL (Verilog or VHDL, ABEL x CPLD), Schematic Drawings, Bubble
Diagram
Step 2: Synthesis
Translates .v, .vhd, .sch files into a netilist file (.ngc)
Step 3: Implementation
FPGA: Translate/Map/Place & Route, CPLD: Fitter
Step 4: Configuration/Programming
Download a BIT file into the FPGA
Program JEDEC file into CPLD
Program MCS file into Flash PROM
Simulation can occur after steps 1, 2, 3
The tools used in this thesis are **XILINX ISE 14.7** for simulation and Synthesis. The programs are written in verilog language.
Xilinx Tools is a suite of software tools used for the design of digital circuits implemented using Xilinx Field Programmable Gate Array (FPGA) or Complex Programmable Logic Device (CPLD). The design procedure consists of (a) design entry, (b) synthesis and implementation of the design, (c) functional simulation and (d) testing and verification. Digital designs can be entered in various ways using the above CAD tools: using a schematic entry tool, using a hardware description language (HDL) – Verilog or VHDL or a combination of both. In this thesis we will only use the design flow that involves the use of Verilog HDL.

Creating a New Project
Xilinx Tools can be started by clicking on the Project Navigator Icon on the Windows desktop. This should open up the Project Navigator window on your screen. This window shows (see Figure 1) the last accessed project.



HA Project Status (05/18/2015 - 09:09:41)			
Project File:	HA.xise	Parser Errors:	No Errors
Module Name:	HA	Implementation State:	Synthesized
Target Device:	xc3s100e-5vq100	• Errors:	No Errors
Product Version:	ISE 12.1	• Warnings:	No Warnings
Design Goal:	Balanced	• Routing Results:	
Design Strategy:	Xilinx Default (unlocked)	• Timing Constraints:	
Environment:	System Settings	• Final Timing Score:	

Device Utilization Summary (estimated values)			
Logic Utilization	Used	Available	Utilization
Number of Slices	1	960	0%
Number of 4 input LUTs	2	1920	0%
Number of bonded IOBs	4	66	6%

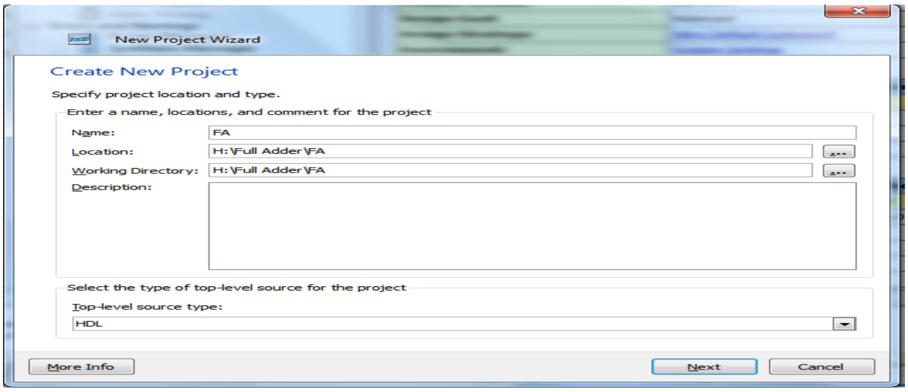
Detailed Reports					
Report Name	Status	Generated	Errors	Warnings	Infos
Synthesis Report	Current	Mon 18. May 09:09:41 2015	0	0	0
Translation Report					
Map Report					
Place and Route Report					
Power Report					

Console

```
Process "Synthesize - XST" completed successfully
Launching Design Summary/Report Viewer...
```

Opening a project

Select **File->New Project** to create a new project. This will bring up a new project window (Figure 2) on the desktop. Fill up the necessary entries as follows:



Create New Project

Specify project location and type.

Enter a name, locations, and comment for the project

Name: FA

Location: H:\Full Adder\FA

Working Directory: H:\Full Adder\FA

Description:

Select the type of top-level source for the project

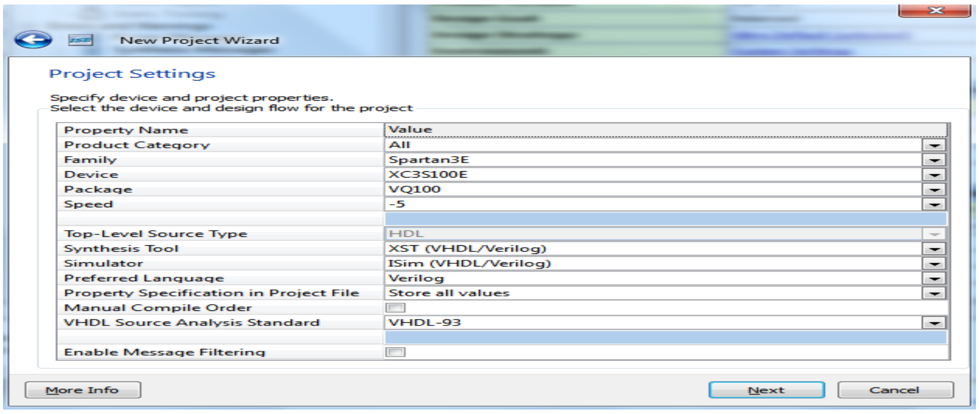
Top-level source type:

HDL

Project Name: Write the name of your new project

Project Location: The directory where you want to store the new project (Note: DO NOT specify the project location as a folder on Desktop or a folder in the Xilinx\bin directory. Your H: drive is the best place to put it. The project location path is NOT to have any spaces in it eg: H:\Full Adder\F A is NOT to be used).Leave the top level module type as HDL.

Clicking on NEXT should bring up the following window:



Project Settings

Specify device and project properties.
Select the device and design flow for the project

Property Name	Value
Product Category	All
Family	Spartan3E
Device	XC3S100E
Package	VQ100
Speed	-5
Top-Level Source Type	HDL
Synthesis Tool	XST (VHDL/Verilog)
Simulator	ISim (VHDL/Verilog)
Preferred Language	Verilog
Property Specification in Project File	Store all values
Manual Compile Order	□
VHDL Source Analysis Standard	VHDL-93
Enable Message Filtering	□

For each of the properties given below, click on the '**value**' area and select from the list of values that appear.

Device Family: Family of the FPGA/CPLD used. In this thesis we will be using the **Spartan3E FPGA's**.

Device: The number of the actual device. For this lab you may enter **XC3S100E** (this can be found on the attached prototyping board)

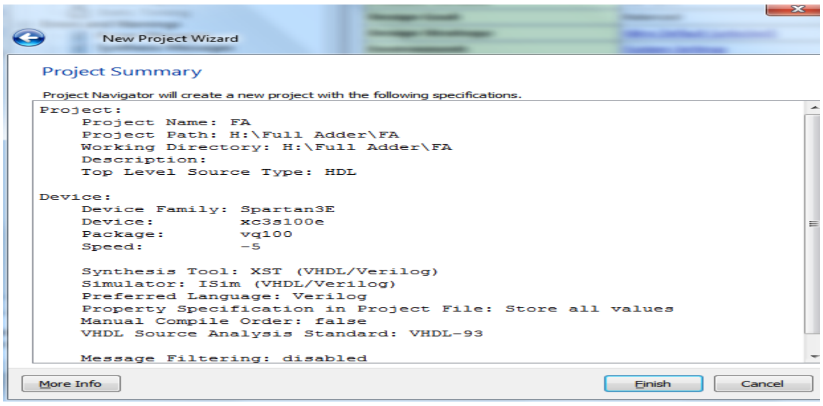
Package: The type of package with the number of pins. The Spartan FPGA used in this lab is packaged in **VQ100** package.

Speed Grade: The Speed grade is "-5".

Synthesis Tool: **XST** [VHDL/Verilog]

Simulator: The tool used to simulate and verify the functionality of the design. Modelsim simulator is integrated in the Xilinx ISE. Hence choose “Modelsim-XE Verilog” as the simulator or even Xilinx ISE Simulator can be used.

Then click on **NEXT** to save the entries.



Project Summary

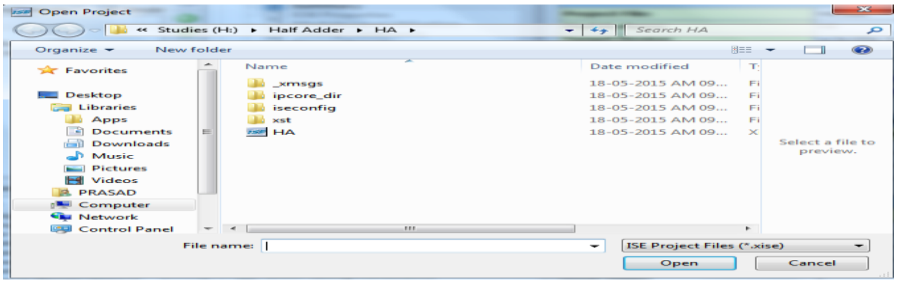
Project Navigator will create a new project with the following specifications.

```
Project:
     Project Name: FA
     Project Path: H:\Full Adder\FA
     Working Directory: H:\Full Adder\FA
     Description:
     Top Level Source Type: HDL

Device:
     Device Family: Spartan3E
     Device:        xc3s100e
     Package:       vq100
     Speed:         -5

     Synthesis Tool: XST (VHDL/Verilog)
     Simulator: ISim (VHDL/Verilog)
     Preferred Language: Verilog
     Property Specification in Project File: Store all values
     Manual Compile Order: false
     VHDL Source Analysis Standard: VHDL-93

     Message Filtering: disabled
```

A project summary window is opened click on finish.

In order to open an existing project in Xilinx Tools, select **File->Open Project** to show the list of projects on the machine. Choose the project you want and click **OK.**

Clicking on NEXT on the above window brings up the following window:

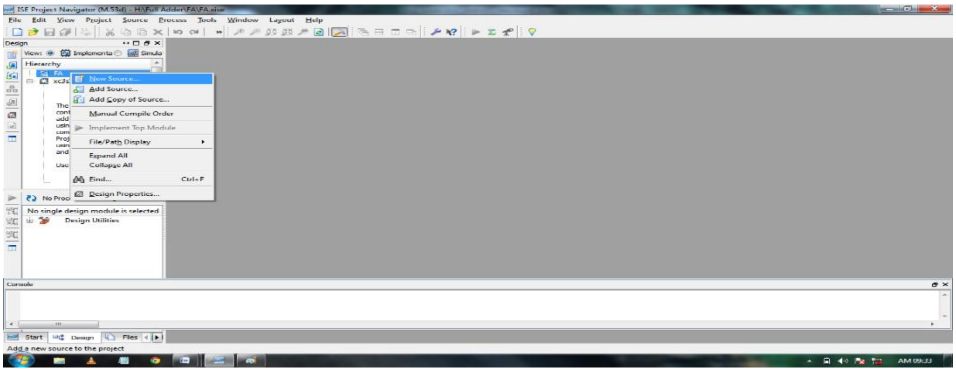


Name	Date modified	T
_xmsgs	18-05-2015 AM 09...	Fi
ipcore_dir	18-05-2015 AM 09...	Fi
iseconfig	18-05-2015 AM 09...	Fi
xst	18-05-2015 AM 09...	Fi
HA	18-05-2015 AM 09...	X

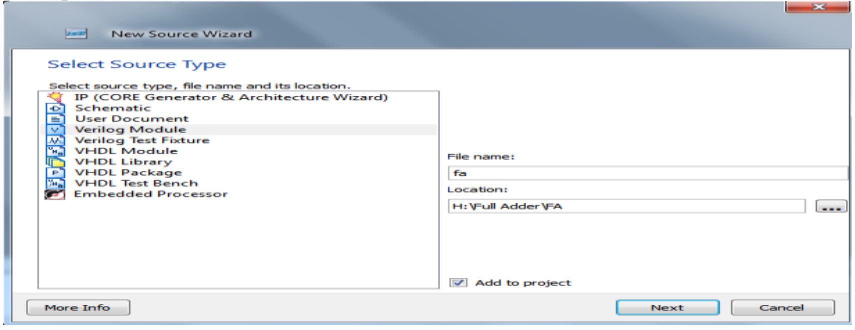
File name: |

ISE Project Files (*.xise)

If creating a new source file, Click on the NEW SOURCE.



A window pop up is opened.



Select Source Type

Select source type, file name and its location.

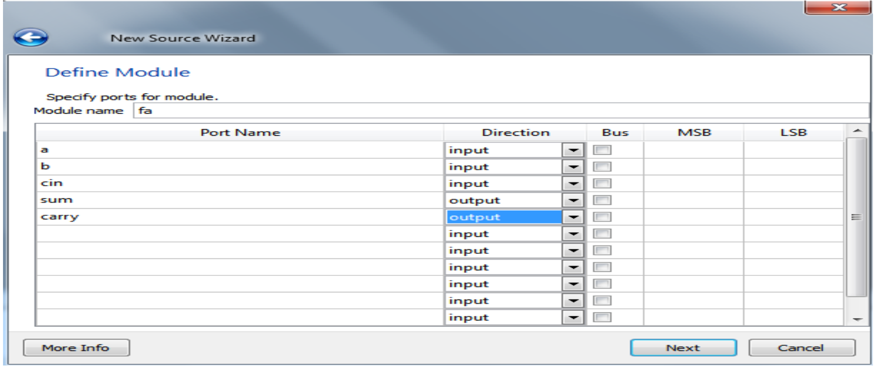
- IP (CORE Generator & Architecture Wizard)
- Schematic
- User Document
- Verilog Module
- Verilog Test Fixture
- VHDL Module
- VHDL Library
- VHDL Package
- VHDL Test Bench
- Embedded Processor

File name: fa

Location: H:\Full Adder\FA

☑ Add to project

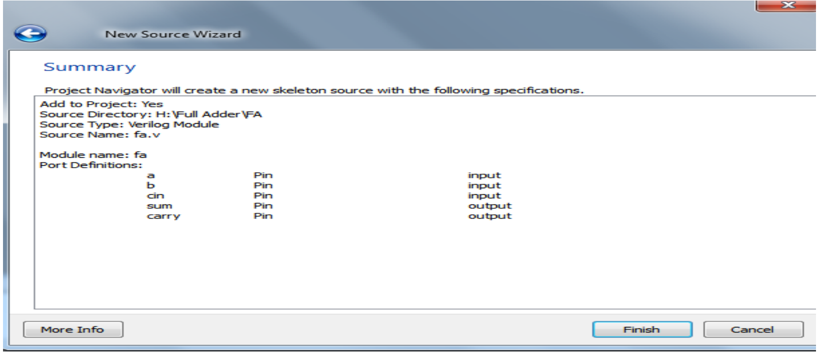
Select **Verilog Module** and in the "File Name:" Enter the name of the Project. Then click on **Next** to accept the entries. This pops up the following window.



Specify ports for module.
Module name: fa

Port Name	Direction	Bus	MSB	LSB
a	input	□		
b	input	□		
cin	input	□		
sum	output	□		
carry	output	□		
	input	□		
	input	□		
	input	□		
	input	□		
	input	□		
	input	□		

In the **Port Name** column, enter the names of all input and output pins and specify the **Direction** accordingly. A Vector/Bus can be defined by entering appropriate bit numbers in the **MSB/LSB** columns. Then click on **Next>**to get a window showing all the new source information.

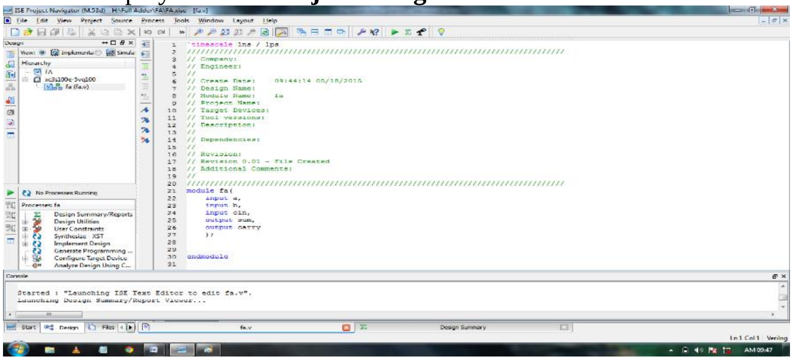


Project Navigator will create a new skeleton source with the following specifications.

```
Add to Project: Yes
Source Directory: H:\Full Adder\FA
Source Type: Verilog Module
Source Name: fa.v

Module name: fa
Port Definitions:
                  a          Pin                  input
                  b          Pin                  input
                  cin        Pin                  input
                  sum        Pin                  output
                  carry      Pin                  output
```

click on **Finish** to continue.

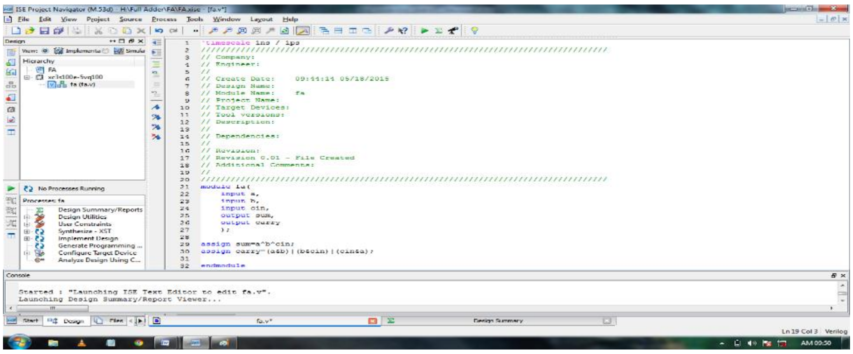
The source file will now be displayed in the **Project Navigator** window.



```
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Company:
// Engineer:
//
// Create Date:    09:44:14 05/18/2015
// Design Name:
// Module Name:    fa
// Project Name:
// Target Devices:
// Tool versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////
module fa(
    input a,
    input b,
    input cin,
    output sum,
    output carry
    );


endmodule
```

```
Started : "Launching ISE Text Editor to edit fa.v".
Launching Design Summary/Report Viewer...
```

The source file window can be used as a text editor to make any necessary changes to the source file. All the input/output pins will be displayed. Save your Verilog program periodically by selecting the **File->Save** from the menu. You can also edit Verilog programs in any text editor and add them to the project directory using "Add Copy Source".

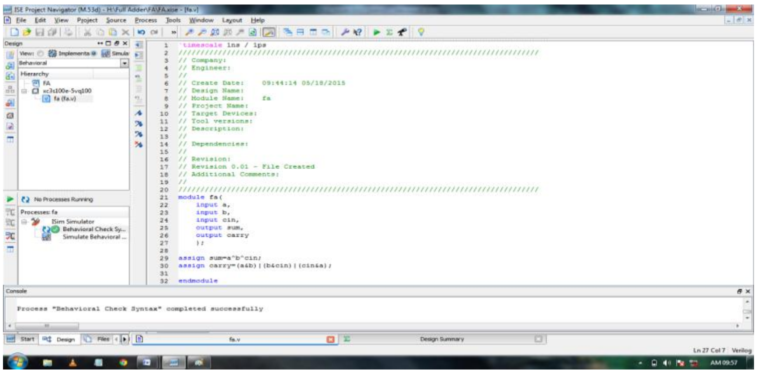


```
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Company:
// Engineer:
//
// Create Date:    09:44:14 05/18/2015
// Design Name:
// Module Name:    fa
// Project Name:
// Target Devices:
// Tool versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////
module fa(
    input a,
    input b,
    input cin,
    output sum,
    output carry
    );

assign sum=a^b^cin;
assign carry=(a&b)|(b&cin)|(cin&a);

endmodule
```

```
Started : "Launching ISE Text Editor to edit fa.v".
Launching Design Summary/Report Viewer...
```

Simulating and Viewing the Output Waveforms:

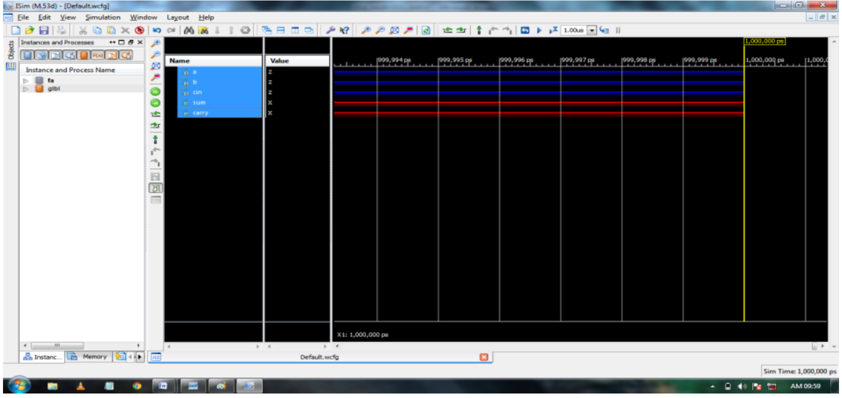
Click on simulation select the existing file and expand ISim Simulator and click on Behavioral check syntax to check the Errors.



```
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Company:
// Engineer:
//
// Create Date:    09:44:14 05/18/2015
// Design Name:
// Module Name:    fa
// Project Name:
// Target Devices:
// Tool versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////
module fa(
    input a,
    input b,
    input cin,
    output sum,
    output carry
    );

assign sum=a^b^cin;
assign carry=(a&b)|(b&cin)|(cin&a);

endmodule
```

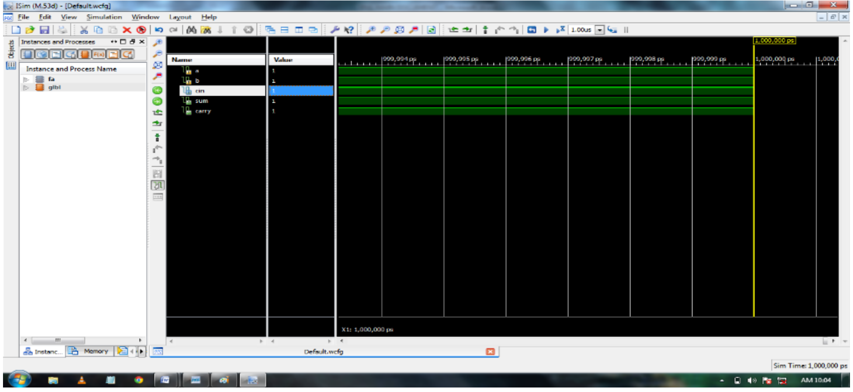
```
Process "Behavioral Check Syntax" completed successfully
```

If there are no errors click on simulate behavioral model. A window pop up is opened.



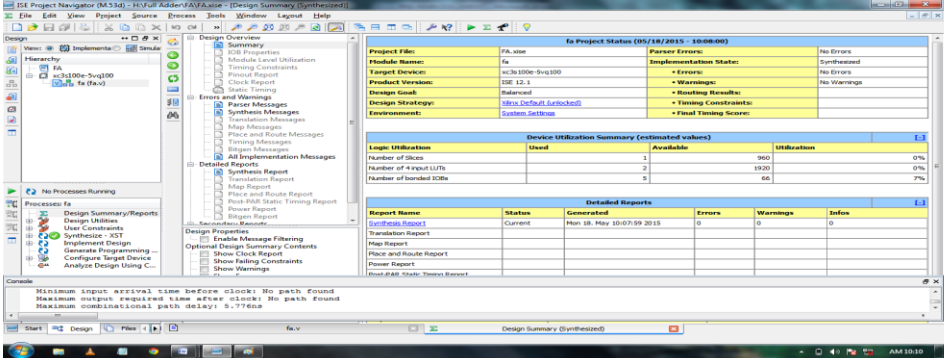
Here we can give the inputs. Right click on the selected input click on force constant and enter the input value click on Ok.

Click on Run option in the tool bar to check input and output waveforms.



Synthesis and Implementation of the Design:

Click on Implementation select the existing file and double click on Synthesize-XST. If there are errors correct it. If there are no errors click on design summary and reports.



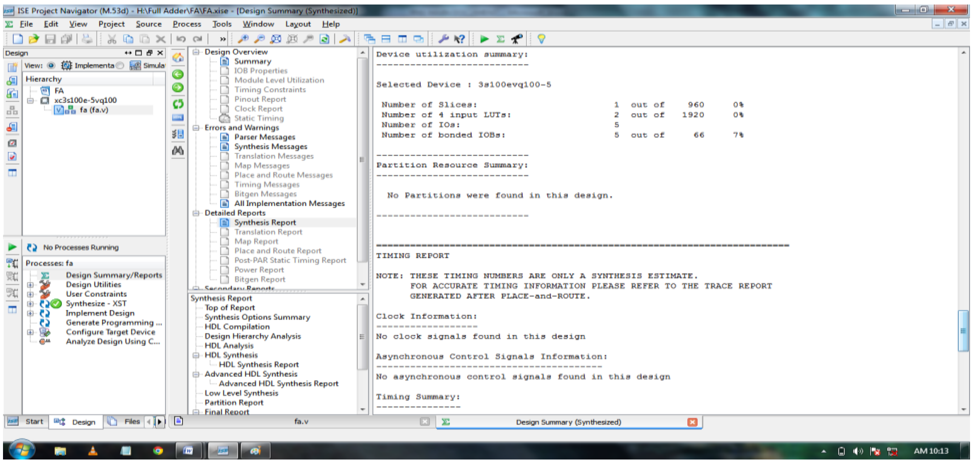
fa Project Status (05/18/2015 - 10:08:00)			
Project File:	FA.xise	Parser Errors:	No Errors
Module Name:	fa	Implementation State:	Synthesized
Target Device:	xc3s100e-5vq100	• Errors:	No Errors
Product Version:	ISE 12.1	• Warnings:	No Warnings
Design Goal:	Balanced	• Routing Results:	
Design Strategy:	Xilinx Default (unlocked)	• Timing Constraints:	
Environment:	System Settings	• Final Timing Score:	

Device Utilization Summary (estimated values)			
Logic Utilization	Used	Available	Utilization
Number of Slices	1	960	0%
Number of 4 input LUTs	2	1920	0%
Number of bonded IOBs	5	66	7%

Detailed Reports					
Report Name	Status	Generated	Errors	Warnings	Infos
Synthesis Report	Current	Mon 18. May 10:07:59 2015	0	0	0
Translation Report					
Map Report					
Place and Route Report					
Power Report					
Post-PAR Static Timing Report					

```
Minimum input arrival time before clock: No path found
Maximum output required time after clock: No path found
Maximum combinational path delay: 5.776ns
```

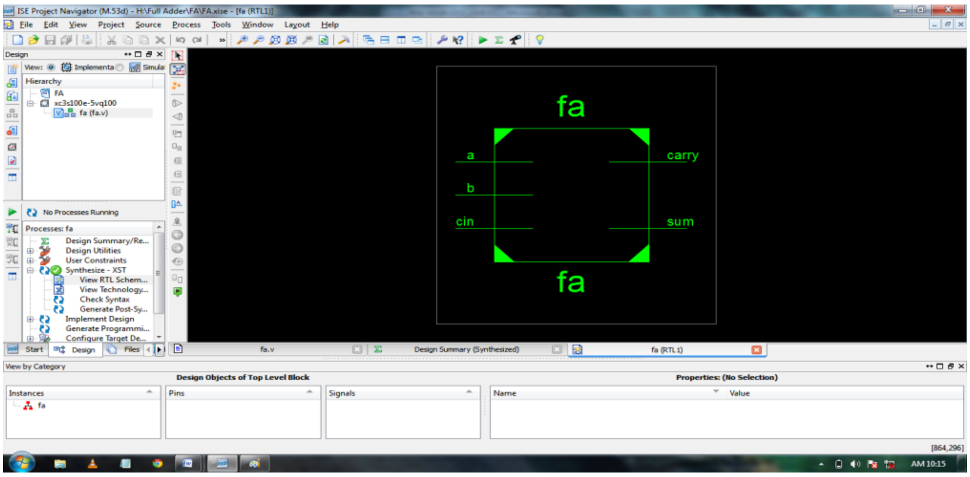
Open the Synthesis Report in the Detailed Reports to see the Device utilization Summary and Timing Report of the current project.



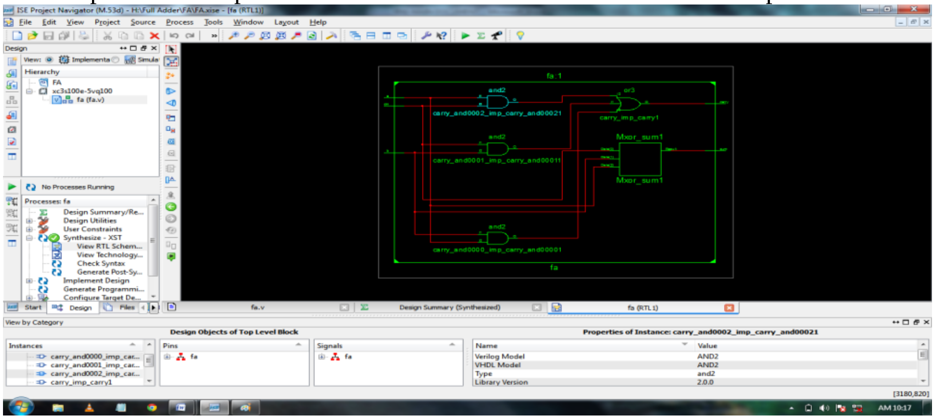
```
Device utilization summary:
---------------------------

Selected Device : 3s100evq100-5

 Number of Slices:                       1  out of    960     0%
 Number of 4 input LUTs:                 2  out of   1920     0%
 Number of IOs:                          5
 Number of bonded IOBs:                  5  out of     66     7%

---------------------------
Partition Resource Summary:
---------------------------

  No Partitions were found in this design.

---------------------------


=========================================================================
TIMING REPORT

NOTE: THESE TIMING NUMBERS ARE ONLY A SYNTHESIS ESTIMATE.
      FOR ACCURATE TIMING INFORMATION PLEASE REFER TO THE TRACE REPORT
      GENERATED AFTER PLACE-and-ROUTE.

Clock Information:
------------------
No clock signals found in this design

Asynchronous Control Signals Information:
----------------------------------------
No asynchronous control signals found in this design

Timing Summary:
---------------
```

View RTL Schematic:

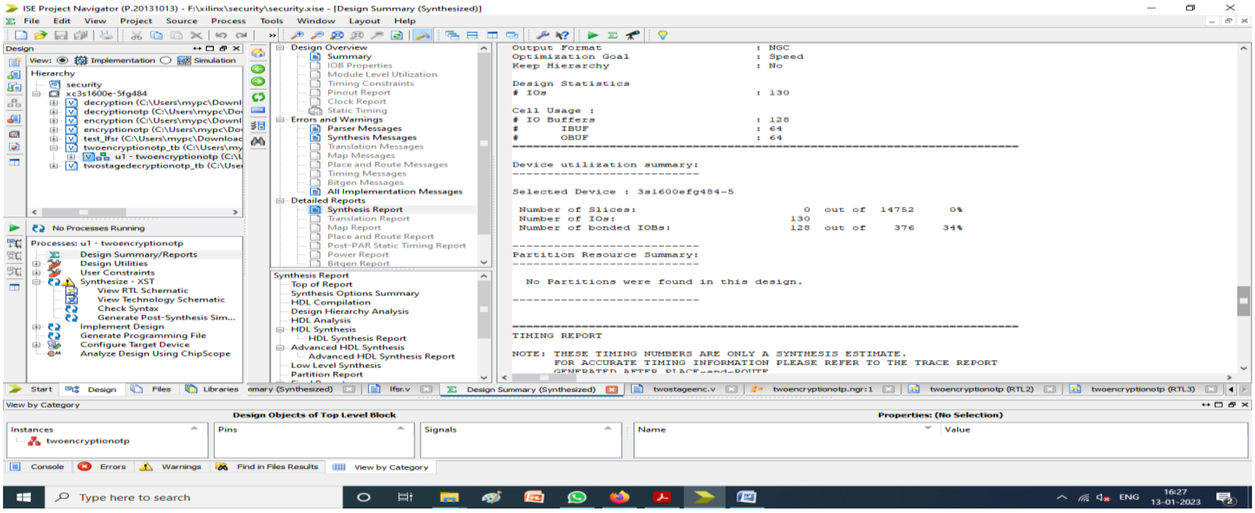
Expand Synthesize-XST and click on view RTL Schematic and click ok.



The window with Top module is opened to view the internal modules click on the top module.


ISE Project Navigator (M.53d) - H:\Full Adder\FA\FA.xise - [fa (RTL1)]
fa:1
and2
or3
carry_and0002_imp_carry_and00021
carry_imp_carry1
Mxor_sum1
carry_and0001_imp_carry_and00011
carry_and0000_imp_carry_and00001
Design Objects of Top Level Block
Properties of Instance: carry_and0002_imp_carry_and00021
Instances
carry_and0000_imp_car...
carry_and0001_imp_car...
carry_and0002_imp_car...
carry_imp_carry1
Name
Value
Verilog Model
AND2
VHDL Model
AND2
Type
and2
Library Version
2.0.0
AM 10:17

V. SIMULATION RESULTS


ISE Project Navigator (P.20131013) - F:\xilinx\security\security.xise - [Design Summary (Synthesized)]
File Edit View Project Source Process Tools Window Layout Help
Design
View: Implementation Simulation
Hierarchy
security
xc3s1600e-5fg484
decryption (C:\Users\mypc\Downl
decryptionotp (C:\Users\mypc\Do
encryption (C:\Users\mypc\Downl
encryptionotp (C:\Users\mypc\Do
test_lfsr (C:\Users\mypc\Downloa
twoencryptionotp_tb (C:\Users\my
u1 - twoencryptionotp (C:\U
twostagedecryptionotp_tb (C:\Use
No Processes Running
Processes: u1 - twoencryptionotp
Design Summary/Reports
Design Utilities
User Constraints
Synthesize - XST
View RTL Schematic
View Technology Schematic
Check Syntax
Generate Post-Synthesis Sim...
Implement Design
Generate Programming File
Configure Target Device
Analyze Design Using ChipScope
Design Overview
Summary
IOB Properties
Module Level Utilization
Timing Constraints
Pinout Report
Clock Report
Static Timing
Errors and Warnings
Parser Messages
Synthesis Messages
Translation Messages
Map Messages
Place and Route Messages
Timing Messages
Bitgen Messages
All Implementation Messages
Detailed Reports
Synthesis Report
Translation Report
Map Report
Place and Route Report
Post-PAR Static Timing Report
Power Report
Bitgen Report
Synthesis Report
Top of Report
Synthesis Options Summary
HDL Compilation
Design Hierarchy Analysis
HDL Analysis
HDL Synthesis
HDL Synthesis Report
Advanced HDL Synthesis
Advanced HDL Synthesis Report
Low Level Synthesis
Partition Report
Start Design Files Libraries
mary (Synthesized) lfsr.v Design Summary (Synthesized) twostageenc.v twoencryptionotp.ngr:1 twoencryptionotp (RTL2) twoencryptionotp (RTL3)
View by Category
Design Objects of Top Level Block
Properties: (No Selection)
Instances
twoencryptionotp
Pins
Signals
Name
Value
Console Errors Warnings Find in Files Results View by Category
Type here to search
ENG
16:27
13-01-2023

```
Output Format                      : NGC
Optimization Goal                  : Speed
Keep Hierarchy                     : No

Design Statistics
# IOs                              : 130

Cell Usage :
# IO Buffers                       : 128
#      IBUF                        : 64
#      OBUF                        : 64
=========================================================================

Device utilization summary:
---------------------------

Selected Device : 3s1600efg484-5

 Number of Slices:                        0  out of  14752     0%
 Number of IOs:                         130
 Number of bonded IOBs:                 128  out of    376    34%

---------------------------
Partition Resource Summary:
---------------------------

  No Partitions were found in this design.

---------------------------


=========================================================================
TIMING REPORT

NOTE: THESE TIMING NUMBERS ARE ONLY A SYNTHESIS ESTIMATE.
      FOR ACCURATE TIMING INFORMATION PLEASE REFER TO THE TRACE REPORT
      GENERATED AFTER PLACE-and-ROUTE
```

Fig. Area Report

VI. CONCLUSION AND FUTURE WORK

Electronic voting machines are primary for elections in college level, panchayat or state, country and international levels to maintain integrity in the entire process. We have considered 3 contestants and their voters are stored in separate registers and the total votes for all contestants are also counted and stored for final analysis in the election system. Security can be provided using a 6-bit OTP generation process using pseudo random binary sequence generator. Thus, the process provides a privacy for voters, safe and secure conduction of elections in all levels.

REFERENCES

[1]. Smith, T.F., Waterman, M.S.: Identification of Common Molecular Subsequences. J. Mol. Biol. 147, 195–197 (1981) .

[2]. May, P., Ehrlich, H.C., Steinke, T.: ZIB Structure Prediction Pipeline: Composing a Complex Biological Workflow through Web Services. In: Nagel, W.E., Walter, W.V., Lehner, W. (eds.) Euro-Par 2006. LNCS, vol. 4128, pp. 1148–1158. Springer, Heidelberg (2006)

[3]. Czajkowski, K., Fitzgerald, S., Foster, I., Kesselman, C.: Grid Information Services for Distributed Resource Sharing. In: 10th IEEE International Symposium on High Performance Distributed Computing, pp. 181–184. IEEE Press, New York (2001)

[4]. Foster, I., Kesselman, C., Nick, J., Tuecke, S.: The Physiology of the Grid: an Open Grid Services Architecture for Distributed Systems Integration. Technical report, Global Grid Forum (2002) .

[5]. N.S.N. LAKSHMI PATHI RAJU, A.PRAVIN, N.S.MURTHY SHARMA, S.S.KIRAN A Novel Proposal On Implementation Of Polling Percentage Improvement System Through Embedded Based Integration Of electronic Voting Machine And Other Methodologies – IJ ETA ETS ISSN: 0974-3588 | JULY "12 – DECEMBER "12 | Volume 5 : Issue 2 pp 1-5

[6]. Jeremy Clark, Aleks Essex and Carlisle Adams "Secure and Observable Auditing of Electronic Voting Systems using Stock Indices" 0840-7789/07/$25.00 ©2007 IEEE

[7]. Leyou Zhang, Yupu Hu, Xu'an Tian and Yang Yang "Novel Identitybased Blind Signature for ElectronicVoting System" 978-0-7695-3987-4/10 $26.00 © 2010 IEEE DOI 10.1109/ETCS.2010.198 4.

[8]. Ying Qiu and Huafei Zhu "Somewhat Secure Mobile Electronic-voting Systems Based on theCut-and-Choose Mechanism" 978-0-7695-3931-7/09 $26.00 © 2009 IEEE DOI 10.1109/CIS.2009.39

[9]. Arunkumar.N and Arunkumar.P.L "Analysis, Design & Real-Time Implementation of Electronic Voting Machine". 6. J. Paul Gibson, Eric Lallet, and JeanLuc Raffy "Engineering a Distributed e-Voting System Architecture: Meeting Critical Requirements" H. Giese (Ed.): ISARCS 2010, LNCS 6150, pp. 89– 108, 2010.c_Springer-Verlag Berlin Heidelberg 2010.

[10]. Rui Joaquim, André Zúquete and Paulo Ferreira "REVS – A ROBUST ELECTRONIC VOTING SYSTEM" 8. Mohammad Hajjar, Bassam Daya, Anis Ismail and Haissam Hajjar "AN E-VOTING SYSTEM FOR LEBANESE ELECTIONS" Journal of Theoretical and Applied