Implementation of Logic tegrated Circuit Tester System Based on FPGA Technology

Raghad Z. Tawfiq* M.Sc. Firas M. Naif*

Received on: 18/12/2005
Accepted on: 14/5/2006

Abstruct

Field programmable gate arrays (FPGAs) are integrated circuits which can be user customized to implement arbitrary digital functions. In this paper, we describe a typical TTL Logic Integrated Circuit (IC) tester system design based on FPGA technology. IC testers are traditionally implemented in a microprocessor. Compared with conventional technologies FPGA allowed designers to create large designs, test them and make modifications very easily and quickly.

The implementation of the IC tester system have been done by using very high speed hardware descriptive language (VHDL) and it's implementation on filed programmable gate array (FPGA) by using the synthesis tools of (Xilinx foundation series 4.1i program), which is used to implement the reconfigurable hardware design of the IC tester system. The final design is prototyped into the prototyping board for experimental testing.

Keywords:
IC Tester, TTL, Field Programmable Gate array (FPGA), VHDL

الخلاصة

أن رقاقة مصفوفات بوابات المجال المبرمج (FPGA) هي عبارة عن مجموعة دوائر متكاملة لها القدرة على بناء دوال منطقية مختلفة. في هذا البحث تم تصميم و تطبيق جهاز لفحص الدوائر المتكاملة المنطقية TTL (Integrated Logic Circuit) بالاعتماد على تقنية رقاقة مصفوفات بوابات المجال المبرمج (FPGA).قديما كان جهاز فحص الدوائر الألكترونية المتكاملة يعتمد على نوع من انواع المعالجات الدقيقة (Microprocessors) في تصميمه ، مقارنة بالامكانيات التي وفرتها تقنية رقاقة مصفوفات بوابات المجال المبرمج نجد ان التصميم اصبح قابل للتطوير و الفحص و اسهل و اسرع في الانجاز.

تم تصميم جهاز فحص الدوائر الألكترونية المتكاملة بأستخدام لغة وصف الدوائر الفائـــقة السرعة (VHDL) و ذلك بأستخدام برنامج (Active VHDL Ver. 3.5) في حين تمت عملية التركيب و التنفـــيذ في رقاقة مصفوفات بوابات المجال المبرمج بأستخدام برنامج (Xilinx foundation series 4.1i program). ومن ثم تم ربط التصميم النهائي ليكون جاهزاً للأختبار العملي.

*Control and system Dept, University of Technology Baghdad- Iraq

1. Introduction

Field-Programmable Gate Array (FPGA) is a programmable device capable of implementing complex digital system [1]. It is a uniform structured VLSI chip that can be configured and reconfigured for different designs [2,3].

Field Programmable Gate Array is a uigital device that, thanks to the possibility of implementing logic circuits by programming the required function, offer the benefits of low costs, short manufacturing turnaround time and easy design changes. As a result, most prototypes and many production designs are now implemented on FPGAs, making hardware implementation economically feasible even for those applications which were previously restricted to software implementation [4].One of the most suitable languages for the FPGA devices is the Hardware Description Language [2, 3, 5].

VHDL is an acronym for VHSIC hardware description language. VHSIC in turn stand for very high-speed integrated circuits. The original standard for VHDL was adopted in 1987; a revised standard was adopted in 1993 and called IEEE 1164[2, 6].

VHDL is designed to fill a number of needs in the design process. Firstly, it allows description of the structure of a design. Secondly, it allows the specification of the function of designs using familiar programming language forms. Thirdly, as a result, it allows a design to be simulated before being manufactured [7].

VHDL is a powerful hardware description language that can be used to model a digital system at many levels of abstraction, ranging from the algorithmic level to the gate level [5, 8].

In traditional FPGA design flow, a user enters a description of the desired circuit by using a hardware description language (HDL) such as VHDL. A CAD tool is used to synthesize a netlist from the HDL description. Another CAD tool then used to decompose the netlist to fit the logic resources of the FPGA, and then a place and route (P&R) tool is used. Following this, the logic and interconnection of the FPGA is defined, and the CAD tool outputs a bitstream, which can be downloaded to the FPGA's configuration RAM [9].

2. TTL Packaged Logic

TTL is a family of packaged logic components that dominated logic designs between its introductions in the 1960s up to the 1980s. It is still used to upgrade the performance of existing designs, as well as for certain niche applications. A TTL *integrated circuit* package typically contains from 10 to 100 gates. The Texas Instruments 74-series components provide the standard numbering scheme used by industry. For example, the name for a package containing four 2-input NAND gates is "7400," while a "7404" contains six (hex) inverters.

In TTL technology, logic gates are available in rectangular dual in-line packages, also known as "dips." Pins that connect internal logic to the outside are placed along the two long edges of the package, numbered anti counterclockwise starting with pin #1. A 14-pin package is shown in figure (1), along with a diagram of its internal logic and pin connectivity. The pin #7 is usually connected to ground (GND), while the pin #14 is connected to the power supply (VCC). A typical 14-pin

package measures approximately 0.75 inch in one dimension and 0.3 inch in the other. Some of the kinds of building blocks found in the TTL family are:
Quadruple 2- input NAND gate (7400)
Quadruple 2- input AND gate (7408)
Quadruple 2- input OR gate (7432)
Quadruple 2- input XOR gate (7486)
Hex inverter NOT gate (7404)

The TTL family also supports even larger aggregations of logic gates that provide higher-level functions.

TTL components operate across a range of voltages, from 5 V to 1.8 V. Some members of the family have been designed so that a 3.3 V component can communicate with a 1.8 V [10].

3. Prototype Design Phase

This phase represents the design and implementation of the IC tester prototype, and divided into two parts: the first part explain the system hardware design, and the second part concerned with system software design.

3.1 System Hardware Design

A complete IC tester prototype hardware design is represented briefly in the following sections. Figure (2) shows the schematic sheet for the complete designed system, while figure (3) shows a photo picture of it.

3.1.1 Power Supply

A power supply is designed to provide the system with the required voltage to operate the system. For the power supply design, a transformer is used to drop the 220-Volt AC to 12- Volt AC, the output is fed to a bridge circuit for rectification. The capacitor of $1000\,\mu$F is used for smoothing the DC voltage at 17-Volt, because the gain of the bridge is about 1.4. At the end, a regulator of type MC7805 is used to regulate the voltage to 5-Volt that is used to supply the system.

3.1.2 The Display Unit

The display unit consist of four seven-segment display to display the number of the IC to be tested and then display the result of testing as (FAIL) or (PASS).

The seven-segment display provides seven LEDs in one package to create numerical displays. Each LED called segment and all LEDs labeled (a, b, c, d, e, f, g) Each LED can turn **on** or **off** independently to represent one segment of a digit. By turning **on** selected segments, different number or character may be displayed.

In the design, each seven-segment will received the required signals from the output ports of the FPGA through a deriving circuit as shown in figure (2). The driving circuit for each seven-segment display consists of seven transistor of type BC107. The collector of each transistor is connected to the anode of each led in the seven-segment display through a 220 ohm resistor, and connected to the FPGA output port through a 3k ohm resisters to restrict the maximum current (and therefore the maximum brightness of the display).

3.1.3 Keypad and Keypad Encoder

The keypad is a three by three matrix of mechanical switches labeled from (0 to 9) and (*) key to be used as start-testing bottom, and (#) key is not used. To read a key if pressed it is needed to be denounced, i.e., to get ride of glitches associated with mechanical switches. A keypad encoder integrated circuit of type 74C922 is used for the design, while gives a valid key code after denouncing it and it is tire state output. Pin 12 of this encoder represent the data available signal (DA). When any key is pressed this pin will be logic 1, and this output signal will be as an input signal to the FPGA, so letter k will represent it in the VHDL program, while the pins from 14 to 17

3.1.4 FPGA Device

The IC tester system is designed by VHDL programming language and implemented using 4000 series family field programmable gate array as show in figure (2) especially XC4003E from this family which is an enhanced architecture based on the 4000 family, but offers many additional features.

The XC4003E supports system clock rate up to 70MHz and internal performance in excess of 150 MHz. The device is customized by loading configuration data into internal memory cells. The FPGA then can be either actively read its configuration data from an external serial or byte-parallel PROM (master mode), or the configurable data can be written into FPGA from an external device (slave and peripheral mode) [11].

The master parallel up mode is used by selecting <100> on the mode pins (M2, M1, M0) of the FPGA. The FPGA will directly connect to E^2PROM (X28HC64P) and the configuration of this type of FPGA family can be loaded while it is connected into the system.

3.2 System Software Design

The IC tester circuit has been designed and implemented using VHDL programming language throughout the using of FPGA technology. So the hardware description language (HDL) is used to describe this circuit at the behavioral level.

represents the key code and they are input signals to the FPGA. They are represented by a vector X(0:3) in the VHDL program. Figure (2) shows the keypad, keypad encoder, and their connections to the FPGA.

After written and compiled using Active- HDL program, the source program will be synthesized, tested, implemented and finally configured using Xilinx Foundation Series (4.1i) CAD software package consecutively.

In the present design the entity part include all the input and output ports of the FPGA represented by the line (y0, y1, y3, y4, y6, y7, y9, y10) and x(0:3) as input port and (y2, y5, y8, y11) and (s0(1:7), s1(1:7), s2(1:7), s3(1:7)) as output port and (k) as input port, and the architecture part will define and manage our proposed design.

The main program will first start by initialization of the input and output ports of the FPGA after that it will proceed to find out whether a key is pressed or not by checking the signal (k). If any key is pressed the FPGA will inter the key code, which represent the IC number and display it on the seven-segment display unit and the FPGA will wait for pressing the start-testing key (*). After this the program of the FPGA will recognize which IC should be tested depending on the number of IC interred and the result of testing will display on the four seven-segment display unit as (FAIL) if the IC is failed or (PASS) when their is no fault in the IC through the output ports s0(1:7), s1(1:7), s2(1:7), s3(1:7)) which connected to the display unit through a driving circuits .

The following program shows the behavioral VHDL code that describes the IC tester system:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use
IEEE.STD_LOGIC_ARITH.ALL;
useIEEE.STD_LOGIC_UNSIGNED.A
LL;
ENTITY ictester is PORT(
K:in STD_LOGIC;
X:in  STD_LOGIC_VECTOR(0  to
3);
s0,s1,s2,s3:out
STD_LOGIC_VECTOR(1 to 7);
Y0,Y1,Y3,Y4,Y6,Y7,Y9,Y10:out
STD_LOGIC);Y2,y5,Y8,Y11:in
STD_LOGIC);
END ictester;
ARCHITECTURE  Behavioral  OF
ictester is
BEGIN
PROCESS(K)
VARIABLE
D0,D1,D2,D3:STD_LOGIC_VECTOR
(0 to 3 );
VARIABLE
N:STD_LOGIC_VECTOR(0 to 3);
VARIABLE  FF:INTEGER  RANGE  0
to 4 := 0;
begin
if (K'event and K='1') then
      if x/="1111" then
           ff:=ff+1;
      if ff=1 then
case x is
when "0000"=>s0 <="1111110";
when "0001"=>s0 <="0110000";
When "0010"=>s0 <="1101101";
when "0011"=>s0 <="1111001";
when "0100"=>s0 <="0110011";
when "0101"=>s0 <="1011011";
when "0110"=>s0 <="1011111";
when "0111"=>s0 <="1110000";
when "1000"=>s0 <="1111111";
when "1001"=>s0 <="1110011";
when "1010"=>s0 <="1110111";
when "1011"=>s0 <="0011111";
when "1100"=>s0 <="1001110";
when "1101"=>s0 <="0111101";
when "1110"=>s0 <="1001111";
when "1111"=>s0 <="1000111";
when others=>s0 <="0000000";
end case;
D0:=X;
      end if;
if ff=2 then
case x is
when "0000"=>s1 <="1111110";
when "0001"=>s1 <="0110000";
when "0010"=>s1 <="1101101";
when "0011"=>s1 <="1111001";
when "0100"=>s1 <="0110011";
when "0101"=>s1 <="1011011";
when "0110"=>s1 <="1011111";
when "0111"=>s1 <="1110000";
when "1000"=>s1 <="1111111";
when "1001"=>s1 <="1110011";
when "1010"=>s1 <="1110111";
when "1011"=>s1 <="0011111";
when "1100"=>s1 <="1001110";
when "1101"=>s1 <="0111101";
when "1110"=>s1 <="1001111";
when "1111"=>s1 <="1000111";
when others=>s1 <="0000000";
end case;
D1:=X;
   end if;
      if ff=3 then
case x is
when "0000"=>s2 <="1111110";
when "0001"=>s2 <="0110000";
when "0010"=>s2 <="1101101";
when "0011"=>s2 <="1111001";
when "0100"=>s2 <="0110011";
when "0101"=>s2 <="1011011";
when "0110"=>s2 <="1011111";
when "0111"=>s2 <="1110000";
when "1000"=>s2 <="1111111";
when "1001"=>s2 <="1110011";
when "1010"=>s2 <="1110111";
when "1011"=>s2 <="0011111";
when "1100"=>s2 <="1001110";
when "1101"=>s2 <="0111101";
when "1110"=>s2 <="1001111";
when "1111"=>s2 <="1000111";
when others=>s2 <="0000000";
end case;
D2:=X;
end if;
      if ff=4 then
case x is
when "0000"=>s3 <="1111110";
when "0001"=>s3 <="0110000";
when "0010"=>s3 <="1101101";
when "0011"=>s3 <="1111001";
when "0100"=>s3 <="0110011";
when "0101"=>s3 <="1011011";
when "0110"=>s3 <="1011111";
when "0111"=>s3 <="1110000";
when "1000"=>s3 <="1111111";
when "1001"=>s3 <="1110011";
when "1010"=>s3 <="1110111";
when "1011"=>s3 <="0011111";
when "1100"=>s3 <="1001110";
when "1101"=>s3 <="0111101";
when "1110"=>s3 <="1001111";
when "1111"=>s3 <="1000111";
when others=>s3 <="0000000";
end case;
D3:=X;
ff:=0;
   end if;
      else
-----Check AND 7408---------
IF D2&D3="00001000" THEN
Y0<='0';
Y1<='0';
Y3<='0';
Y4<='0';
Y6<='0';
Y7<='0';
Y9<='0';
Y10<='0';
```

```
N:=Y2&Y5&Y8&Y11;
IF N/="0000" THEN
s0<="1000111";
s1<="1110111";
s2<="0110000";
s3<="0001110";
ELSE
Y0<='1';
Y1<='0';
Y3<='1';
Y4<='0';
Y6<='1';
Y7<='0';
Y9<='1';
Y10<='0';
N:=Y2&Y5&Y8&Y11;
END IF;
IF N/="0000" THEN
s0<="1000111";
s1<="1110111";
s2<="0110000";
s3<="0001110";
ELSE
Y0<='0';
Y1<='1';
Y3<='0';
Y4<='1';
Y6<='0';
Y7<='1';
Y9<='0';
Y10<='1';
N:=Y2&Y5&Y8&Y11;
END IF;
IF N/="0000" THEN
s0<="1000111";
s1<="1110111";
s2<="0110000";
s3<="0001110";
ELSE
Y0<='1';
Y1<='1';
Y3<='1';
Y4<='1';
Y6<='1';
Y7<='1';
```

```
Y9<='1';
Y10<='1';
N:=Y2&Y5&Y8&Y11;
END IF;
IF N/="1111" THEN
s0<="1000111";
s1<="1110111";
s2<="0110000";
s3<="0001110";
ELSE
s0<="1100111";
tatement  of  checking  NAND
7400

----------------------------
             :
             :
------Check XOR 7486--------

Statement  of  checking  XOR
7486

-------Check OR 7432--------

IF D2&D3="00110010" THEN
Y0<='0'
Y1<='0';
Y3<='0';
Y4<='0';
Y6<='0';
Y7<='0';
Y9<='0';
Y10<='0';
N:=Y2&Y5&Y8&Y11;
IF N/="0000" THEN
s0<="1000111";
s1<="1110111";
s2<="0110000";
s3<="0001110";
ELSE
Y0<='1';
Y1<='0';
Y3<='1';
Y4<='0';
Y6<='1';
```

```
s1<="1110111";
s2<="1011011";
s3<="1011011";
END IF;
----------------------------
             :
             :

----Check NAND 7400---------

S
Y7<='0';
Y9<='1';
Y10<='0';
N:=Y2&Y5&Y8&Y11;
END IF;
IF N/="0000" THEN
s0<="1000111";
s1<="1110111";
s2<="0110000";
s3<="0001110";
ELSE
Y0<='0';
Y1<='1';
Y3<='0';
Y4<='1';
Y6<='0';
Y7<='1';
Y9<='0';
Y10<='1';
N:=Y2&Y5&Y8&Y11;
END IF;
IF N/="0000" THEN
s0<="1000111";
s1<="1110111";
s2<="0110000";
s3<="0001110";
ELSE
Y0<='1';
Y1<='1';
Y3<='1';
Y4<='1';
Y6<='1';
Y7<='1';
Y9<='1';
```

```
Y10<='1';
N:=Y2&Y5&Y8&Y11;
END IF;
IF N/="1111" THEN
s0<="1000111";
s1<="1110111";
s2<="0110000";
s3<="0001110";
ELSE
s0<="1100111";
s1<="1110111";
s2<="1011011";
s3<="1011011";
END IF;
     END IF;
     end if;
     end if;
END IF;
END PROCESS;
END Behavioral;
```

In the second step a synthesis tools are used to fabricate and optimize the design described with the VHDL language. Synthesis is the process of converting a behavioral description of a design using hardware description language (VHDL) into gate level netlist. Netlist is a file that is generated by the synthesis tools to implement the functionality of the design described by the designer. Using this synthesized circuit of the design, it is then targeted toward a specific technology library, which uses the components available in the library to map the components required in the design with the ones that are mapped to the circuit file generated by the synthesis tool, these components are placed and routed.

The next step is the design implementation. This step is concerned with the exact selection of the circuit primitive and their placement and routing as shown in figure (4). After the completion of implementation step the floor planner can be used to know more information about the design connectivity and resource requirements, and the design mapping via location constraints, as shown in figure (5). Also a table of xilinx XC4003E pad specification is generated that shows all the signal names and the equivalent pin numbers as shown in figure (6). After this step, the programming step is began to generate a programming file, which specifies the operation of the FPGA device. The data of this file is stored in an E^2PROM. When the power is applied to the FPGA chip, the storage cells are loaded automatically from the memory. Figure (7) shows the process manager after completing design synthesis and implementation successfully.

4.Testing Phase

When the (hardware and software) implementation of the IC tester system is completed, the testing phase begins.

To test the design, first we use this system to test the 7408 IC. The IC will be placed on the IC base then the number of IC is interred through the keypad so that the number will be displayed on the display unit. Pressing the start-testing key (*) will start the testing operation, the result of testing will be displayed as FAIL or PASS according to the testing operation.

Figure (8) shows the photo picture of the 7408 IC testing case. While figure (9) show the photo picture of the result when the IC tested is PASS (there is no error in the IC), and figure (10) shows the photo picture of the result when the IC tested is FAIL (there is an error in the IC). Figure (11) shows the photo picture of the 7432 IC testing case, while figure (12) show the photo picture of the 7486 IC testing case.

5. Conclusion

In this paper, a complete design for a IC tester system has been made based on the FPGA technology.

Using FPGA technology in our design has enabled us to replace the microprocessor used in a conventional design, moreover it is easy to modify or add to your design and then reprogram the FPGA. Also the low cost of implementation and short time needed to physically realize our design using this technology, provide us with enormous advantages over traditional approaches for building prototype hardware which in turn leads to improve the performance of the system.

The FPGAs have provided a fast development cycle, and the flexibility to alter the design for bug fixes and functional enhancements.

6. References

[1] T. S. Czajkowski, "*A Synthesis Oriented Omniscient Manual Editor For FPGA Design*", M.Sc. Thesis, Toronto University, 2004.

[2] S. Brown and Z. Vranesic, "*Fundamentals of Digital Logic with VHDL Design*", McGraw-Hill Company, 2000.

[3] J. F. Wakerlay, "*Digital Design: Principles and Practices*", Prentice Hall, 2000.

[4] S. D. Brown, J. Rose, "*FPGA and CPLD Architectures: A Tutorial*", IEEE Design and Test of Computers, Vol. 13, No. 2, pp. 42-57, 1996.

[5] J. Bhasker, "*A VHDL Primer*", Prentice Hall PTR, third edition, 1999.

[6] M. Zwolinski, "*Digital System Design With VHDL*", Prentice Hall Inc., 2000.

[7] P. J. Ashenden, "*The VHDL Cookbook*", First Edition , 1990.

[8] F. Scarpino, "*VHDL And AHDL Digital System Implementation*", Prentice Hall, Inc., 1998.

[9] A. DeHon, "The density advantage of configurable computing," *Computer*, vol. 33, no. 4, pp. 41–49, 2000.

[10] R. H. Katz, "*Contemporary Logic Design*", Proctor-Willenbacher, Book, pp. 83-138, 2000.

[11] Xilinx Co., "*XC4003e and XC400X Series Field Programmable Gate Array data Book*", 2000

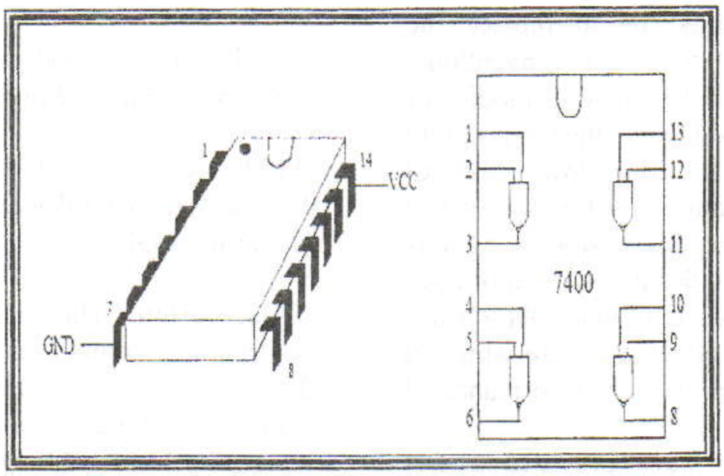


Figure (1) 14-pin package.

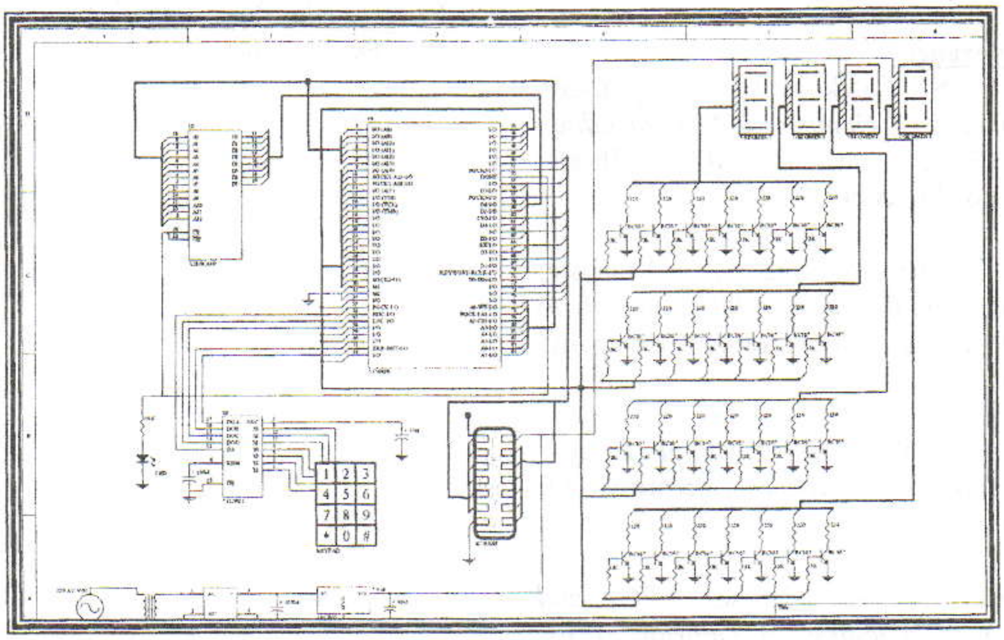


Figure (2) The schematics sheet for the complete designed system.

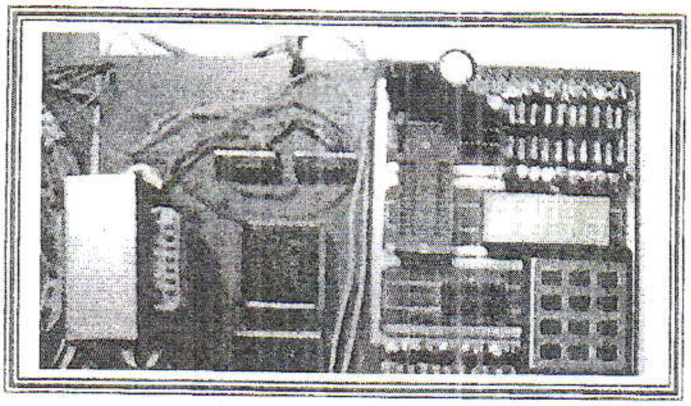
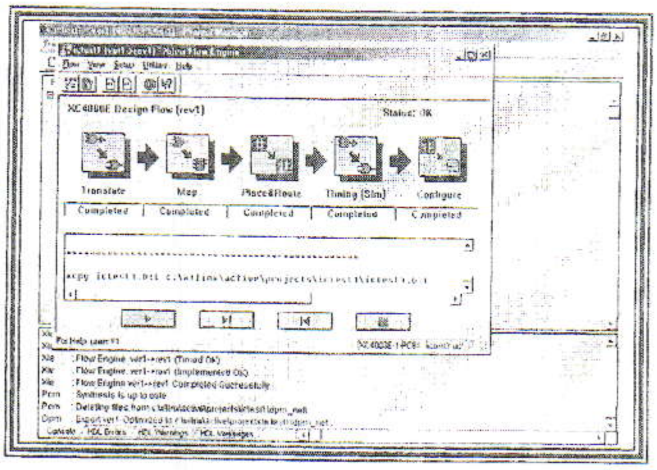


Figure (3) Photo picture for the complete system.



```
xcpy itest1.bit c:\xilinx\active\projects\itest1\itest1.bit
```

```
Xle  :Flow Engine ver1->rev1 (Timed OK)
Xle  :Flow Engine ver1->rev1 (Implemented OK)
Xle  :Flow Engine ver1->rev1 Completed Successfully
Prim :Synthesis is up to date
Prim :Deleting files from c:\xilinx\active\projects\itest1\lpm_net
Opm  :Export ver1 Optimized to c:\xilinx\active\projects\itest1\lpm_net
```

Figure (4) Design flow implementation.

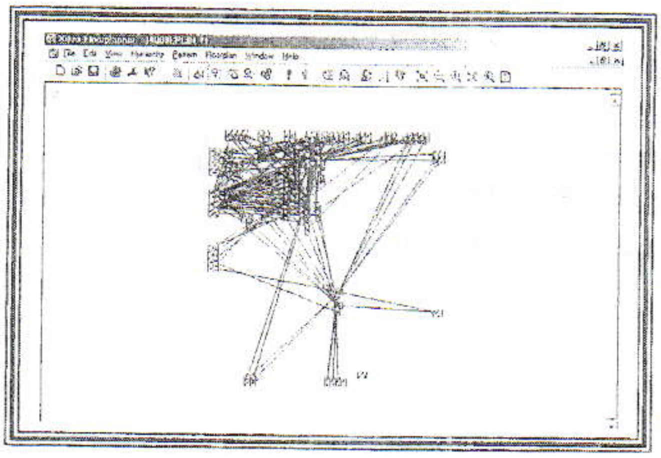
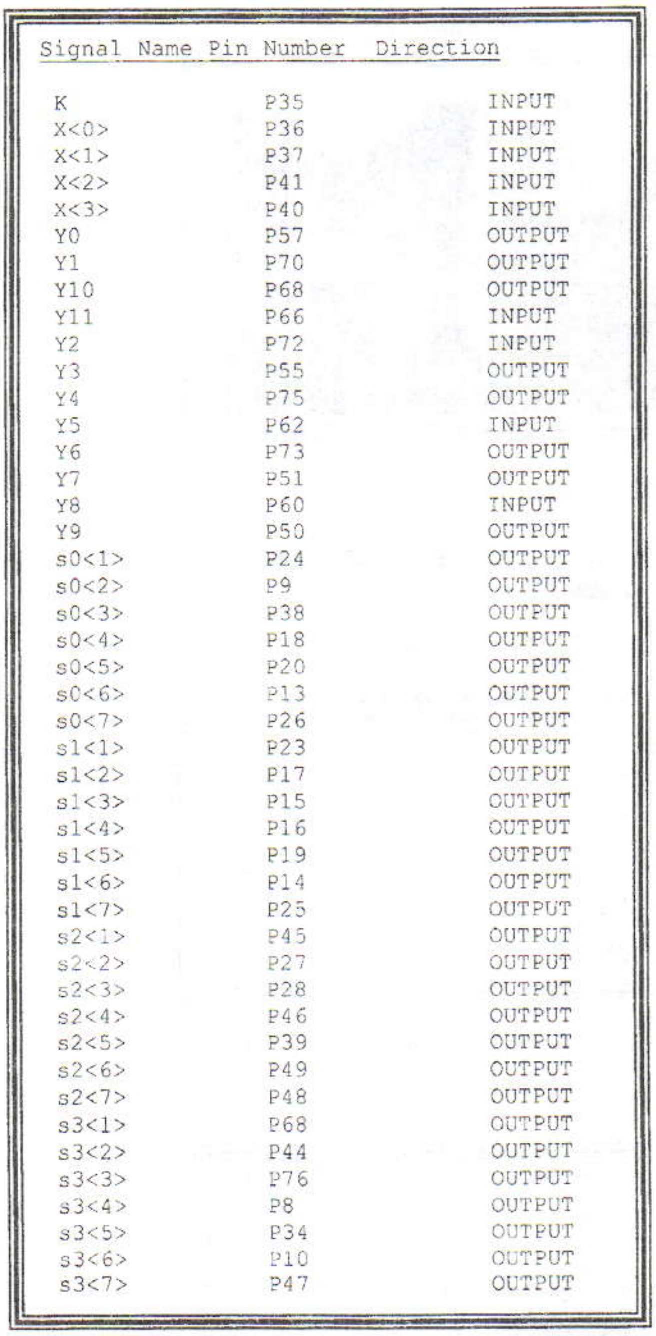


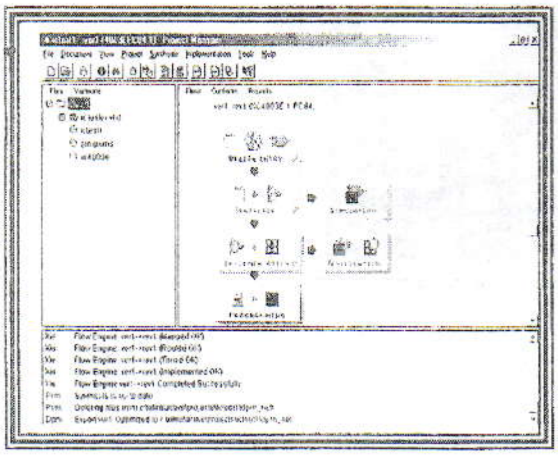
Figure (5) The floor planner.



```
Signal Name Pin Number  Direction

K            P35         INPUT
X<0>         P36         INPUT
X<1>         P37         INPUT
X<2>         P41         INPUT
X<3>         P40         INPUT
Y0           P57         OUTPUT
Y1           P70         OUTPUT
Y10          P68         OUTPUT
Y11          P66         INPUT
Y2           P72         INPUT
Y3           P55         OUTPUT
Y4           P75         OUTPUT
Y5           P62         INPUT
Y6           P73         OUTPUT
Y7           P51         OUTPUT
Y8           P60         INPUT
Y9           P50         OUTPUT
s0<1>        P24         OUTPUT
s0<2>        P9          OUTPUT
s0<3>        P38         OUTPUT
s0<4>        P18         OUTPUT
s0<5>        P20         OUTPUT
s0<6>        P13         OUTPUT
s0<7>        P26         OUTPUT
s1<1>        P23         OUTPUT
s1<2>        P17         OUTPUT
s1<3>        P15         OUTPUT
s1<4>        P16         OUTPUT
s1<5>        P19         OUTPUT
s1<6>        P14         OUTPUT
s1<7>        P25         OUTPUT
s2<1>        P45         OUTPUT
s2<2>        P27         OUTPUT
s2<3>        P28         OUTPUT
s2<4>        P46         OUTPUT
s2<5>        P39         OUTPUT
s2<6>        P49         OUTPUT
s2<7>        P48         OUTPUT
s3<1>        P68         OUTPUT
s3<2>        P44         OUTPUT
s3<3>        P76         OUTPUT
s3<4>        P8          OUTPUT
s3<5>        P34         OUTPUT
s3<6>        P10         OUTPUT
s3<7>        P47         OUTPUT
```

Figure (6) The pad report.


File Document View Project Synthesis Implementation Tools Help
Files Versions
Flow Contents Reports

```
Flow Engine: xert->next (Mapped OK)
Flow Engine: xert->next (Routed OK)
Flow Engine: xert->next (Timed OK)
Flow Engine: xert->next (Implemented OK)
Flow Engine: xert->next Completed Successfully
Synthesis is up to date
[illegible]
[illegible]
```

Figure (7) The process manager after completing design synthesis and implementation.

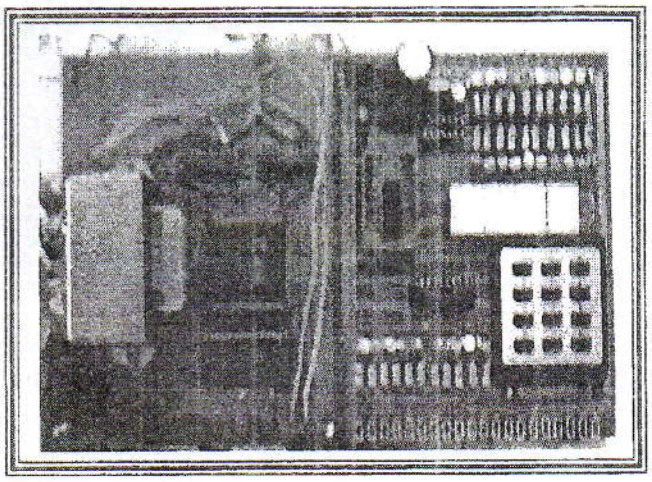


Figure (8) Photo picture of the 7408 IC.

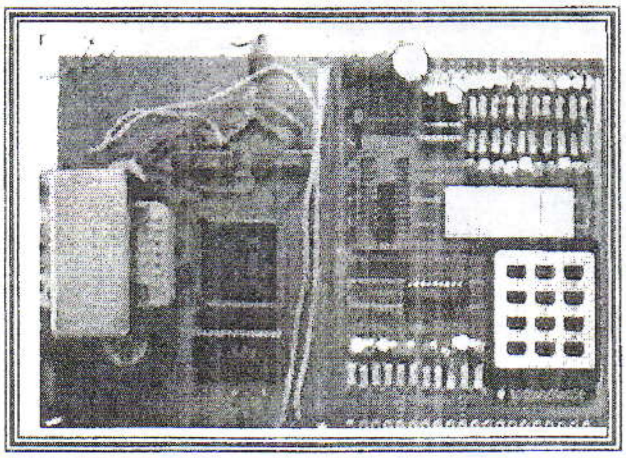


Figure (9) Photo picture of the result when the IC tested is PASS.

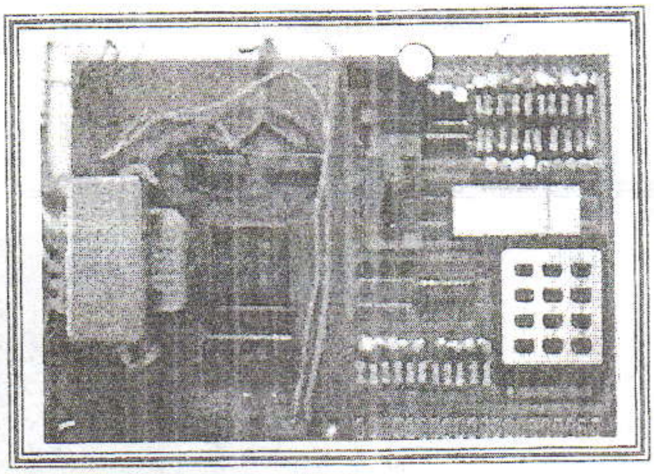


Figure (10) Photo picture of the result when the IC tested is FAIL.

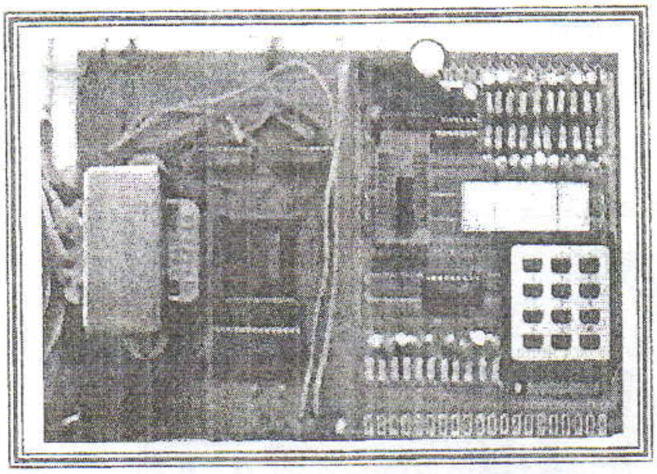


Figure (11) Photo picture of the 7432 IC.