Design and Implementation of Synthesizable VHDL Model for General PCMCIA I/O Cards Controller

Yousra Abd Mohammed

Received on :30/5/2005
Accepted on :6/3/2008

Abstract

The portable and nomadic computer market has driven the development of PCMCIA Cards to address the expansion needs for the user. These cards provide a vast variety of hardware devices which are rugged, credit-card sized, lightweight, and power efficient. These cards are easy to use, especially for the non-technical user. Since the sockets are accessible from the outside of the system, the system does not have to be powered-off, opened, and rebooted to add or remove a device. This dynamic insertion and removal feature inherently makes these devices power manageable and also allows devices to easily be shared among different computers.

This paper is concerned with type II PC cards, which mean I/O cards, therefore a design and implementation of synthesizable VHDL model for control system (Controller) of the PCMCIA I/O cards is presented.

The implementation of the control system (controller) has been done by using very high speed hardware descriptive language (VHDL) and its implementation on field programming gate array (FPGA) type Xilinx Spartan 2 (XC2S30-6 Pq208) by using synthesis and implement tools of ISE6.3 program.

The used of FPGA technology is optimal for this paper because it offers high reliability and flexibility in modifying and even developing the required design with a reduction in the required number of hardware components, also the non recurring engineering cost.

The timing behavior of the controller is be tested and verified to ensure that it meets the performance requirements by using simulation tools of Active-HDL program AND Daley report of ISE program, therefore examples of simulation results of read/write transfers for both an attribute memory and I/O devices are presented in this paper.

Key word: PCMCIA, type II PC cards, I/O Controller, VHDL, and FPGA.

الخلاصة

سوق الحاسبة النّقّالة قد ساقتْ تَطَور البطاقات PCMCIA أن يعنون حاجات التّوسع للمستعمل. هذه البطاقات تُزود تنويعةَ واسعةَ من الأجهزة المادية القوية و المتينة ، و بحجم بطاقة إئتمان ،و ذات وزن خفيف، وتُشغيلُ كفوء. هذه البطاقات سهلة الإستعمالُ، خاصةً للمستعمل غيرِ التّقنـي. لانه ال (sockets) موصولةُ من خارج النّظام،ولا حاجة لقطع القدرة ثم أعادة تشغيل و تحميل النّظام من جديد من أجل أن يضيفَ أو يزيلَ أداة. هذه خاصية الإدخال والإزالة الديناميةِ يجعلان هذه الأدوات بشكل تُشغيلُ سهلة الانقياد وتَسمح للأدوات أيضا أن تَكُون مشَاركة بشكل سهل بين الحاسبات المختلفة.

هذه المقالة تهتم بالنوع الثاني من هذه البطاقات أي بطاقات الإدخال و الإخراج ،لـ ذلك في هذه المقالةُ صمم ونُفَذَ (Synthesizable VHDL Model) لنظام سيطرة (مسـيطر) لهـ ذا النوع من البطاقات.

https://doi.org/10.30684/ etj.26.7.4
University of Technology-Iraq, Baghdad, Iraq/2412-0758

أن تصميم و تنفيذ نظام السيطرة (المسيطر) قد تـم باسـتخدام برنـامج (ISE6.3) المستند على لغة (VHDL) كبيئة برمجية و رقاقة المصفوفات البرمجية نـوع (XC2S30-6 Pq208) التي تنتجها شركة (XILINX) كأداة للكيان المادي.

أن استخدام هذه التقنية مهم جدا" في التصميم لانه يوفر موثوقية عالية و مرونة فـي تغير و تحديث مواصفات التصميم و التي تسهم في تقليل عدد المكونات المادية المستخدمة فـي التصميم و بالتالي تقليل الكلفة الكلية للمنظومة.

أن أل (timing behavior) للمسيطر قد تم تحليله للتأكد من انه يسـد حاجـات الأداء المصمم لها باستخدام أدوات المحاكاة لبرنامج أل(Active-HDL) ، لذلك تم تقديم في هذه المقالة بعض من نتائج المحاكاة لعمليات القراءة و الكتابة لأجهـزة الإدخـال و الإخـراج و الـذاكرة التعريفية.

Introduction

The PC Card Standard provides physical specifications for three types of PC Cards, with additional provisions for extended cards. All three card types measure the same length and width and use the same 68-pin connector. The only difference between the card types is thickness. The thicknesses are 3.3, 5.0, and 10.5 millimeters for Type I, Type II, and Type III cards respectively. Because they differ only in thickness, a thinner card can be used in a thicker slot, but a thicker card can not be used in a thinner slot [1].

The card types each have features that fit the needs of different applications. Type I PC Cards are typically used for memory devices such as RAM, Flash, OTP, and SRAM cards. Type II PC Cards are typically used for I/O devices such as data/fax modems, LANs, and mass storage devices. Type III PC Cards are used for devices whose components are thicker, such as rotating mass storage devices. Extended cards allow the addition of components that must remain outside the system for proper operation, such as antennas for wireless applications [2, 3].

The I/O PCMCIA Cards

I/O (Input/Output) cards are actually devices to perform some special function. I/O devices are instruments that control incoming and outgoing data flow. Traditional I/O devices for desktop PCs come in the form of interface cards that are plugged into the slots on the mainboard of the computer and include such devices as network cards, modems and fax modems.

In order to support portable computers, which do not support the installation of interface cards; these devices were designed as PCMCIA cards which could be plugged into the standard PCMCIA slot used for memory cards. I/O PCMCIA cards are usually Type II cards, i.e. cards of 5mm thickness [4].

However that I/O cards are available from a number of different manufacturers and there are different applications for this type of cards, though there is a most important common component among the basic structure of these cards which is the controller that controls I/O and Attribute memory transfers, so that

the subject of this paper is the controller.

In this paper, a general PCMCIA controller for all types of I/O cards is designed by using VHDL language as a software environment based on FPGA.

The Proposed PCMCIA I/O PC Card Controller Description

The controller is used to implement PCMCIA type II compatible I/O cards as shown in the system block diagram and in the internal chip block diagram (FPGA) in Figure (1). Both PCMCIA signals and memory devices connect directly to the controller with no additional components required.

The controller contains a complete address and data buffer, address decoder, and read and write control logic for I/O devices and an attribute memory.

The Proposed PCMCIA I/O PC Card Controller Signal Configuration

The controller signals are defined by table (1). The controller signal description table lists and describes the function of each signal used in this designed chip and also the buffer type implemented for the corresponding signal. A pound, "#", appended to a signal name indicates the signal name indicates the signal is active low.

The Proposed PCMCIA I/O PC Card Controller Operation

Since the I/O cards containing both attribute memory and I/O devices, so that these cards must be supported four types of transactions (I/O Read, I/O Write, Attribute Memory Read, and Attribute Memory Write).

The Output Enable (OE#), Write Enable (WE#), I/O Read (IORD#), I/O Write (IOWR#), and Register (REG#) signals define the transaction type. Table (2) lists each of the PC Card transaction definition signals and indicates the command signal combinations for attribute memory and I/O accesses. Note that REG# specifies access to either attribute memory address locations or I/O address locations. The memory and I/O command lines determine which address space is being accessed.

I/O registers incorporated into PC Cards can be either 8-bits or 16-bits wide. I/O cards that contain 16-bit registers assert the IOIS16# signal when 16-bit register is addressed.

The Proposed PCMCIA I/O PC Card Controller Transfer Function

I/O input/output transfer to/from I/O cards may be either 8-bit or 16-bit.

When a 16-bit transfer is attempted from a 16-bit port, the signal IOIS16# must be asserted by the card. Otherwise, the IOIS16# signal must be negated. Table (3) and Table (4) show the I/O input and output transfer function for all I/O Cards respectively.

Software Design and Synthesize of the Proposed I/O Controller

The development of the proposed I/O Controller software using VHDL language as software environment based on Xilinx Spartan 2 (XC2S30-6 Pq208) FPGA chip as hardware device, so the development becomes more reliable, modular, and adaptable to new requirements.

The FPGA program of this controller has been first written and compiled by using VHDL programming language throughout

the using of **Active-HDL** program to carry on the designed function of the controller unit. The functionality of the controller entity is defined by behavioral architecture description; figure (2) shows the general form of PCMCIA I/O PC Card controller design as programming structure and figure (3) shows the hierarchy block diagram of this controller as has been done by using schematic Editor of **OrCAD** (Organization Computer Aided Design) program.

After the design entry has been entered, the first Simulation phase (Functional) is used to debug VHDL design resources, in this paper functional simulation of **Active-HDL** program is used to produce the timing diagram results of the controller, Figures from (4) to (7) show examples of these timing diagrams.

Once the high-level design is validated, the process of physically implementing the design begins [5, 6]. First the VHDL source code of the controller design is synthesized using **ISE** (Integrated Software Environment) FPGA Express software. Synthesis takes the behavioral description of the design and formulates the physical layout of the circuit and the result is optimized for the targeted device (XC2S30-6 Pq208) as shown in figure (8), which is summarized an output of synthesize report of **ISE** Package.

Implementation of the Design for Proposed I/O Controller and Report Summary

The design implementation is concerned with the exact selection of the circuit primitives and their placement and routing within some given constraints. This process has several steps: the first step is to translate the design, which includes a produce to a complete hardware design. Then the map step that translate the gates level design into hardware primitives available in (XC2S30-6 Pq208) FPGA. Once the design has been applied to these primitives they must be assigned to physical locations on the chip and route the connections between them, this task is the function of the place and route [7, 8]. Once the design is fully placed and routed accurate timing information about the design can be generated and the design can be downloaded to the FPGA and tested.

In this paper the design is implemented by the implementation tools of **ISE** package, after the completion of the implementation steps an implementation reports file for each step will be generated as shown in figures (9), (10), (11), (12), and (13) consecutively.

When the process of generating a device-level implementation of the design is complete, the results can be examined using floor planner as shown in figure (14).

Conclusion

In this paper, a complete design for general PCMCIA I/O Cards Controller has been made based on VHDL and FPGA as software and hardware respectively.

Using FPGA in the development of the proposed controller provides a cost-effective method with high flexibility for tolerating permanent faults in the system through its configuration. Using FPGA technology in the design of this controller has also enabled to replace

a number of chips in the exiting system results in a reduction in the number of hardware components and substantial area reduction. Moreover, the low cost of implementation and short time needed to physically realize the design using this approach, is provided enormous advantage over traditional approaches for building prototype hardware which is in turn leads to improve the performance of the system with higher flexibility in achieving such requirements.

From the first look to the figure of the Delay report, it can be noted from the netlist delays that the delay of the net "REG_IBUF" is the limiting factor that limits the whole design's speed. However, this delay is very little as compared with bus speed (10 MHz) so that there is no problem from this delay in the proposed design.

Future work

Based on this design and the results of the proposed system, the following points are suggested for future work to improve the system performance.

1. Using improved versions of synthesis tools when they are available, this is because the limiting factor in developing more complex designs on hardware systems are synthesis tools.
2. If The I/O controller is used for special type of I/O cards then the controller can be developed to include more facilities and capabilities by building the attribute memory device inside the FPGA because on chip RAM is faster than off chip RAM implementation; also this is very useful to reduce the size of the memory card.

References

[1] Don Anderson; "PCMCIA System Architecture/16-BIT PC Cards"; MindShare, Inc.; Second Edition; 1995.

[2] Don Anderson and Tom Shanley; "CARDBUS System Architecture"; MindShare, Inc.; 1996.

[3]Stephen Bigelow; "Troubleshooting, Maintaining and Repairing PCs"; McGrow-Hill Companies, Inc.; August; 2001.

[4] ftp://ftp.emacinc.com/m-3112.pdf ; "PC/104 PCMCIA Module"; Part No. 2007311000 User's Manual; First Edition; Printed in Taiwan; July; 1997.

[5] Sudhakar Yalamanchill; "Introductory VHDL: From Simulation To Synthesis"; Prentice-Hall, Inc.; 2001.

[6] XILINX Corporation; "Synthesis and Simulation Design Guide"; Xilinx Development System; 2003; at http://www.xilinx.com.

[7] XILINX Corporation; "Spartan and Spartan-XL Families Field Programmable Gate Arrays"; Xilinx Product Specification; DS060 (V1.7) June 27, 2002. at http://lhcb-online.web.cern.ch/lhcb-online/ecs/ccpc/docs/XCS05xl%20datasheet.pdf.

[8] Chan P., Mourad,S. "Digital Design Using Field Programmable Gate Arrays", Prentice-Hall, Inc.; 1994.

Table (1) Controller Signals Description

Name	Type	Description
Datain [15:0]	Bidir	PCMCIA Data Bus (lower & upper data path).
Dataout	Bidir	I/O Controller Data Bus (lower & upper data path).
Address[25:0	Input	PCMCIA Address Bus.
ADD [25:0]	Output	I/O Controller Address Bus.
CE# [1:0]	Input	Active low, PCMCIA byte enable for odd and even bytes.
REG#	Input	Active low, PCMCIA signal low for attribute memory and I/O. devices.
OE#	Input	Active low, PCMCIA output enable signal.
WE#	Input	Active low, PCMCIA write enable signal.
AOE#	Output	Active low, Attribute memory output enable.
AWE#	Output	Active low, Attribute memory write enable.
CS#	Output	Active low, Chip enable for Attribute memory.
IORD#	Input	Active low, this signal is asserted during I/O read transfer from PCMCIA cards.
IOWR#	Input	Active low, this signal is asserted during I/O write transfer from PCMCIA cards.
IOIS16#	Output	Active low, this signal is asserted during I/O transfer from/to PCMCIA cards, if the device size is 16-bits.
INPACK#	Output	Active low, this signal is asserted during I/O read transfer from a PCMCIA card when it recognizes the address. The signal enables the socket's data path transceiver so that the addressed PCMCIA Card can deliver valid data to the system.

Table (2) I/O PCMCIA I/O Transaction Definition

Transaction Type	REG#	OE#	WE#	IORD#	IOWR#
I/O Read	0	1	1	0	1
I/O Write	0	1	1	1	0
Attribute Memory Read	0	0	1	1	1
Attribute Memory Write	0	1	0	1	1

Table (3) I/O Input Function for All I/O Cards

Function Mode	REG#	CE2#	CE1#	A0	IORD#	IOWR#	Data 15:8)	Data (7:0)
Standby mode	X	H	H	X	X	X	High-Z	High-Z
Byte Input (8-bit)	L	H	L	L	L	H	High-Z	Even-byte
	L	H	L	H	L	H	High-Z	Odd-byte
Word Access (16-bit)	L	L	L	L	L	H	Odd-byte	Even byte
I/O inhibit	H	X	X	X	L	H	High-Z	High-Z
High Byte Only	L	L	H	X	L	H	Odd-byte	High-Z

Table (4) I/O Output Function for All Cards

Function Mode	REG#	CE2#	CE1#	A0	IORD#	IOWR#	Data (15:8)	Data(7:0)
Standby mode	X	H	H	X	X	X	X	High-Z
Byte Input (8-bit)	L	H	L	L	H	L	X	Even-byte
	L	H	L	H	H	L	X	Odd-byte
Word Access (16-bit)	L	L	L	L	H	L	Odd-byte	Even byte
I/O inhibit	H	X	X	X	H	L	X	X
High Byte Only	L	L	H	X	H	L	Odd-byte	X

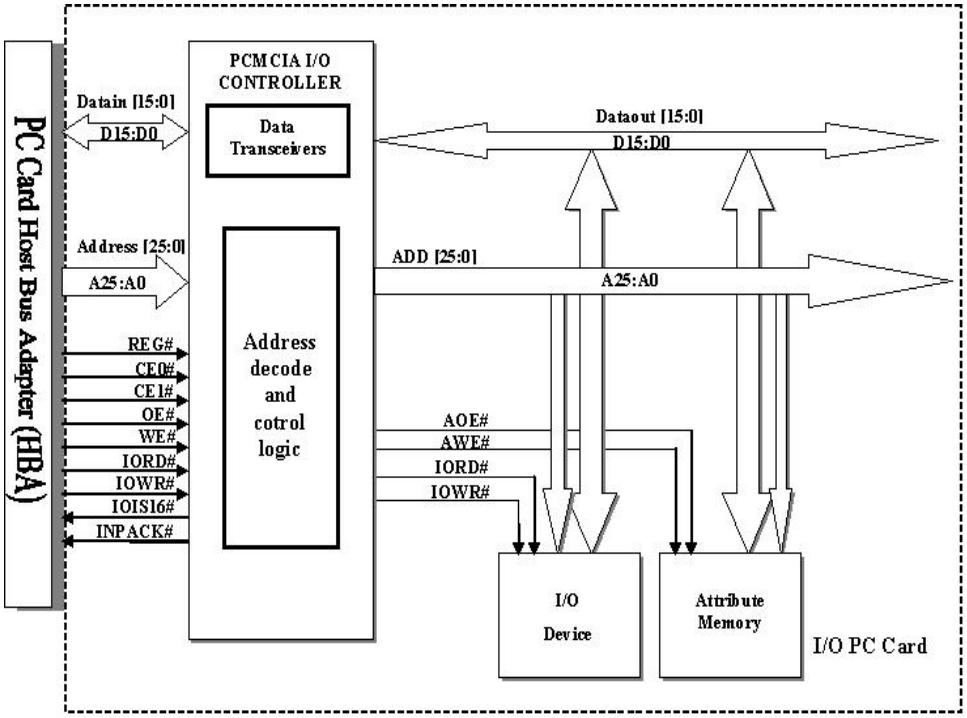

Fig. (1)Functional Block Diagram of I/O PCMCIA Cards

```
PCMCIA I/O Card Controller program
Entity Declaration (define the type and mode for all signals)
Signals Initialization
main ()
{
if (REG# = 0) then
        if (IORD# & IOWR# = 1) then
  {Attribute memory is selected for reading or writing
configuration information from/to card information structure (CIS)
according to the states of card enable CE# [1:0], output enable (OE#), and
write enable (WE#) signals}

      else if (IORD# or IOWR# = 0) then
        {I/O device is selected for reading or writing working data
according to the states of CE# [1:0], IORD#, IOWR#, IOIS16, and
INPACK# signals}
end if
end if
}       /* end of PCMCIA I/O Card Controller program */
```

Fig. (2) General Form of PCMCIA I/O Card Controller Design

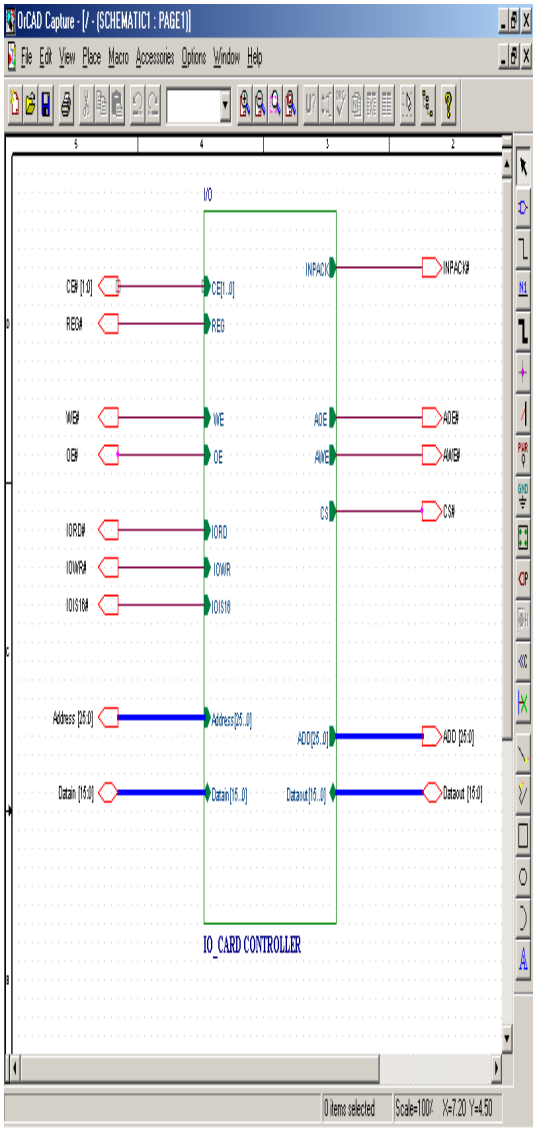

Fig. (3) Hierarchy Block of I/O PC Card Controller

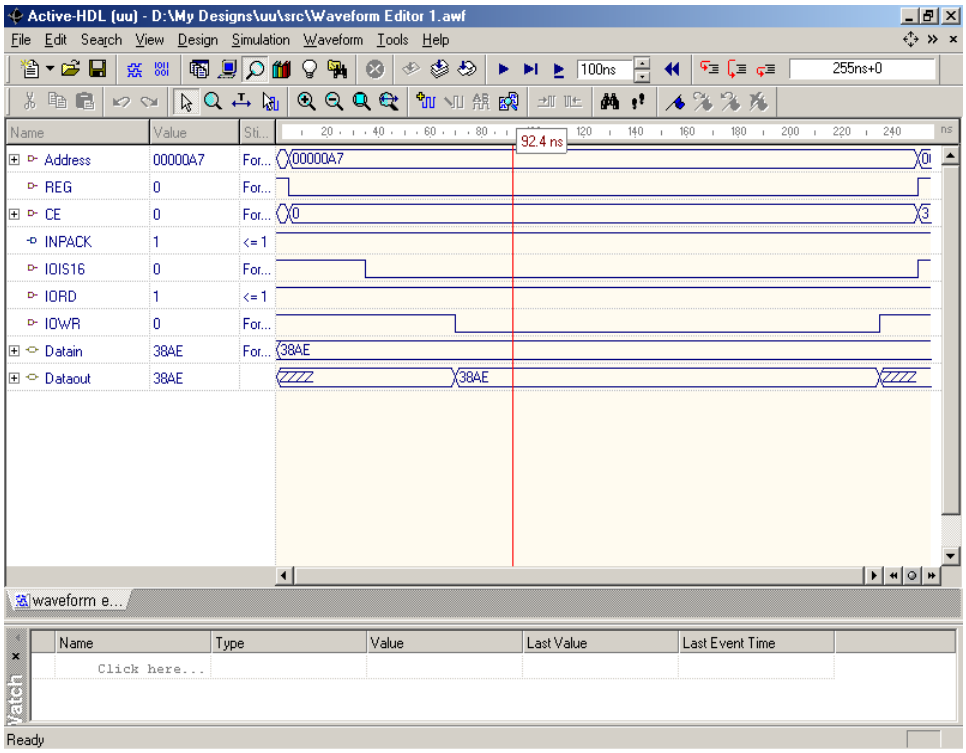

Fig.(4) I/O Write WHEN CE=00 for 16-BIT Device

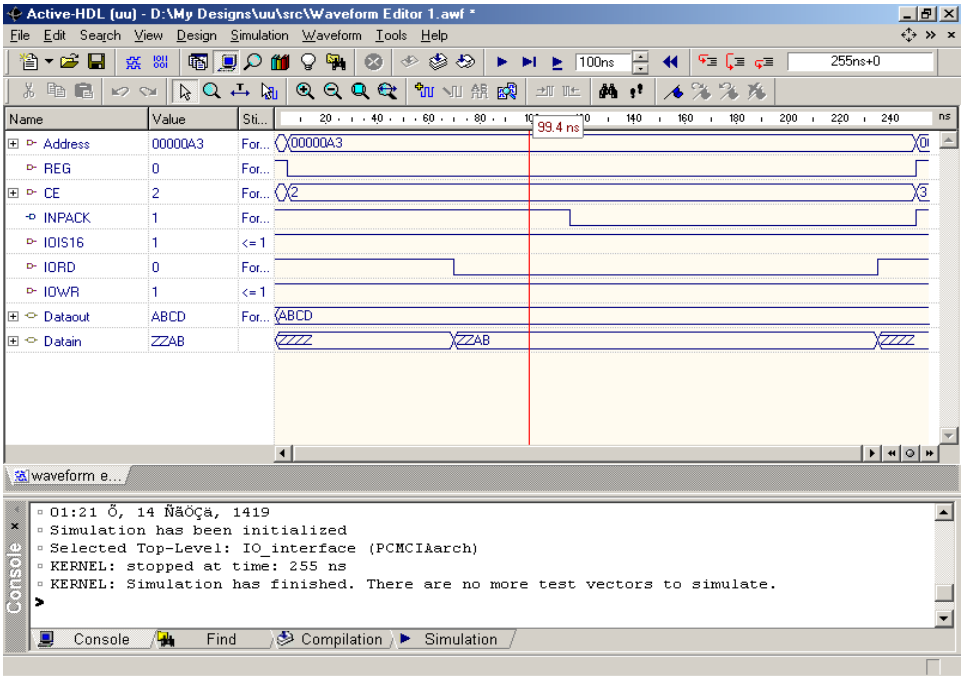

Fig. (5) I/O Read when CE=10 and A0=0 for 8-Bit Device

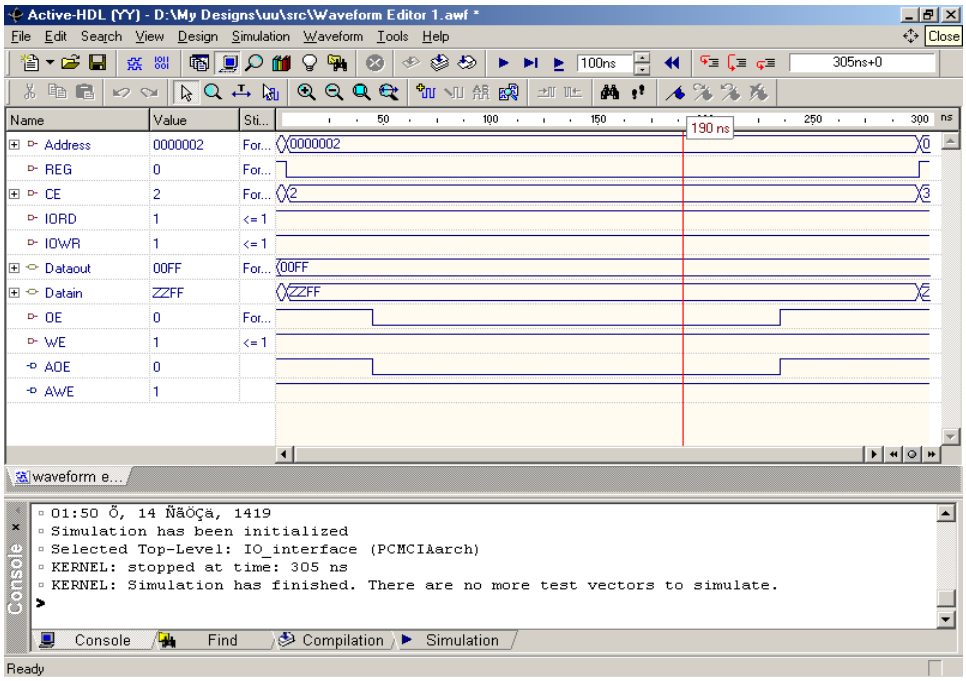

Fig (6) Attribute Memory Read

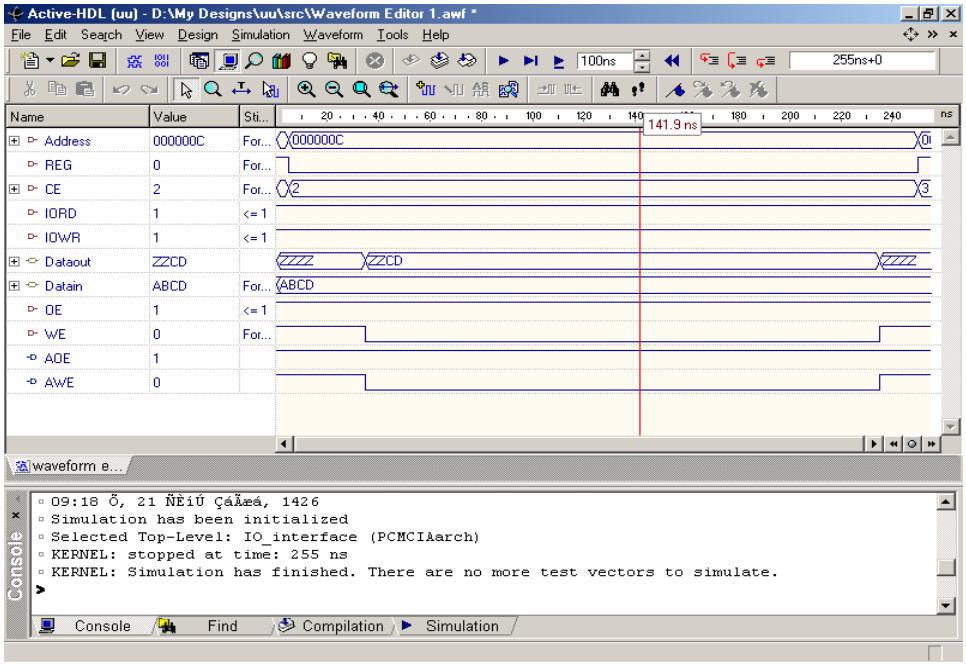

Fig (7) Attribute Memory Writ

```
Release 6.3i - xst G.35
Copyright (c) 1995-2004 Xilinx, Inc.  All rights reserved.
--> Reading design: io_interface.prj
---- Source Parameters
Input File Name                    : io_interface.prj
Input Format                       : mixed
--- Target Parameters
Output File Name                   : io_interface
Output Format                      : NGC
Target Device                      : xc2s30-6-pq208
Final Results
RTL Top Level Output File Name     : io_interface.ngr
Top Level Output File Name         : io_interface
Output Format                      : NGC
Optimization Goal                  : Speed
Keep Hierarchy                     : NO

Design Statistics
# IOs                              : 96
Macro Statistics :
# Multiplexers                     : 16
# 2-to-1 multiplexer               : 16
# Tristates                        : 32
# 1-bit tristate buffer            : 32

Cell Usage :
# BELS                             : 65
#      LUT2                        : 13
#      LUT3                        : 1
#      LUT4                        : 51
# IO Buffers                       : 96
#      IBUF                        : 34
#      IOBUF                       : 32
#      OBUF                        : 30
=========================================================================
Device utilization summary:
---------------------------

Selected Device : 2s30pq208-6

 Number of Slices:                        37  out of   432     8%
 Number of 4 input LUTs:                  65  out of   864     7%
 Number of bonded IOBs:                   96  out of   136    70%
```

Fig (8) The Output of Synthesize Report File

```
Copyright (c) 1995-2004 Xilinx, Inc.  All rights reserved.
Command Line: ngdbuild -intstyle ise -dd c:\hhh/_ngo -i -p xc2s30-pq208-
6io_interface.ngc io_interface.ngd

        Reading NGO file "c:/hhh/io_interface.ngc" ...
        Reading component libraries for design expansion...

        Checking timing specifications ...
        Checking expanded design ...

        NGDBUILD Design Results Summary:
          Number of errors:   0
          Number of warnings:   0

        Total memory usage is 37120 kilobytes

        Writing NGD file "io_interface.ngd" ...

Writing NGDBUILD log file "io_interface
# Multiplexers                     : 16
# 2-to-1 multiplexer               : 16
# Tristates                        : 32
# 1-bit tristate buffer            : 32

Cell Usage :
# BELS                             : 65
#      LUT2                        : 13
#      LUT3                        : 1
#      LUT4                        : 51
# IO Buffers                       : 96
#      IBUF                        : 34
#      IOBUF                       : 32
#      OBUF                        : 30
=========================================================================
Device utilization summary:
---------------------------

Selected Device : 2s30pq208-6

 Number of Slices:                        37  out of   432     8%
 Number of 4 input LUTs:                  65  out of   864     7%
 Number of bonded IOBs:                   96  out of   136    70%
```

Fig (9) The Output of Translate Report File

```
Release 6.3i Map G.35
Xilinx Mapping Report File for Design 'io_interface'

Design Information
------------------
Command Line   : C:/Xilinx/bin/nt/map.exe -intstyle ise -p xc2s30-pq208-6 -cm
area -pr b -k 4 -c 100 -tx off -o io_interface_map.ncd io_interface.ngd
io_interface.pcf
Target Device  : x2s30
Target Package : pq208
Target Speed   : -6
Mapper Version : spartan2 -- $Revision: 1.16.8.2 $
Mapped Date    : Tue June 23 5:44:29 2000

Design Summary
--------------
Number of errors:     0
Number of warnings:   0
Logic Utilization:
  Number of 4 input LUTs:          65 out of   864   7%
Logic Distribution:
    Number of occupied Slices:                    34 out of   432   7%
    Number of Slices containing only related logic:    34 out of    34  100%
    Number of Slices containing unrelated logic:       0 out of    34   0%
Total Number of 4 input LUTs:       65 out of   864   7%
  Number of bonded IOBs:           96 out of   132  72%

Total equivalent gate count for design:  486
Additional JTAG gate count for IOBs:  4,608
Peak Memory Usage:  57 MB
# IO Buffers                : 96
#     IBUF                  : 34
#     IOBUF                 : 32
#     OBUF                  : 30
-----------------------------------------------------------------------
Device utilization summary:
---------------------------

Selected Device : 2s30pq208-6

 Number of Slices:                  37  out of   432    8%
 Number of 4 input LUTs:            65  out of   864    7%
 Number of bonded IOBs:             96  out of   136   70%
```

Fig (10) The Output of Map Report File

```
Release 6.3i Par G.35
Copyright (c) 1995-2004 Xilinx, Inc.  All rights reserved.
MYCOMPUTER: Tue June 23 5:44:33 2005
C:/Xilinx/bin/nt/par.exe -w -intstyle ise -ol std -t 1 io_interface_map.ncd
io_interface.ncd io_interface.pcf
Constraints file: io_interface.pcf
Loading device database for application Par from file "io_interface_map.ncd".
"io_interface" is an NCD, version 2.38, device xc2s30, package pq208, speed
-6
Loading device for application Par from file '2s30.nph' in environment
C:/Xilinx.
Device speed data version:  PRODUCTION 1.27 2004-06-25.

Device utilization summary:
   Number of External IOBs          96 out of 132    72%
   Number of LOCed External IOBs    0 out of 96      0%
   Number of SLICEs                 34 out of 432    7%
```

Fig (11) The Output of Place and Route Report File

Delay Report

Release 6.3i - reportgen G.35
Copyright (c) 1995-2004 Xilinx, Inc. All rights reserved.
Tue June 23 5:02:36 2005
File: io_interface.dly
The 20 worst nets by delay are:

Max Delay	Netname
4.964	REG_IBUF
4.679	IOWR_IBUF
3.951	IORD_IBUF
2.632	Address_0_IBUF
2.464	ADD_10_OBUF
2.357	ADD_23_OBUF
2.333	INPACK_OBUF
2.309	AWE_OBUF
2.189	ADD_9_OBUF
2.165	CS_OBUF
2.119	Address_5_IBUF
2.118	Address_22_IBUF
2.112	_n0135
1.979	_n0140
1.950	_n0133
1.944	Address_16_IBUF
1.779	ADD_25_OBUF
1.763	CE_1_IBUF
1.756	_n0125
1.697	ADD_7_OBUF

Fig (12) The Output of Delay Report File

Release 6.3i - Bitgen G.35
Copyright (c) 1995-2004 Xilinx, Inc. All rights reserved.

Loading device database for application Bitgen from file "io_interface.ncd".
"io_interface" is an NCD, version 2.38, device xc2s30, package pq208, speed -6
Loading device for application Bitgen from file '2s30.nph' in environment
C:/Xilinx.
Opened constraints file io_interface.pcf.

Tue June 23 5:19:17 2005

C:/Xilinx/bin/nt/bitgen.exe -intstyle ise -w -g DebugBitstream:No -g Binary:no -g Gclkdel0:11111 -g Gclkdel1:11111 -g Gclkdel2:11111 -g Gclkdel3:11111 -g ConfigRate:4 -g CclkPin:PullUp -g M0Pin:PullUp -g M1Pin:PullUp -g M2Pin:PullUp -g ProgPin:PullUp -g DonePin:PullUp -g TckPin:PullUp -g TdiPin:PullUp -g TdoPin:PullUp -g TmsPin:PullUp -g UnusedPin:PullDown -g UserID:0xFFFFFFFF -g StartUpClk:CClk -g DONE_cycle:4 -g GTS_cycle:5 -g GSR_cycle:6 -g GWE_cycle:6 -g LCK_cycle:NoWait -g Security:None -g DonePipe:No -g DriveDone:No io_interface.ncd

Summary of Bitgen Options:

Option Name	Current Setting
Compress	(Not Specified)*
Readback	(Not Specified)*
DebugBitstream	No**
ConfigRate	4**
StartupClk	Cclk**
CclkPin	Pullup**
DonePin	Pullup**
M0Pin	Pullup**
M1Pin	Pullup**
M2Pin	Pullup**
ProgPin	Pullup**
TckPin	Pullup**
TdiPin	Pullup**

Fig (13) The Output of Generation Format Report File

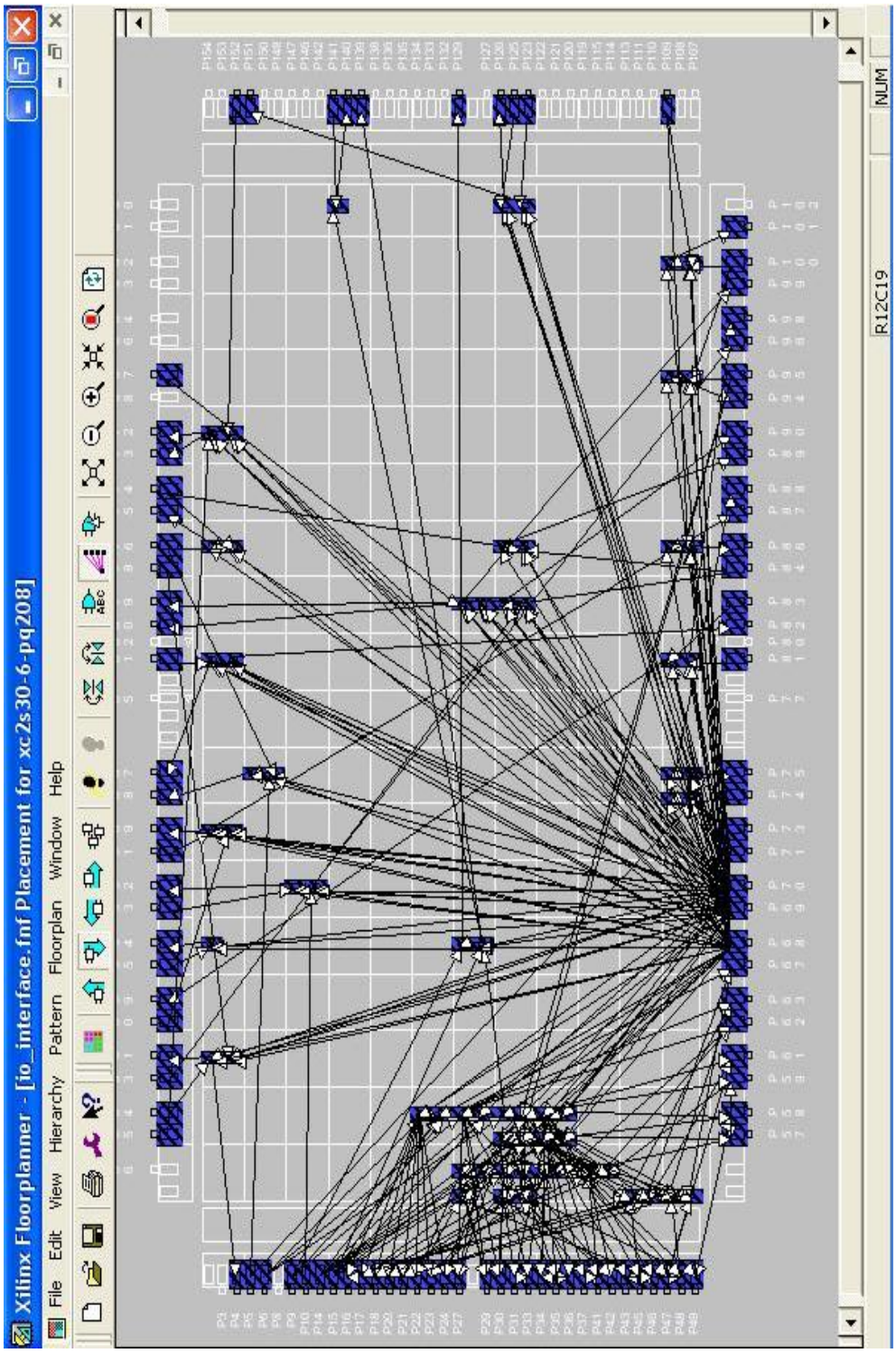

Fig (14) The Floor Planner of the I/O Controller Design