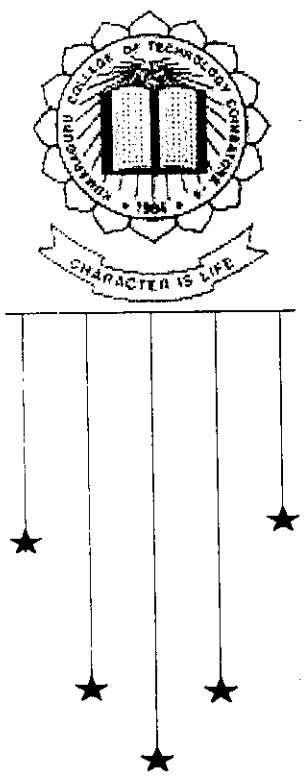
DESIGN AND SIMULATION OF CONTROL OF AUTOMATION USING VERILOG HDL



PROJECT REPORT
2000 - 2001

Submitted By

LAKSHMIKANTH ARUN .A
PRADEEP .P
RUCKMANI .A
SHARMILA .D

Under the Guidance of

Mr.S. KARTHIK, M.E.,
Lecturer,
Electrical and Electronics Department.

In partial fulfillment of the requirements for the award of the degree of

BACHELOR OF ENGINEERING IN

ELECTRICAL AND ELECTRONICS ENGINEERING

of the Bharathiar University, Coimbatore.

Department of Electrical and Electronics Engineering

Kumaraguru College Of Technology

Coimbatore- 641 006.

KUMARAGURU COLLEGE OF TECHNOLOGY
COIMBATORE – 641006.

DEPARTMENT OF ELECTRICAL AND ELECTRONICS ENGINEERING

Certificate

This is to certify that the project entitled

DESIGN AND SIMULATION OF CONTROL OF AUTOMATION USING VERILOG HDL

has been submitted by

Mr/Ms. LAKSHMIKANTH ARUN, PRADEEP. P, RUKKMAN. A, SHARMILA. S

in partial fulfillment of the requirements for the award of Degree Of

BACHELOR OF ENGINEERING in

ELECTRICAL AND ELECTRONICS ENGINEERING

branch of the Bharathiar University, Coimbatore,
during the academic year 2000-2001.

(Guide) 19/3/2001

Dr. K. A. PALANISWAMY, B.E,M.Sc.(Engg),Ph.D
MISTE,C.Engg(I),FIE
Professor and Head
Department of Electrical and Electronics Engineering
Kumaraguru College of Technology
Coimbatore 641 006

(Head Of Department)

Certified that the candidate with register number _____ was examined in the project work and viva voce Examination held on _____

(Internal Examiner)

(External Examiner)

Dedicated to

Our Beloved Parents

And Friends, who made us

What We are!

Acknowledgement

ACKNOWLEDGEMENT

We are greatly indebted to our beloved guide **Mr. S. Karthik M.E., MISTE., MIEEE.,** Lecturer in Electrical and Electronics Engineering for his invaluable guidance and timely support. As a token of our gratitude, we extend our sincere thanks and we honour him for his assistance towards this cause.

We extend our special acknowledgement to our beloved Professor and Head of the Department **Dr.K.A.Palanisamy M.sc.(Engg, Ph.D., FIE, C.Eng.(I), MISTE** whose consistent support and enthusiastic involvement helped us a great deal in successful completion of this project.

We thank our Principal **Mr. K.K. Padmanaban B.Sc. (Engg), M.Tech, Ph.D.,** for his kind patronage. We owe our sincere gratitude to the management for the interest and encouragement in making the project a success.

In this regard, it is our pleasure to thank our class advisor **Mr.T.Kannan, M.E., MISTE., MSSI.,** Lecturer, Department Of EEE, for his special interest shown in progress of the project.

Last but not the least we take the privilege to extend our thanks to the non-teaching staff and our dear friends for their support and assistance.

SYNOPSIS

The design of control of automation control is a VLSI (Very Large Scale Integration) approach for controlling the physical parameters such pressure, temperature and flow rate. All the logic circuits involved are represented in Verilog Hardware Descriptive Language, which is more advantageous and efficient. The main purpose of using Verilog HDL is to bring the compactness and modifiability in the design for future expansion.

The main idea of our project deals with designing a control system to control the physical parameters such as water feed rate, water pressure, steam temperature, steam pressure and its flow rate and to maintain the parameters at the specified level for effective functioning of the system. The entire control logic is made out of Verilog HDL .

Once the logic is designed, it can be implemented, synthesized, optimized and fabricated on a chip. The parameters to be controlled are sensed by means of the primary sensing element, converted into clock pulses and are given as input to the chip. The chip is designed in such a way that it co-ordinates

with the input signal. The parameter values are corrected and adjusted according the design value with the help of Verilog HDL software. For our verification, the values obtained from the output of the chip can be displayed. This design opens new vistas of application for boiler control, ocombustion and IC engine control in an efficient manner which would be tedious with the available trends.

CONTENTS

CHAPTER – I

INTRODUCTION

Steam is mainly required for power generation, process heating and space heating and also in various industrial applications. Due to the requirement of high efficiency, the steam is produced at high pressure and the temperature of steam is also very high in the rate of 1800 degree centigrade. To ensure safe and efficient operation of the system as a whole, the parameters such as pressure, temperature and flow rate of steam should be maintained at a tolerable level without fluctuations. So, now there is a call for more and more better control system, which are simpler in design and easy to implement.

1.1 OBJECTIVE OF THE PROJECT

This is precisely, the main theme behind our control logic, which senses the values of the parameters mentioned above and manipulates it according to the requirements of the system.

When steam is used as the working medium in turbines, power plants and in various industries, the physical parameter should be maintained at a constant level as required. When the value of these parameters change or goes below or above the tolerable limit, then

- Desired efficiency will not be achieved.
- Performance of the system gets reduced.
- Overheating of the system may take place ultimately leading to bursting.
- Safety for the working personnel involved is not ensured.
- The system will not be able to meet the requirements if the parameters are fluctuating.
- Life of the system gets reduced

So a control system is necessary to maintain all the parameters at the specified value for the efficient functioning of the system. Designing a better control system using digital techniques is the main objective of our project.

1.2 CONVENTIONAL METHOD

In the past years, the control of these parameters was implemented by means of pneumatic control system. Recorders were placed at each and every stage to record the values and the values were adjusted manually. But this system was not efficient as controlling all the parameters manually and simultaneously proved to be tedious.

1.3 PRESENT TREND

After the explosive growth of the computer industry in recent years, the control switched over from pneumatic methods to DDCS (Distributed Digital Control System). In this system, computer is used. WINDOWS was used as the operating system.

1.3.1 DISTRIBUTED DIGITAL CONTROL SYSTEM(DDCS)

Fig.1.1 shows the general block diagram of DDCS. The parameter is first sensed by means of the primary sensing element. Then intermediate processor, which consists of an Analog to Digital converter processes it and the digital signal, is given as input to the computer through data acquisition cards. The software is written with Windows as the operating system. Then the corresponding outputs are given to the control circuit.

Though the method is highly advantageous when compared to the conventional method, this trend has a series of drawbacks.

1.3.2 DISADVANTAGES

The disadvantages of DDCS is as follows:

- When a computer is used, many interfacing circuitries are necessary. Designing and interfacing these circuits is a complicated process.
- Interfacing circuits are costly
- Due to the interfacing devices, time delay is increased and the control is not effective.
- Power consumption is more.
- The lines should be free from interference and other disturbances.
- The system must be protected from voltage spikes.
- Isolators must be provided both on the input and output side.

In order to simplify the above process and to have effective control of all the parameters, we have proposed a control system where in the control is implemented in Verilog HDL software.

1.4 CHAPTERS COVERED

The various chapters covered are as follows:

The digital design is done in CMOS VLSI technology, which has now created a revolution in microelectronics field. So the various layers of abstractions and representations in VLSI are briefly explained in the **Chapter–II**. A very brief and broad overview of the trends in VLSI technology is also presented in this section.

Verilog HDL is used as the basic tool to build the control logic. The **Chapter-III** deals with the basics of the hardware language, importance of HDLs, design flow, various design methodologies, the various features of Verilog HDL ,tool flow diagram, design synthesis, advantages of verilog HDL and the program structure of Verilog HDL, which is the master to the entire project.

The **Chapter-IV** is concerned with the basic block diagram required for the design, control block diagram that is to be implemented in Verilog HDL. A broad overview of the design blocks and the top down design methodology, which are of primary concern, is also presented here in a vivid manner.

Details regarding the software i.e., the input signals, the counter, register and the comparator blocks, the enable signals used to enable the various blocks at appropriate time intervals, the process of controlling, the manipulation of errors, interconnection of individual blocks to represent an integrated block have been covered in the **Chapter-V.**

To put the above implementation into effective use, a test bench is to be created to simulate the above design and to verify the functionality of these circuits. Thus a stimulus block is used to test the design block and this is usually the top-level block. The **Chapter-VI** is concerned with test benches.

Due to the extensive growth of VLSI technology and hardware languages, the design styles are undergoing rapid revolution. This chapter also covers how the newly available trends in HDLs can expand the design presently developed by us.

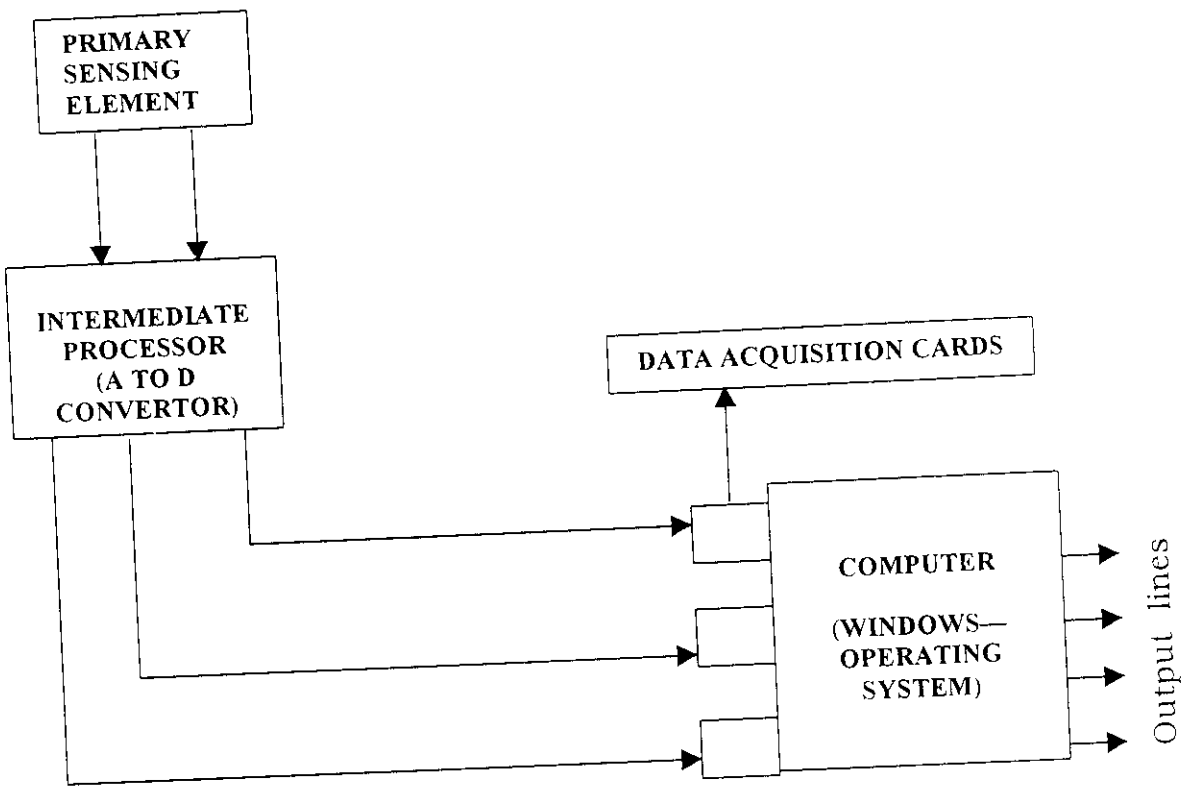


Fig.1.1 Distributed Digital Control System

CHAPTER - II

VLSI

2.1 INTRODUCTION

Digital circuit design has evolved rapidly starting over the past two decades starting from vacuum tubes and transistors. The growth of semiconductor (SC) technology led to the invention of integrated circuits where logic gates were placed on a chip. The first Integrated Chip (IC) chips were SSI(Small Scale Integration) where the number of components were limited. As technologies became sophisticated, the number of gates on a chip increases and they were called MSI (Medium Scale Integration). With the advent of LSI (Large Scale Integration) designers were able to put thousands of components on a single chip. At this point, design processes started getting very complicated and designers felt the need to increase the number and to automate the processes.

The term Very Large Scale Integration (VLSI) reflects the capabilities of the semiconductor industry to fabricate a complex electronic circuit consisting of more than 1,00,000 components on a single silicon substrate.

2.2 OVERVIEW OF DIGITAL DESIGN

In the digital design, Silicon a semiconductor forms the basic starting material for a large class of integrated circuits. Because of the circuits, verification of these circuit on a breadboard is impossible. Automated design tools became critical for verification and design of VLSI digital circuits. By using these tools, designers can easily build small building blocks and then derive higher level blocks from them. Logic simulators are also available to test the functionality of VLSI digital circuits before fabrication.

2.3 FEATURES OF VLSI

Several factors attributed to this tremendous growth such as,

- Reduction of line width of basic device.
- Development of high-resolution techniques.
- Improved processing capabilities
- Improves reliability of processing.
- Simplicity in layout design.
- Better understanding of system level design.
- Availability of automated tools for circuit layout, simulation, verification and testing.

2.4 LAYERS OF ABSTRACTION

The VLSI design process spans a diversified spectrum of disciplines in various fields. Because of the diversity of tasks and design issues, it is essential to break the process into a number of design layers.

- Physical Layer
- Layout Layer
- Circuit and Logic Layer
- Function Layer
- System Layer

2.4.1 PHYSICAL LAYER

Design issues that are of concern at this stage are the speed of the circuit as determined by,

- Circuit parameters
- Signals in connecting wires
- Total power consumption of the circuit
- Heat buildup
- Precise sequencing of voltage and current that controls the flow of information within the circuit.

2.4.2 LAYOUT LAYER

Layout layers is nothing but it is an

- Interface between physical world and electrical world
- Serves as a link between circuit and fabrication process that builds the circuit.

Design issues that is of primary concern are

- To optimize the area of the layout
- Context surrounding the layout
- Whether the lines in the circuit are global, carrying power, ground or a control line.

2.4.3 CIRCUIT AND LOGIC LAYER:

Design issues of Circuit and Logic Layer are

- It hides the details of delay and timing considerations by making the circuit work synchronously.
- This level uses an abstraction of the underlying electrical circuit in which currents and voltages are limited to discrete levels

This Stage of abstraction is,

- Correctness of the overall logic operations of the circuit in terms of the elementary operations performed by the logical elements in the specified timing sequence.
- The simplicity and elegance of the design so that it blends well with the higher-level algorithmic process at the functional level and with the flow of data.

2.4.4 FUNCTION LAYER:

Function layer of design is a

- Conceptual solution to the problem in the form of an algorithmic process.
- Specification for major function blocks.
- Their interconnection to implement the heart of the algorithm.

Design issues that must be taken care of Function layer are

- Evolution of regularity of structure.
- Data flow organization.
- Communication needs between the blocks.

2.4.5 SYSTEM LAYER:

System Layer is the highest level of design. This is concerned with

- Connecting the major subsystems
- Communication interfaces with external world.
- Selecting layers for carrying global control, data power
- Placement of major subsystems
- Routing strategies.

2.5 TECHNOLOGY TRENDS IN VLSI:

The technology of semiconductor devices is advancing at a rapid rate and is highly preferred for the most high speed and low noise applications. In this regard, the Metal Oxide Semiconductor (MOS) technology is growing in leaps and bounds because of high volume commercial applications. Primarily there are two MOS technologies available.

- n-Channel Metal Oxide Semiconductor (nMOS)
- p- Channel Metal Oxide Semiconductor (pMOS)

These technologies have some advantages in terms of size or area of silicon need to produce functionality. A combination of nMOS and pMOS led to the development of CMOS with added advantages. CMOS is considered to be a technological revolution in microelectronics field and is the leading VLSI system technology.

2.5.1 pMOS TECHNOLOGY

pMOS technology passes a weak low signal and a good high signal. When a low '0' signal is passed as input, the output is degraded. When a high '1' signal is passed as input, the output is well. The symbol of 'p' channel inverter is shown in Fig 2.1.

2.5.2 nMOS TECHNOLOGY

nMOS technology passes a good low signal and a weak high signal (degraded signal). When a low '0' signal is passed as input, the output is well. When a high '1' signal is passed as input, the output is degraded. The symbol of 'n' channel inverter is shown in Fig. 2.2.

As nMOS technology passes a good low signal and a weak high signal and pMOS technology passes a weak low signal and a good high signal. It is not possible to use the technologies. So to obtain both the signals properly, the CMOS technology was developed by combining both nMOS and pMOS with many advantages.

2.6 CMOS LOGIC:

CMOS - Complementary Metal Oxide Semiconductor

CMOS combines the features of nMOS and pMOS in which both the signals high and low can be passed undegraded. It provides an inherently low power static circuit technology that has the capability of providing a lower power-delay product when compared to nMOS and pMOS.

2.6.1 REPRESENTATION OF CMOS INVERTER:

- P and N channel are used
- Silicon substrate is n type in which p-well is created by diffusion.

When the input voltage Vin = 0, the gate of the p-channel transistor is at V_{dd} below the source potential, that is Vgs = V_{dd}, which will turn on this transistor, offering a low resistance path to load capacitance C, which will be charged up to V_{dd}. No current flows through the n-channel transistor, which is turned off since Vgs = 0 for this transistor. If the input voltage is now increased to its threshold voltage and then to V_{dd}, the n-channel transistor will conduct while the p-channel transistor is turned off, discharging the load capacitance C to ground potential. The current flows until the output node reaches V_{dd}(when charging) or ground (when discharging), the transistors to provide either the charge or the discharge current.

2.6.2 FEATURES OF CMOS:

- Size is minimum. It stands at 1.5microns(um).
- Maximum number of gates per chip is 1000,000.
- Gate delay is less than 1ns.
- 120 to 125 pin packages are available.
- 1000 pin packages are under development.
- Marking technology is by using electron beam and X-rays.

2.7 COMPARISION BETWEEN nMOS, pMOS AND CMOS

❖ **Power Consumption**

nMOS draws current when input is at logic 1. pMOS draws current when input is at logic 0. CMOS draws no current at any stable state hence less power consumption when compared to other two technologies

❖ **Dynamic Power Consumption**

nCMOS current flows only during transaction. So it has dynamic power consumption.

❖ **Ratioless:**

Output does not depend on the dimension of the p and n channel transistors, so CMOS can be made of minimum size.

❖ **Noise Immunity:**

Here both the signal are transmitted undegraded. Hence there is larger output voltage and higher noise immunity when compared to nMOS and pMOS.

2.8 CIRCUIT REPRESENTATION:

Any complex digital design is expressed in terms of these design domains.

- Behavioral Representation
- Structural Representation
- Physical Representation

2.8.1 BEHAVIORAL REPRESENTATION:

This domain specifies what a system does and how a particular design should respond to a series of inputs. The algorithms of the design are written in Hardware Descriptive Language (HDLs). HDLs support specific hardware concepts such as concurrency, time and word size in a convenient manner.

2.8.2 STRUCTURAL REPRESENTATION:

This specifies how entities are connected together to effect the prescribed behavior. This description indicates a list of modules and their interconnection. It also indicates how components are interconnected to form a function.

2.8.3 PHYSICAL REPRESENTATION:

This indicates how to build a structure to implement the prescribed behavior. This provides a list of leaf cells and their interconnection.

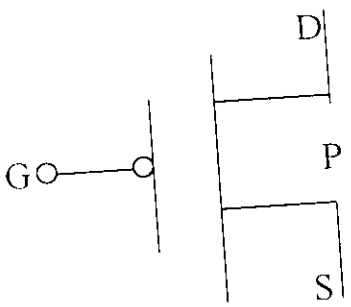
2.9 CONCLUSION

There are many technologies available, but due to the following advantages, CMOS Technology is preferred when compared to others.

- CMOS technology shows lowest power per gate
- CMOS technologies are cheapest to manufacture
- Design costs are cheap
- Delay is reduced
- Heat dissipation is less

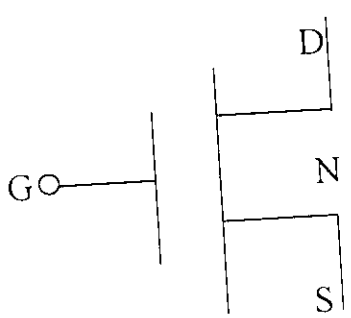
CMOS VLSI design today can be highly automated and almost all Hardware Descriptive Language (HDLs) tend to be increasingly favoured for describing VLSI system designs.

So the structure or behavior of a system can be captured in a Hardware Description Language. There are a wide variety of commercial and public domain Hardware Descriptive Languages such as VHDL, ELLA, Verilog® . In out project, the design is done in Verilog® because of its advantages and extended features as described in **Chapter - III**.



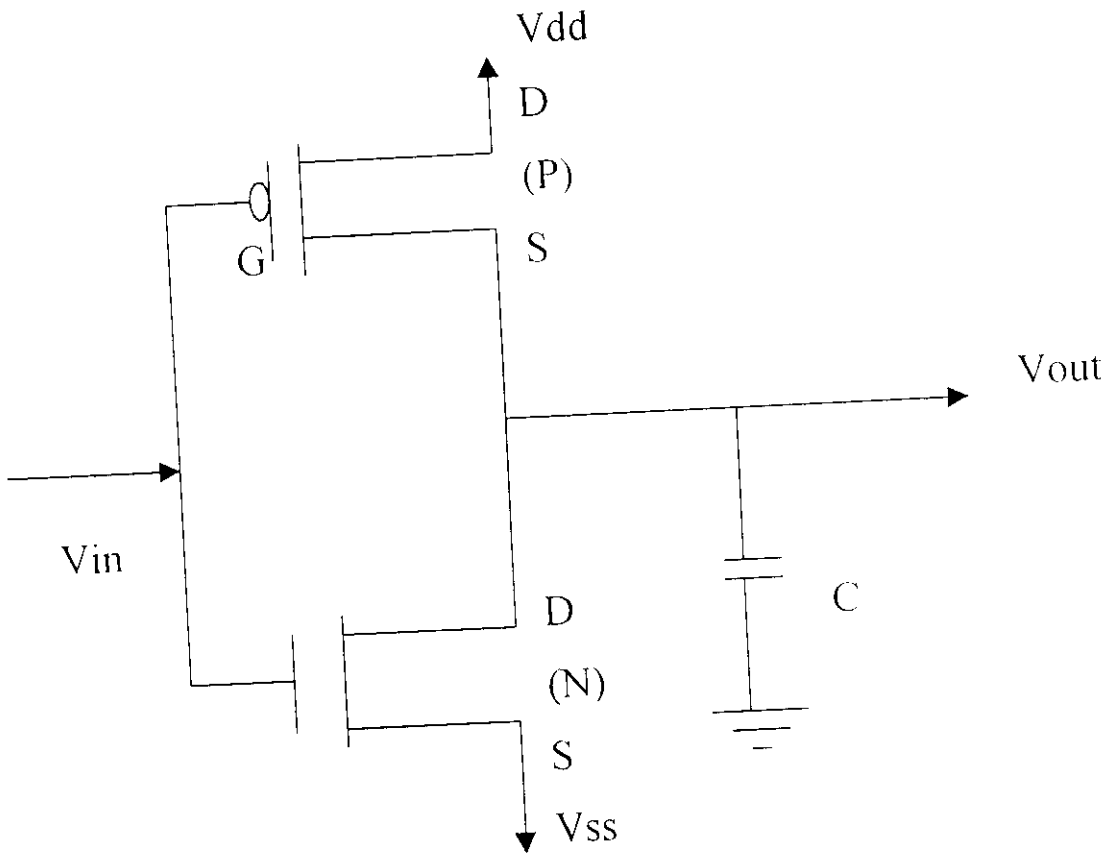
G - Gate
D - Drain
S - Source

Fig. 2.1 Symbol of p Channel Inverter.



G - Gate
D - Drain
S - Source

Fig. 2.2 Symbol of n Channel Inverter.



V_{dd} - Positive Supply Voltage

C - Load Capacitance

N - n Channel transistor

P - p Channel transistor

Vss - Ground potential

Fig. 2.3 CMOS Inverter

CHAPTER – III

VERILOG HDL

3.1. INTRODUCTION

From a very long time, programming languages such as C, C++, Pascal, etc., were used to describe sequential computer programs. Similarly, in digital design field, Hardware Descriptive languages are used to describe digital circuits. To compete with the developing electronics world, vendors built products with increase in its functionality, its performance, decrease in its cost, power consumption and size. On this way, the heart of our project is Verilog Hardware Descriptive language. The main reason for which VerilogHDL has become the hero of electronics world is that enables designer to describe large circuits in a very simple manner.

3.2. IMPORTANCE OF HDL

- Designs can be described at a very abstract level without choosing a specific fabrication technology.
- When a new technology emerges, designers need not redesign their circuit. The logic synthesis tool

available will create a new gate-level netlist and will optimize the circuit and timing for the new technology.

- By describing designs in HDL, functional verification can be done in the design cycle.
- Design bugs can be easily eliminated. Hence design cycle time is reduced significantly.
- Manufacturing and development cost is reduced.
- Designing with HDL is analogous to computer programming. This provides a concise representation of the design.
- All the complex digital circuits can be easily represented in HDL with sophisticated tools.

3.3. FEATURES OF VERILOG HDL

- Verilog HDL is a general-purpose hardware description language that is easy to learn and easy to use.
- It is similar in syntax to "C" programming language.

- A designer can define a hardware model in terms of switches, gates, RTL (Register Transfer Level) or behavioral code.
- A designer has to learn only one language for stimulus and hierarchical design.
- Almost all logic synthesis tools support VerilogHDL.
- Designing a chip in VerilogHDL is very easy.

3.4. DESIGN FLOW

In HDL design, specifications are written first. Specifications describe the functionality, interface and overall architecture of the digital circuit to be designed .At this point of time, the designer need not think about the implementation . A Behavioral description is then created to analyse the design in terms of functionality, performance and other issues. Behavioral description is written with HDL. A typical design flow is shown in Fig. 3.1.

The behavioral description is converted to an RTL description in a HDL. Here, the designer has to describe the data flow that will implement the desired digital circuit. Logic synthesis tools convert the RTL description to a gate-level

netlist. A gate-level netlist is a description of the circuit in terms of gates and connections between them. The gate level netlist is input to an "Automatic Place and Route" tool , which creates a layout . The layout is verified and fabricated on chip.

Thus, most digital design activity is concentrated on optimizing the RTL description of the circuit. By using this method design time is reduced. Sophisticated tools help the designer to convert the behavioral description to a final IC chip.

3.5. DESIGN SYNTHESIS

- Define the design requirements.
- Describe the design in VerilogHDL (formulate and code the design)
- Simulate the source code.
- Synthesize, optimize and fit (place and route) the design.
- Simulate the Post- layout (fit) design model.
- Program the device.

3.5.1. DEFINING THE DESIGN REQUIREMENTS

Before writing codes for the design, the design objectives and requirements must be clearly defined. They are:

- The function of the design.
- The clock to output times.
- The maximum frequency of operation.
- Critical paths.

3.5.2. DESCRIBING THE DESIGN IN VERILOG

The first step in designing is to decide upon the design methodology. The designer must use a good methodology to do efficient Verilog HDL based design.

3.5.3. SIMULATION OF SOURCE CODE

For large designs, simulating the source code with a Verilog HDL simulator will prove a time-efficient process. This type of simulation reduces our time of designing. With source code simulation, flaws can be detected easily in the design, allowing debugging and making corrections with the least possible impact to the schedule.

3.5.4. SYNTHESIZING

Synthesizing is a process by which netlists or equations are created from design description .In other words it can be described as taking a design description as input and producing output. (Logic equations or netlists).

3.5.5. OPTIMIZING

After writing the Verilog HDL code. To check whether it is an effective design, we go for optimization. Optimization is very essential because some forms of expressions may be mapped to logic resources more efficiently than the others.

3.5.6.FITTING

Fitting is defined as the process of taking the logic produced by the synthesis and optimization process and placing it into a logic device, transforming the logic to obtain the best fit.

3.5.7. PLACE AND ROUTE

The place and route technique has a large impact on the performance of designs. Propagation delay depends significantly

on routing delay. A good placement and route will place critical portions of a circuit close together to eliminate route delays.

3.5.8. SIMULATE THE POST LAYOUT DESIGN MODEL:

Even if we have performed pre-synthesis simulation, we have to simulate the design after it has been fitted (i.e., after placing and routing). A post layout simulation will enable us to verify not only the functionality of the design but also the timing, such as setup, clock-to-output and register-to-register times.

3.6. DESIGN METHODOLOGIES

There are two basic types of digital design methodologies available:

- Top-down design methodology
- Bottom-up design methodology

3.6.1. BOTTOM-UP DESIGN

This is the traditional method of electronic design.Each design is performed at the gate-level using the standard gates. With increasing complexity of new designs this approach is impossible to represent all complex circuts.These traditional bottom-up designs have to give wayto new structural ,hierarchial

design methods. Without these new design practices it would be impossible to handle complexity. Here , the building blocks are identified first.Using these, the higher level blocks and the top level blocks are designed.

3.6.2. TOP_DOWN DESIGN:

This method involves creating design hierarchies. The desired design style of all designers is the top-down design. A real top-down design allows early testing, easy change of technologies, a structured system design and offers many they advantage. Here, we define the top-level block and identify the sub blocks necessary to build the top block. The sub blocks known as the functional units has separate inputs and outputs and perform a particular function. The flow chart for the top down design process is shown in the Fig.3.2. The tool flow diagram is shown in Fig.3.3.

3.7. ADVANTAGES OF VERILOG HDL:

Though there are many Hardware Descriptive Languages, the advantages of Verily HDL over other languages are as follows:

- Less power consumption.
- Flexibility.
- Portability.
- Benchmark capabilities
- Quick time in marketing.
- Low cost.

3.8. PROGRAM STRUCTURE OF VERILOG:

Now let us enter into interesting part of verilog, programming structure of verilog. In verilog , the different blocks are defined as modules. A module provides the necessary functionality to the higher level block through its port interface.

MODULE DECLARATION:

module <module_name> (<module_terminal_list>);

// INPUT/ OUTPUT PORT DECLARATIONS:

output (output variables separated by commas);

input (input variables separated by commas);

// INTERNAL NET DECLARATIONS:

wire (list of variables separated by commas);

<module internals>

endmodule

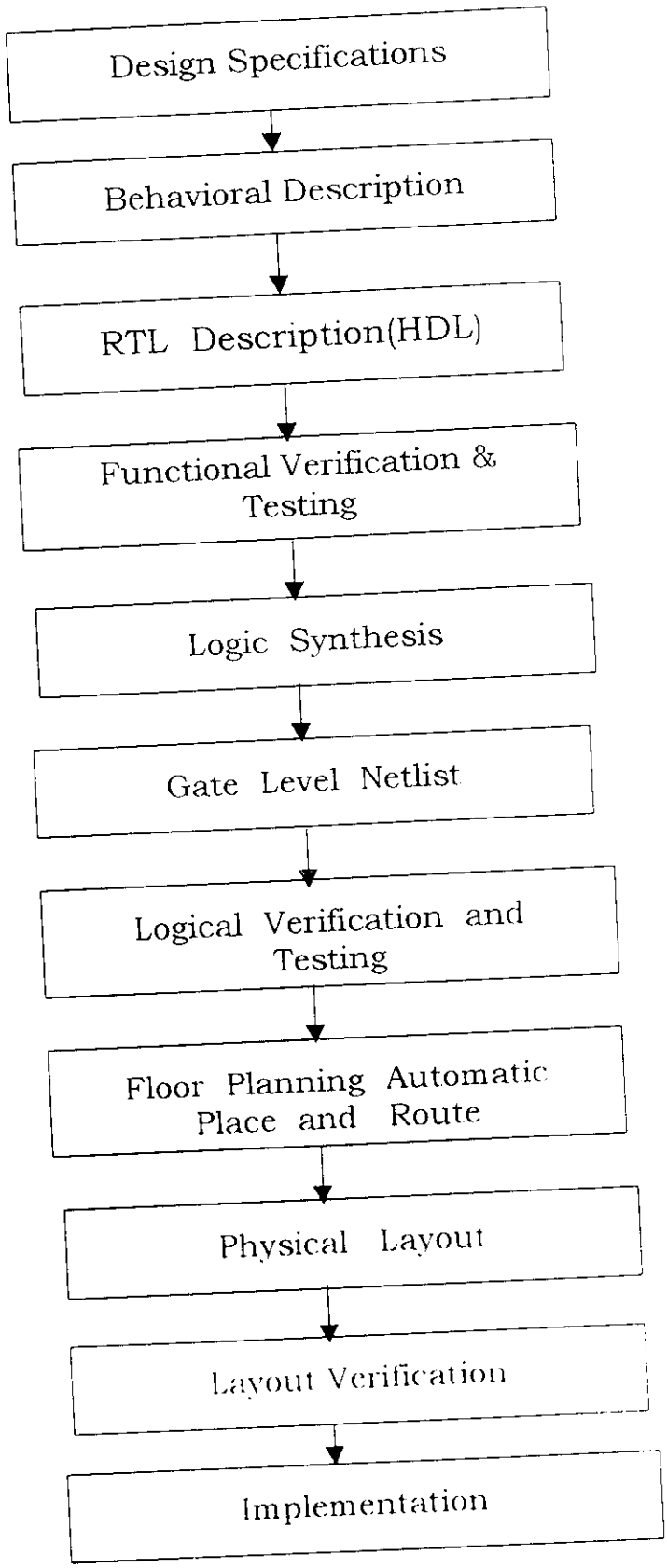


Fig. 3.1 Design Flow

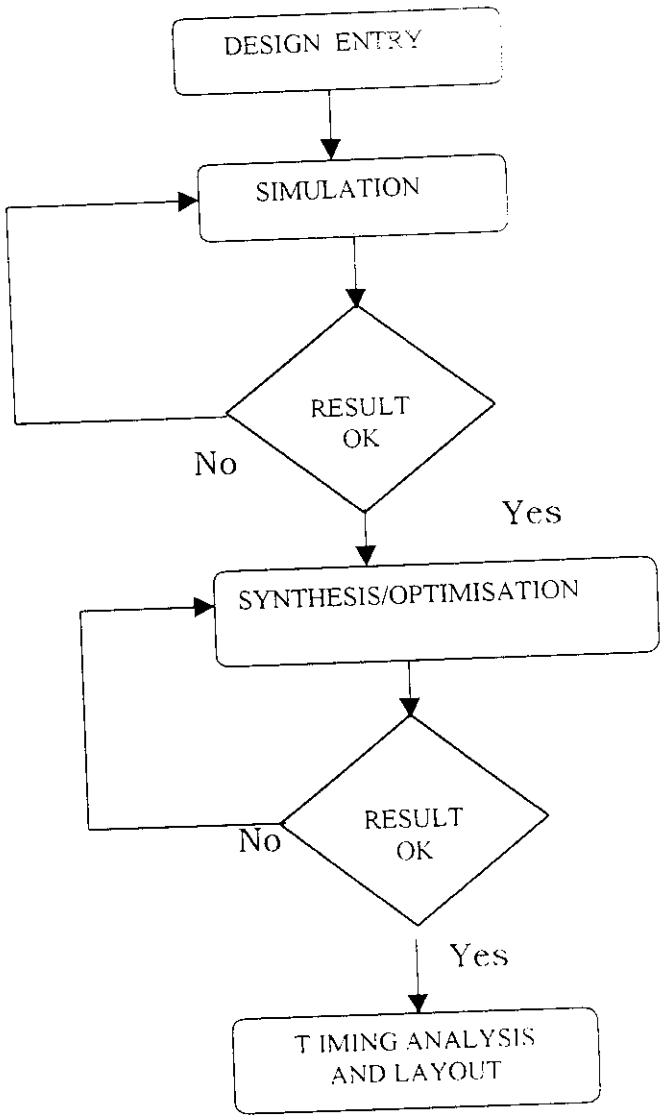


Fig 3.2 Top Down Design Approach

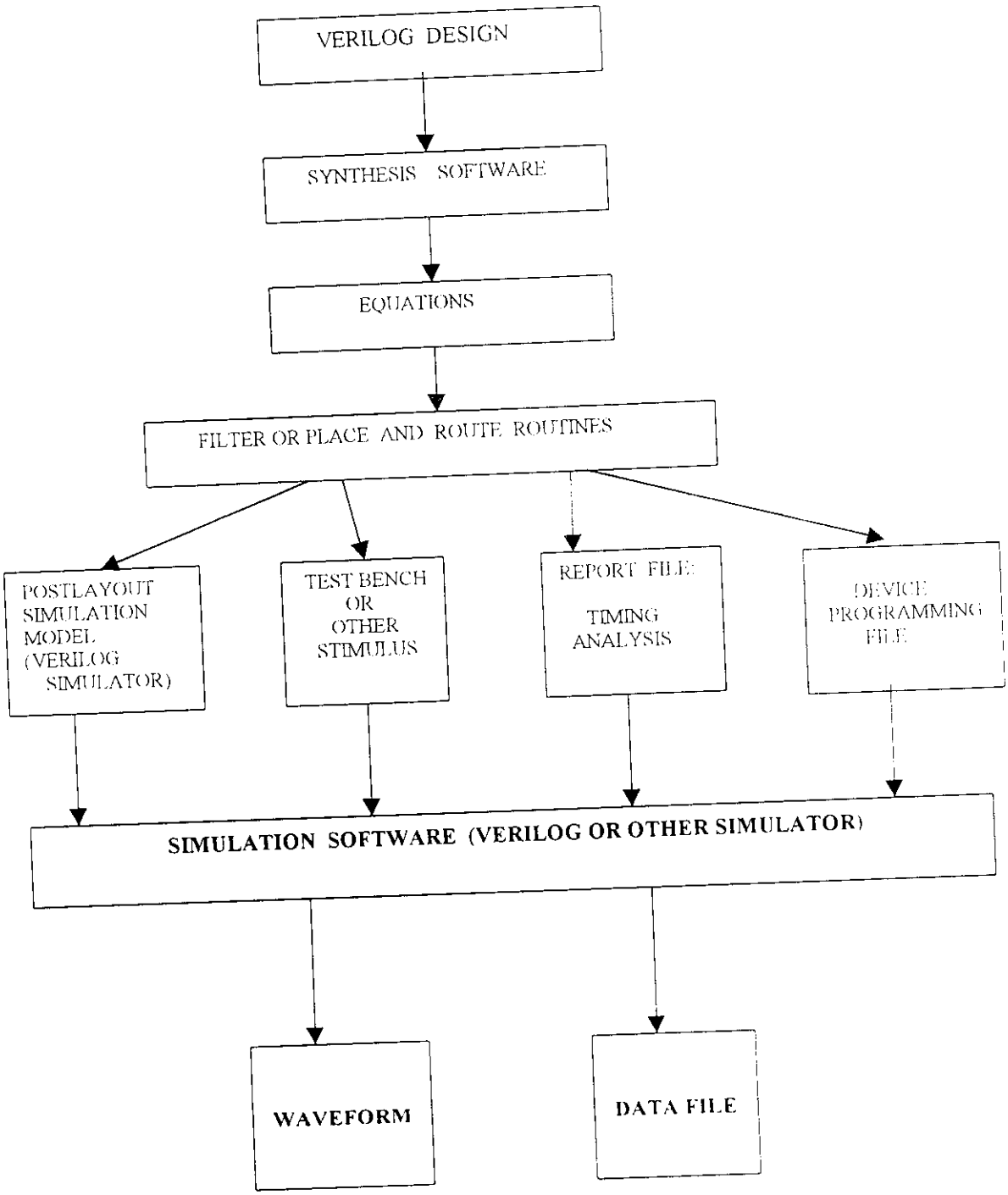


FIG.3.3 Tool Flow Diagram

CHAPTER – IV

CONTROL OF AUTOMATION

4.1. INTRODUCTION

The conventional methods and the present trends such as DDCS (Distributed Digital Control System) and the PLC (Programming Logic Controller) used now in automation control have their own merits and demerits. In order to overcome the various drawbacks of these control systems, we have implemented the automation control in VLSI technique using Verilog HDL as the hardware language.

4.2. GENERAL BLOCK DIAGRAM

The Fig 4.1 shows the general block diagram of the above mentioned control system. The various control parameters are first sensed by means of a Primary Sensing Element (PSE) .The output of the PSE is given to a Pulse Duration Modulator (PDM).

PDM also called as Pulse Width Modulator (PWM) is used for transmitting analog information by sampling the information in regular intervals. The width of each pulse is proportional to

the amplitude of the signal at that instant. PDM is used because demodulation is a very simple process, distortion is less and synchronization is easy.

The clock pulses, which are obtained from PDM, are given as input to the pins of the IC chip. The entire logic written in Verilog HDL is fabricated on a single chip. So the single IC chip does the entire control of the parameters. The output of the chip is stored in a latch. A BCD to Seven-segment driver is used to convert the controlled value obtained from the latch into a seven-segment code and the value is displayed.

4.3. CONTROL BLOCK DIAGRAM

The control block diagram to be implemented in Verilog is as shown in Fig. 4.2. The various components of the block diagram are:

- SENSOR
- COUNTER
- REGISTER
- COMPARATOR
- CONTROL CIRCUIT

Sensor, counter and the comparator block are enab'ed by means of enable signals at appropriate time intervals from the control circuit. Enable signals are in the form of two bit binary such as 00,01 and 10.

Sensor is used to sense the control parameters. The output of the sensor is in the form of clock pulses. The frequency of clock pulses is counted by using a counter. This value is stored in a register for further manipulations. The comparator has two inputs. One is the stored value from the register. The other input to the comparator is the reference input or the desired value. The function of the comparator is to compare both the inputs. The output of the comparator is two bit binary. It can be any one of the following.

- If the output of the comparator is "00" , then the reference input is same as the obtained value.

- If "01" is obtained as output from the comparator, then the reference input is more than the obtained value .So the value of the parameter must be increased.

- If the output of the comparator is "10", then the reference value is less than the obtained value. So the value of the control parameter is decreased so that the desired value and obtained value are same.

The required corrections and adjustments are implemented in other control block called the process block, which manipulates the error and controls the value in steps.

4.4. IMPLEMENTATION IN VERILOG HDL

The above proposal can be implemented in various other technologies available such as microprocessor, microcontroller, using programming languages, etc. If these are used, interfacing circuitry is necessary which is very costly and tedious to connect.

4.4.1. HALLMARKS OF VERILOG HDL

- Portability
- Flexibility
- Allows concise representation of the design.
- General-purpose hardware description language.
- Hardware model can be designed in any way.
- Analogous to "C" programming.
- Designing and fabricating a chip is very easy.

4.5. DESIGN BLOCK DIAGRAM

The design block diagram consists of the following blocks. The stimulus block is the top-level block, which is responsible for instantiating the various blocks under it. Then comes the process block which takes care of entire logic, manipulates the error and adjusts the values according to the desired value. Fig. 4.3 shows the design block diagram.

This block also controls three individual blocks namely pressure control temperature control and flow rate control blocks. Each of these individual blocks has a counter comparator and an error module.

4.6. DESIGN METHODOLOGY

There are two types of design methodologies followed in Verilog HDL.

- Bottom up design methodology.
- Top down design methodology.

The design is written using top down design methodology. Here, the top block is designed first and then the lower level blocks necessary to build the top-level block is identified. The

sub blocks have separate input and output ports and they perform a particular function.

In this design, the top-level block is the stimulus block, which simulates the entire logic. It has two lower level blocks:

- TOTAL BLOCK
- PROCESS BLOCK

The total block is further divided into 3 sub blocks namely,

- PRESSURE CONTROL
- TEMPERATURE CONTROL
- FLOW RATE CONTROL

Each of these individual sub blocks has 3 functional blocks namely,

- COUNTER BLOCK
- COMPARATOR BLOCK
- ERROR BLOCK

The error block of all the individual sub blocks are connected to the process block which is concerned with correction of values. This block is also responsible for maintaining the value of the parameters according the desired value.

4.7. ADVANTAGES

There are numerous advantages of implementing the design in Verilog HDL .A few of that are listed below:

- Design bugs can be easily eliminated
- Less power consumption.
- Time delay is reduced (less than 1 nano second)
- Miniature in size.
- When a new technology emerges. redesigning the entire circuit is not necessary
- Interfacing circuitry is not required
- The entire design can be implemented on a single chip.
- Development and manufacturing cost is very less.
- Functional verification can be done at every stage.
- Maintenance free.

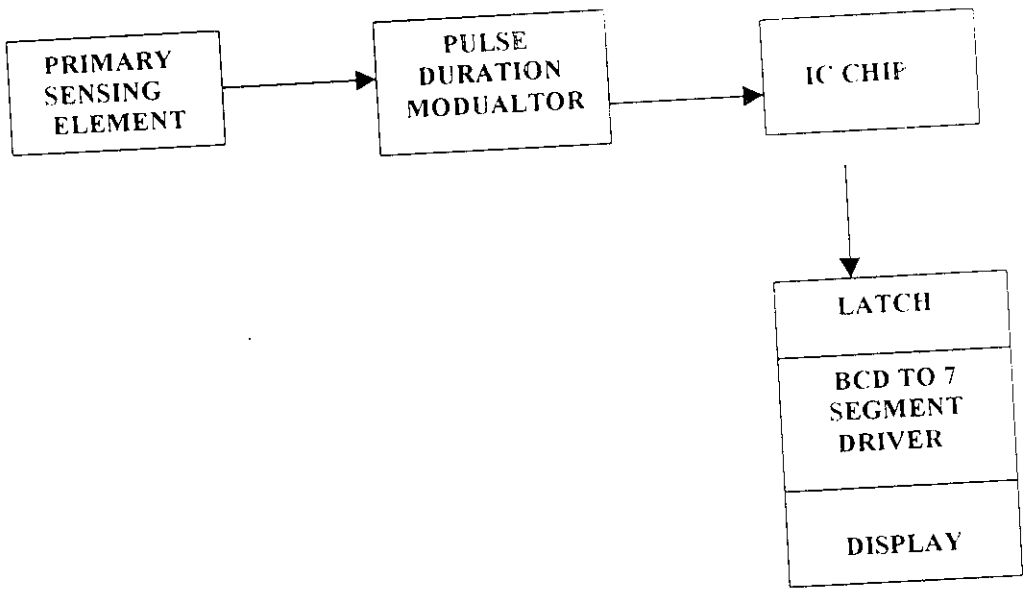


Fig.4.1. General Block Diagram

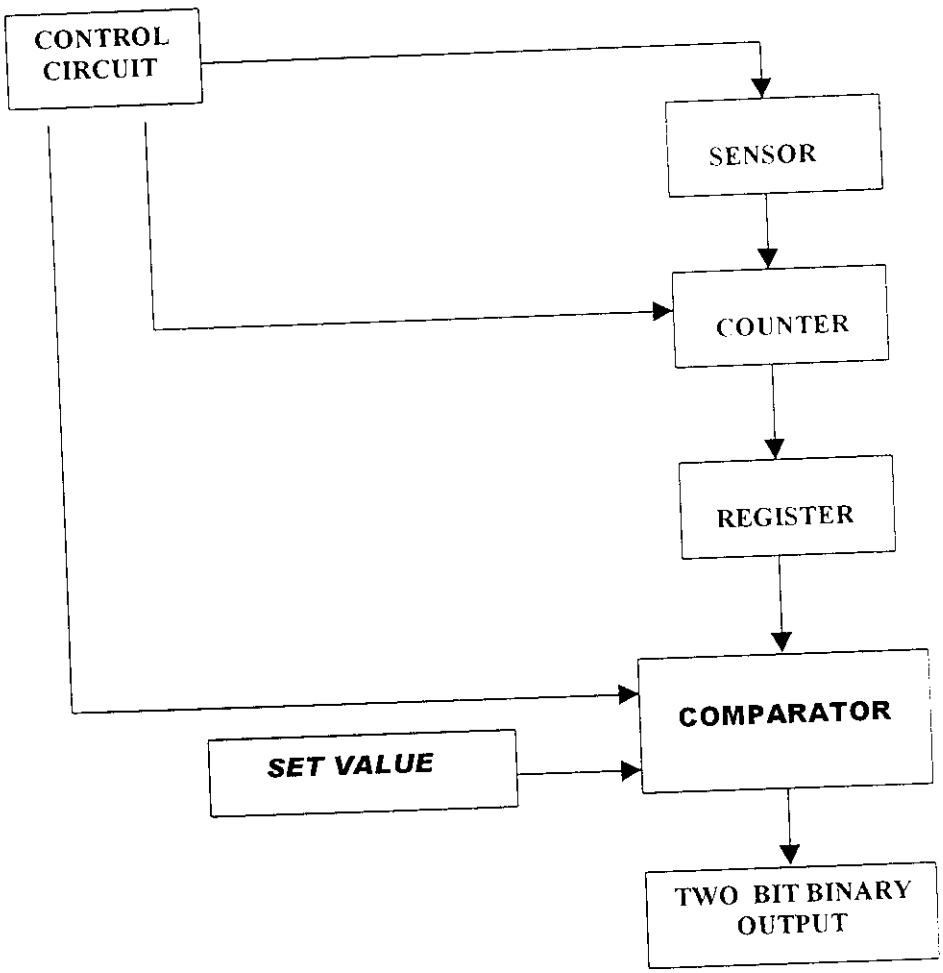


Fig. 4.2. Control block diagram

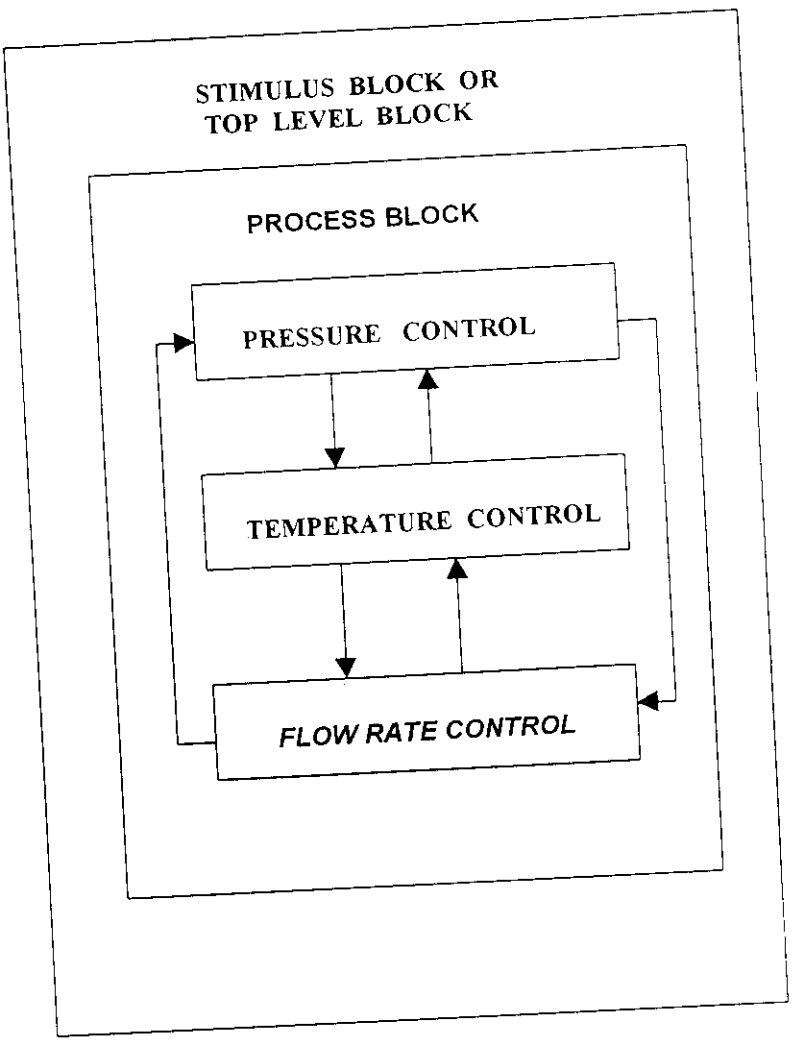


Fig. 4.3. Design Block Diagram

CHAPTER - V

SOFTWARE DESCRIPTION

5.1 INTRODUCTION

The project mainly deals with controlling the parameters and maintaining the value of these parameters within a tolerable value without any fluctuations. The various blocks as indicated in a control block diagram are divided into various modules. Each block performs specific functions for implementation in verilog HDL.

The program structure of verilog HDL is analogous to 'C' programming. The various modules have separate input and output variables and finally all the modules are instantiated from the stimulus block. The verilog HDL code is developed and kept at the end of this chapter.

5.2 REPRESENTATION

There are three types of representation available in verilogHDL.

- Structural Representation
- Behavioral Representation
- Physical Representation

Here the code is written using behavioral representation. The various modules, input and output ports and their individual functions are explained below.

5.3. DESCRIPTION OF THE SOFTWARE

The various modules in the program are as follows:

- Stimulus block
- Process block
- Total block
- Temperature control block
- Pressure control block
- Flow control block

5.3.1. STIMULUS BLOCK

Stimulus block is responsible for instantiating the other blocks. The nets required for the logic are declared in this block. Nets represent connection between hardware elements. This is similar to a data type in C language. The nets required for other blocks such as counter, comparator and error block is declared as vectors (multiple bit width). Here all the nets are declared as 8 bit width variables.

This block instantiates the process and total block. Along with the module declaration , input and output ports of the block are declared.

An initial block is included so that the above process get executed only once during the entire simulation and the simulation gets terminated at 3000 time units.

5.3.2. TOTAL BLOCK

In the total block, module instances are created for the various modules such as

- Temperature control module
- Pressure control module
- Flow rare control module

When instances are created the corresponding input and output port must be specified.

All the three modules mentioned above perform the following functions.

- Counts the clock pulses
- Stores it in a register
- Compares the value of the register with the set value.

- Manipulates the error.
- Adjust the values if necessary step by step such that the value obtained is within the tolerable limit.

For illustration, here we explain the temperature control module and its sub blocks.

5.4. TEMPERATURE CONTROL MODULE

When the module is declared the input and output ports must be specified along with the module. The input ports are sensor input and state variable. The output ports are the comparator output with a vector of two-bit width and the error value, which is of 8-bit width. The tolerance value is declared as eight bit registers. Registers are data storage element, which will retain the value until another value is paste in them. An instance for counter module is created along with the input and output ports.

Depending upon the parameters to be controlled, thew tolerance values are set. Here for temperature, the tolerance is set as(+/-) 0.1 than the set value. The internal details of temperature control module is shown in Fig. 5.4.

5.5. ENABLE SIGNAL

Each and every block should be enabled at appropriate time intervals, the various sub blocks are

- Counter
- Comparator
- Error

The counter block is enabled at 1 time unit and is disabled at 15 time units and the error blocks are set after 10 time units. Usually time units will be in nano seconds.

5.6. COUNTER BLOCK

The output of the counter block is declared as 8 bit width variable Q. The input to the counter block is the clock pulses. So the frequency of the clock pulse is counted and the value of the counter block for the temperature module can be seen in "T_coun_out" in the simulation diagram shown in Fig.5.2, which is of 8 bits.

The value of the counter is stored in a register and is given as input to the comparator block. The output of the counter is stored in the variable count.

5.7. COMPARATOR BLOCK

The input to the comparator are the set value and the output of the count. The tolerance values also must be given as input to the comparator. Here if the set value is equal to the counter output or if the counter output is greater than or equal to the lowest tolerable value and less than or equal to the highest tolerable value, then the output of the comparator is '00'(two bit binary).

If the value in variable count is less than the lowest tolerable value, then the output of the comparator is '01' else the output of the comparator is '10'. The output obtained from the comparator which is a 2 bit binary can be seen in "stimulus. T_com_out in the simulation diagram shown in Fig. 5.2.

5.8. ERROR BLOCK

The inputs given to the error block are the set value, count which are vectors of 8 bit wide. The 2 bit binary output obtained from the comparator is also given as input.

If the comparator output is'00', then as seen from the logic, the set value and the obtained value are same, so there is no error.

But if the output of the comparator is'01', then the error will be the difference between the set value and the obtained value which is stored in variable count.

Error = Set value - Count value

In order to make both the values equal the value of the temperature must be increased.

If '10' is obtained as the output of the comparator, then the error will be the difference between count value and set value.

Error = Count value - set value

Here in order to make the error zero the value of temperature must be decreased.

Here the error is reduced in steps and finally the comparator output will be obtained as '00' when both values are made equal.

The error obtained from the temperature block is seen in 'Stimulus.T_error' in the simulation diagram shown in Fig.5.2.

After the values are adjusted and they are maintained according to the design requirement, the output of the comparator will be'00' as seen in the simulation diagram.

The design mentioned above is similar for the other two modules pressure and flow rate also.

5.9. PROCESS BLOCK

This block is the main controller for all the other blocks. The input ports to the process block are the error variables from pressure, temperature, flowrate control blocks and the output of the comparator from all the blocks.

At the time of starting, the entire program is instantiated only when start is enabled. Once the block is triggered, the start is disabled. As seen in Verilog HDL code the blocks are declared as tasks. The values passed to tasks are registers. But the internal declarations are net variables. So user defined variables are created to transfer these values for further processing.

When start gets enabled, T_fin will always be 1. Once the cycle execution gets over, it becomes zero. The values from the net are transferred to registers. When T_fin is '1' signals get generated. When this value becomes zero after 2 time units the other blocks get triggered.

A pulse, which is edge triggered, is given to the temperature, pressure and flow rate control blocks simultaneously. Once the blocks get instantiated the signal is disabled. Depending upon the output of the comparator the parameters gets manipulated as shown below.

When the output is '00', the parameter value will remain the same.

When the output of the comparator is obtained as '01', then ,

Parameter value = Actual value – Error

If '10' is obtained as output, then

Parameter value = Actual value + Error

Finally the values are displayed in the report file kept at end of this chapter.

SOURCE CODE

// STIMULUS BLOCK OR THE TOP LEVEL BLOCK//

```
module stimulus;
  wire T_sen_in,P_sen_in,F_sen_in;
  wire [7:0] T_coun_out,P_coun_out,F_coun_out;
  wire [1:0] T_com_out,P_com_out,F_com_out;
  wire [7:0] T_error,P_error,F_error;
  wire T_fin,P_fin,F_fin,T_start,P_start,F_start;
  process p(T_start,P_start,F_start,T_sen_in,P_sen_in,F_sen_in,
          T_com_out,P_com_out,F_com_out,
          T_error,P_error,F_error,T_fin,P_fin,F_fin);
  total T(T_fin,P_fin,F_fin,T_com_out,P_com_out,F_com_out,
          T_coun_out,P_coun_out,F_coun_out,
          T_error,P_error,F_error,T_sen_in,P_sen_in,F_sen_in,
          T_start,P_start,F_start);

  initial
   #3000 $finish;
  endmodule
```

// END OF THE STIMULUS MODULE //

// PROCESS BLOCK STARTS //

```
module
process(T_start,P_start,F_start,T_sen_in,P_sen_in,F_sen_in,
        t_com_out,p_com_out,f_com_out,
        t_error,p_error,f_error,T_fin,P_fin,F_fin);
 output T_start,P_start,F_start,T_sen_in,P_sen_in,F_sen_in;
 input T_fin,P_fin,F_fin;
 input [1:0] t_com_out,p_com_out,f_com_out;
 input [7:0] t_error,p_error,f_error;
 //wire T_sen_in,T_fin,t;
 reg T_sen_in,P_sen_in,F_sen_in,T_start,P_start,F_start;
 reg [1:0] T_com_out,P_com_out,F_com_out;
 reg [7:0] T_error,P_error,F_error;
 // integer t_c,p_c,f_c,t,p,f;
  //signal S(T_sen_in,T_fin,t);

//INITIALISATION OF VARIABLES//
 initial
  fork
   T_start=1'b1;
   P_start=1'b1;
   F_start=1'b1;
   #2 T_start=1'b0;
   #2 P_start=1'b0;
   #2 F_start=1'b0;
  join
```

```
always @(negedge T_fin)
 begin
  T_com_out=t_com_out;
  T_error=t_error;
 end
 always @(negedge P_fin)
  begin
   P_com_out=p_com_out;
   P_error=p_error;
  end
  always @(negedge F_fin)
   begin
    F_com_out=f_com_out;
    F_error=f_error;
   end

  initial
     begin
      T_sen_in=1'b0;
      P_sen_in=1'b0;
      F_sen_in=1'b0;
     end

     initial
      begin
```

```
t=2;
p=35;
f=50;
end

always //@(posedge T_fin)
 wait(T_fin) #(t) T_sen_in=~T_sen_in;
always //@(posedge P_fin)
 wait(P_fin) #(p) P_sen_in=~P_sen_in;
always //@(posedge F_fin)
  wait(F_fin) #(f) F_sen_in=~F_sen_in;
always @(negedge T_fin)
 //fork
  begin
  #2 temp(T_com_out,T_error,t);
  #2 T_start=1'b1;
  #2 T_start=1'b0;
  end
  always @(negedge P_fin)
   begin
   #2 press(P_com_out,P_error,p);
   #2 P_start=1'b1;
   #2 P_start=1'b0;
   end
   always @(negedge F_fin)
```

```
begin
#2 flow(F_com_out,F_error,f);
#2 F_start=1'b1;
#2 F_start=1'b0;
end

task temp;
 input [1:0] T_com_out;
 input [7:0] T_error;
 inout [31:0] t;
 begin
   case(T_com_out)
   2'b00 : t=t+0;
   2'b01 : t=t-T_error;
   default: t=t+T_error;
   endcase
   $display("the value of t is %d",t);
  end
endtask

task press;
  input [1:0] P_com_out;
  input [7:0] P_error;
  inout [31:0] p;
  begin
```

```
  case(P_com_out)
  2'b00 : p=p+0;
  2'b01 : p=p-P_error;
  default: p=p+P_error;
  endcase
  $display("  p is %d",p);
 end
endtask

task flow;
  input [1:0] F_com_out;
  input [7:0] F_error;
  inout [31:0] f;
  begin
    case(F_com_out)
    2'b00 : f=f+0;
    2'b01 : f=f-F_error;
    default: f=f+F_error;
    endcase
    $display("  f is %d",f);
   end
  endtask
  endmodule
```

```
// TOTAL BLOCK STARTS//
module total(t_fin,p_fin,f_fin,t_com_out,p_com_out,f_com_out,
        t_coun_out,p_coun_out,f_coun_out,
        t_error,p_error,f_error,
        t_sen_in,p_sen_in,f_sen_in,t_start,p_start,f_start);
 output [1:0] t_com_out,p_com_out,f_com_out;
 output [7:0] t_coun_out,p_coun_out,f_coun_out;
 output [7:0] t_error,p_error,f_error;
 output t_fin,p_fin,f_fin;
 input t_sen_in,p_sen_in,f_sen_in,t_start,p_start,f_start;

//INSTANCES ARE CREATED FOR THE SUD MODULES//
  temp_control
t1(t_fin,t_com_out,t_coun_out,t_error,t_sen_in,t_start);
  press_control
p1(p_fin,p_com_out,p_coun_out,p_error,p_sen_in,p_start);
  flow_control
f1(f_fin,f_com_out,f_coun_out,f_error,f_sen_in,f_start);
endmodule

//TEMPERATURE CONTROL MODULE//
module temp_control(finish,out,q,err,sen_in,start);
  input sen_in,start;
  output [1:0] out;
  output [7:0] q,err;
```

```
output finish;
wire [7:0] q;
reg coun_ena,finish;
reg [7:0] set,low_tol,high_tol,err;
reg [1:0] out;

// COUNTER MODULE IS INSTANTIATED//
 counter_t c1(q, sen_in,coun_ena);
 initial
  begin
  set=8'd8;
  low_tol=set-8'd2;
  high_tol=set+8'd2;
  end
  always @(posedge start)
   begin
    finish=1'b1;
   #1 coun_ena=1'b1;
   #75 coun_ena=1'b0;
   #15 comparator(out,q,set,low_tol,high_tol);
   #10 error(err,set,q,out);
   #2 finish=1'b0;
   end
```

//COMPARATOR MODULE//

```
task comparator;
  output [1:0] out;
  input [7:0] count;
  input [7:0] set,low_tol,high_tol;
  begin
   if(set==count)
     out=2'b00;
   else if((count>=low_tol)&&(count<=high_tol))
     out=2'b00;
   else if(count<low_tol)
     out=2'b01;
   else
     out=2'b10;
   end
  endtask
```

//ERROR MODULE //

```
  task error;
    output [7:0] err;
    input [7:0] set,count;
    input [1:0] com_out;
    begin
```

```
if (com_out==00)
   err=8'd0;
 else if(com_out==01)
   err=set-count;
 else
   err=count-set;
 end
endtask
endmodule
```

//COUNTER MODULE //

```
module counter_t(Q,clock,start);
  output [7:0] Q;
  input clock,start;
  reg [7:0] Q;
  always @(posedge start)
   begin
    Q=8'd0;
    while(start)
     @(clock) Q=(Q+1);
    end
  endmodule
```

// PRESSURE CONTROL MODULE //

```
module press_control(finish,out,q,err,sen_in,start);
  input sen_in,start;
  output [1:0] out;
  output [7:0] q,err;
  output finish;
  wire [7:0] q;
  reg coun_ena,finish;
  reg [7:0] set,low_tol,high_tol,err;
  reg [1:0] out;
  counter_p c1(q, sen_in,coun_ena);
  initial
   begin
   set=8'd8;
   low_tol=set-8'd2;
   high_tol=set+8'd2;
   end
   always @(posedge start)
    begin
    finish=1'b1;
    #1 coun_ena=1'b1;
    #75 coun_ena=1'b0;
    #15 comparator(out,q,set,low_tol,high_tol);
    #10 error(err,set,q,out);
```

```
  #2 finish=1'b0;
  end
```

//COMPARATOR BLOCK//

```
task comparator;
  output [1:0] out;
  input [7:0] count;
  input [7:0] set,low_tol,high_tol;
  begin
   if(set==count)
     out=2'b00;
   else if((count>=low_tol)&&(count<=high_tol))
     out=2'b00;
   else if(count<low_tol)
     out=2'b01;
   else
     out=2'b10;
   end
  endtask
```

// ERROR BLOCK//

```
  task error;
    output [7:0] err;
```

```
input [7:0] set,count;
input [1:0] com_out;
begin
 if (com_out==00)
    err=8'd0;
 else if(com_out==01)
    err=set-count;
 else
    err=count-set;
 end
endtask
endmodule
```

// COUNTER MODULE//

```
module counter_p(Q,clock,start);
   output [7:0] Q;
   input clock,start;
   reg [7:0] Q;
   always @(posedge start)
    begin
     Q=8'd0;
     while(start)
      @(clock) Q=(Q+1);
     end
  endmodule
```

//FLOW RATE CONTROL BLOCK //

```
module flow_control(finish,out,q,err,sen_in,start);
  input sen_in,start;
  output [1:0] out;
  output [7:0] q,err;
  output finish;
  wire [7:0] q;
  reg coun_ena,finish;
  reg [7:0] set,low_tol,high_tol,err;
  reg [1:0] out;
  counter_f c1(q, sen_in,coun_ena);
  initial
   begin
   set=8'd8;
   low_tol=set-8'd2;
   high_tol=set+8'd2;
   end
   always @(posedge start)
    begin
    finish=1'b1;
    #1 coun_ena=1'b1;
    #75 coun_ena=1'b0;
    #15 comparator(out,q,set,low_tol,high_tol);
```

```
  #10 error(err,set,q,out);
  #2 finish=1'b0;
  end
```

//COMPARATOR BLOCK //

```
task comparator;
 output [1:0] out;
 input [7:0] count;
 input [7:0] set,low_tol,high_tol;
 begin
  if(set==count)
    out=2'b00;
  else if((count>=low_tol)&&(count<=high_tol))
    out=2'b00;
  else if(count<low_tol)
    out=2'b01;
  else
    out=2'b10;
  end
endtask
```

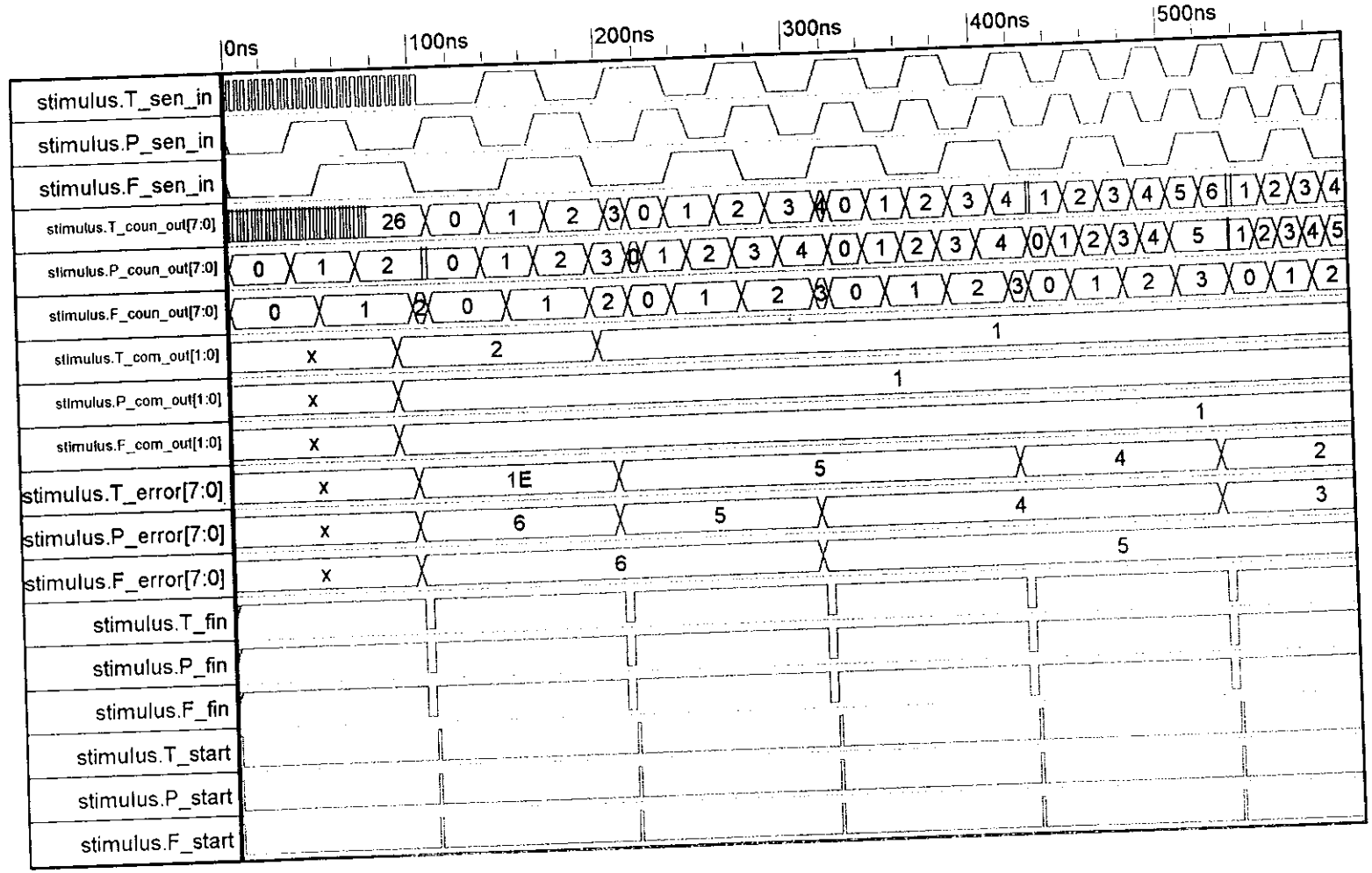
// ERROR MODULE //

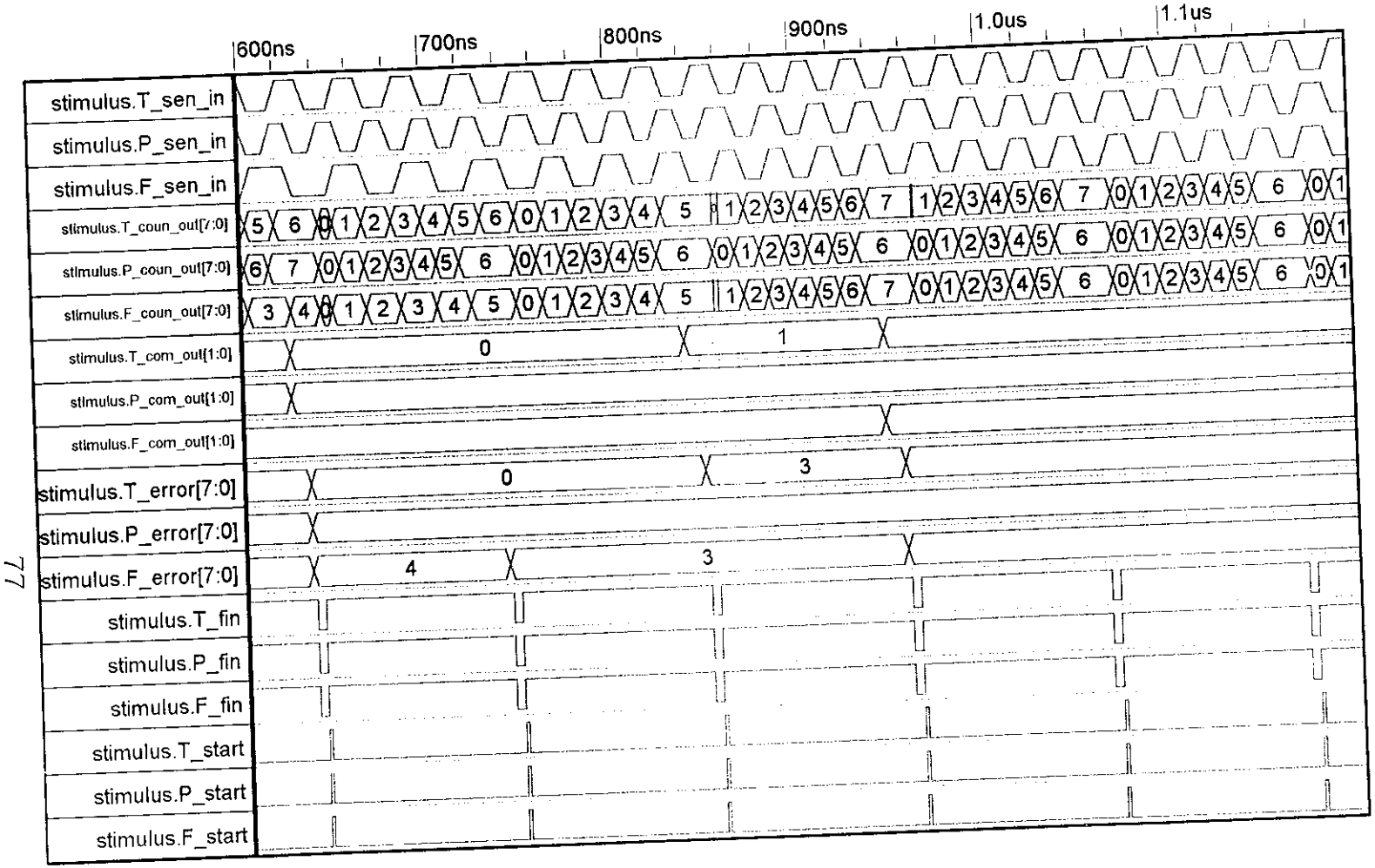
```
task error;
  output [7:0] err;
```

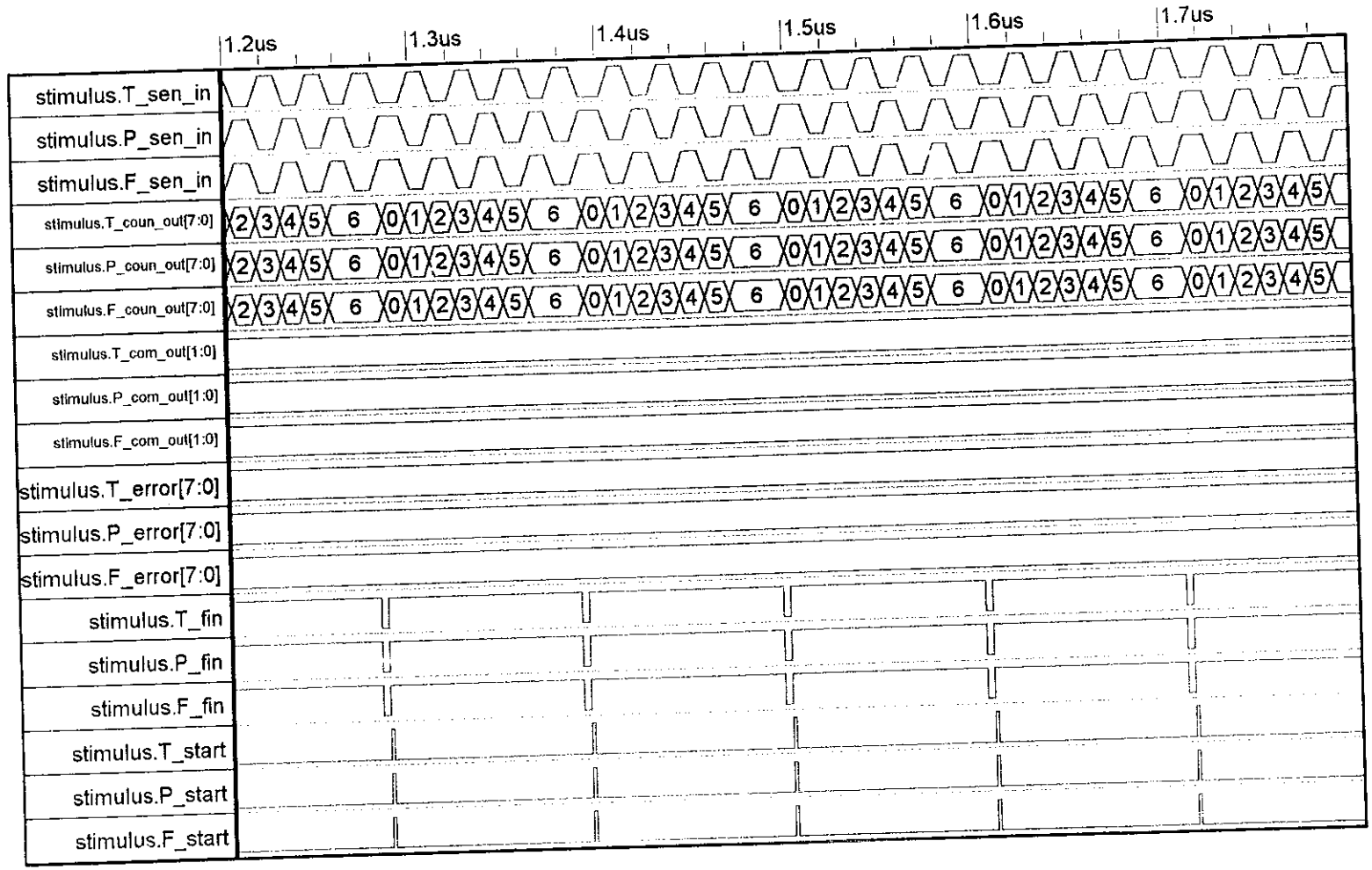
```
input [7:0] set,count;
input [1:0] com_out;
begin
 if (com_out==00)
    err=8'd0;
 else if(com_out==01)
    err=set-count;
 else
    err=count-set;
 end
endtask
endmodule
```

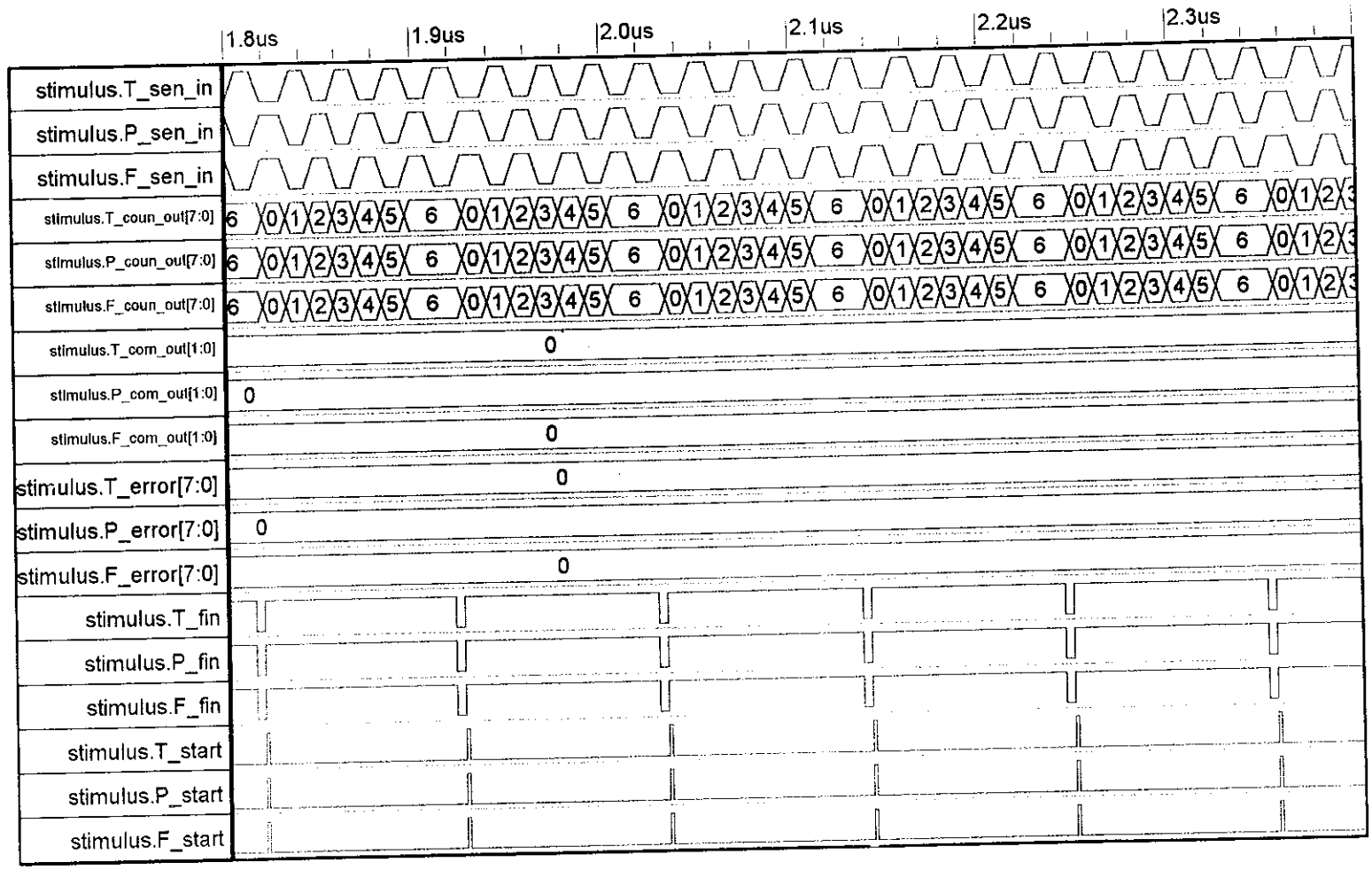
// COUNTER MODULE//

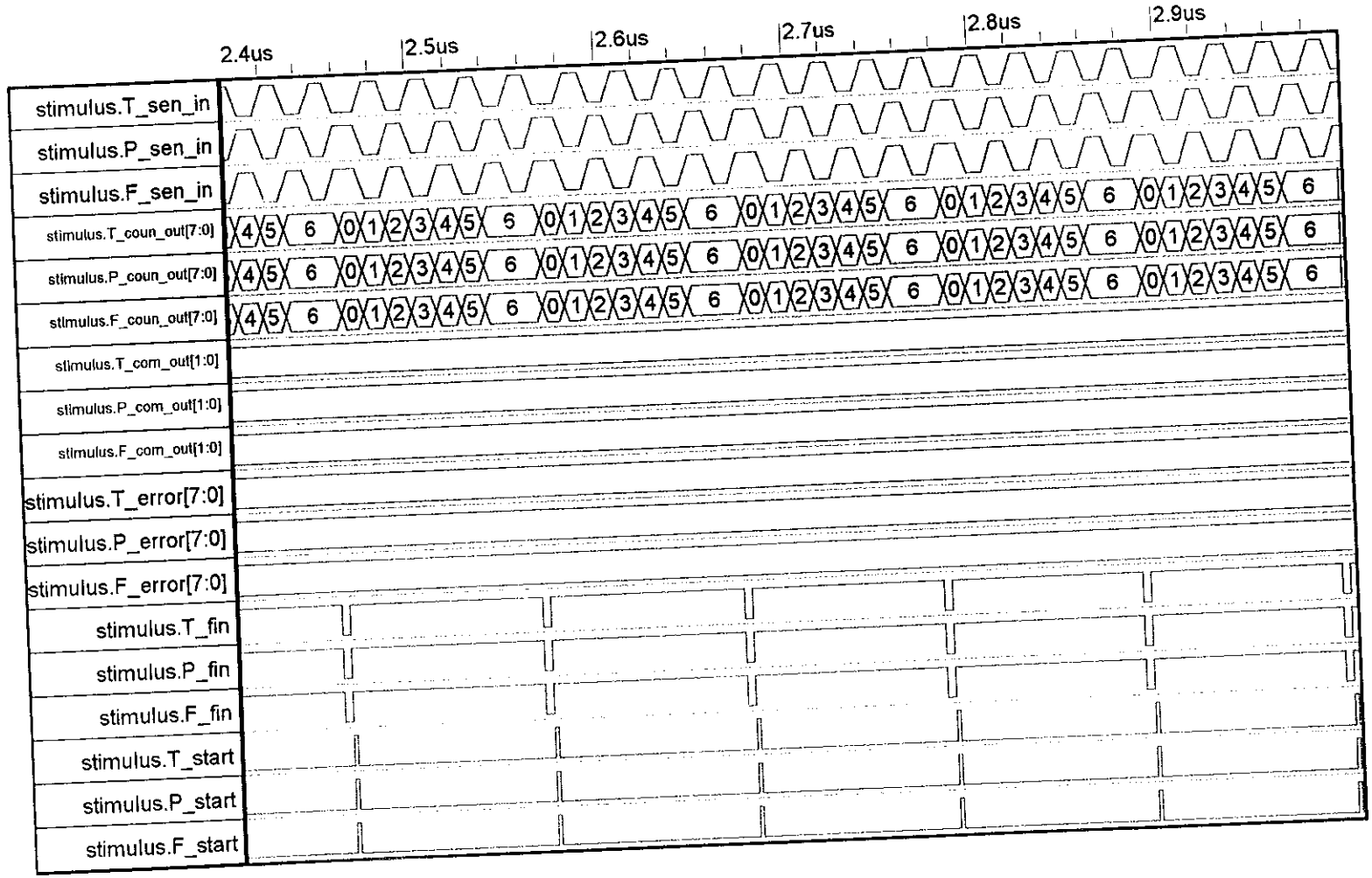
```
module counter_f(Q,clock,start);
   output [7:0] Q;
   input clock,start;
   reg [7:0] Q;
    always @(posedge start)
    begin
     Q=8'd0;
     while(start)
      @(clock) Q=(Q+1);
     end
  endmodule
```











Running...

At slmulation time 105

The input freqency from Temperature sensor : 31.250000 Mnz

The input freqency from Pressure sensor : 34.482759 Mhz

The input freqency from Flow sensor : 22.727273 Mhz

At slmulation time 212

The input freqency from Temperature sensor : 37.037037 Mhz

The input freqency from Pressure sensor : 41.666667 Mhz

The input freqency from Flow sensor : 26.315789 Mhz

At slmulation time 319

The input freqency from Temperature sensor : 45.454545 Mhz

The input freqency from Pressure sensor : 50.000000 Mhz

The input freqency from Flow sensor : 30.303030 Mhz

At slmulation time 426

The input freqency from Temperature sensor : 55.555556 Mhz

The input freqency from Pressure sensor : 62.500000 Mhz

The input freqency from Flow sensor : 35.714286 Mhz

At slmulation time 533

The input freqency from Temperature sensor : 62.500000 Mhz

The input freqency from Pressure sensor : 76.923077 Mhz

The input freqency from Flow sensor : 43.478261 Mhz

At slmulation time 640

The input freqency from Temperature sensor : 62.500000 Mhz

The input freqency from Pressure sensor : 76.923077 Mhz

The input freqency from Flow sensor : 52.631579 Mhz

At slmulation time 747

The input freqency from Temperature sensor : 62.500000 Mhz

The input freqency from Pressure sensor : 76.923077 Mhz

The input freqency from Flow sensor : 62.500000 Mhz

At slmulation time 854

The input freqency from Temperature sensor : 76.923077 Mhz

The input freqency from Pressure sensor : 76.923077 Mhz

The input freqency from Flow sensor : 76.923077 Mhz

At slmulation time 961

The input freqency from Temperature sensor : 76.923077 Mhz

The input freqency from Pressure sensor : 76.923077 Mhz

The input freqency from Flow sensor : 76.923077 Mhz

At slmulation time 1068

The input freqency from Temperature sensor : 76.923077 Mhz

The input freqency from Pressure sensor : 76.923077 Mhz

The input freqency from Flow sensor : 76.923077 Mhz

At slmulation time 1175

The input freqency from Temperature sensor : 76.923077 Mhz

The input freqency from Pressure sensor : 76.923077 Mhz

The input freqency from Flow sensor : 76.923077 Mhz

At slmulation time 1282

The input freqency from Temperature sensor : 76.923077 Mhz

The input freqency from Pressure sensor : 76.923077 Mhz

The input freqency from Flow sensor : 76.923077 Mhz

At slmulation time 1389

The input freqency from Temperature sensor : 76.923077 Mhz

The input freqency from Pressure sensor : 76.923077 Mhz

The input freqency from Flow sensor : 76.923077 Mhz

At slmulation time 1496

The input freqency from Temperature sensor : 76.923077 Mhz

The input freqency from Pressure sensor : 76.923077 Mhz

The input freqency from Flow sensor : 76.923077 Mhz

At slmulation time 1603

The input freqency from Temperature sensor : 76.923077 Mhz

The input freqency from Pressure sensor : 76.923077 Mhz

The input freqency from Flow sensor : 76.923077 Mhz

At slmulation time 1710

The input freqency from Temperature sensor : 76.923077 Mhz

The input freqency from Pressure sensor : 76.923077 Mhz

The input freqency from Flow sensor : 76.923077 Mhz

At slmulation time 1817

The input freqency from Temperature sensor : 76.923077 Mhz

The input freqency from Pressure sensor : 76.923077 Mhz

The input freqency from Flow sensor : 76.923077 Mhz

At slmulation time 1924

The input freqency from Temperature sensor : 76.923077 Mhz

The input freqency from Pressure sensor : 76.923077 Mhz

The input freqency from Flow sensor : 76.923077 Mhz

At slmulation time 2031

The input freqency from Temperature sensor : 76.923077 Mhz

The input freqency from Pressure sensor : 76.923077 Mhz

The input freqency from Flow sensor : 76.923077 Mhz

At slmulation time 2138

The input freqency from Temperature sensor : 76.923077 Mhz

The input freqency from Pressure sensor : 76.923077 Mhz

The input freqency from Flow sensor : 76.923077 Mhz

At slmulation time 2245

The input freqency from Temperature sensor : 76.923077 Mhz

The input freqency from Pressure sensor : 76.923077 Mhz

The input freqency from Flow sensor : 76.923077 Mhz

At slmulation time 2352

The input freqency from Temperature sensor : 76.923077 Mhz

The input freqency from Pressure sensor : 76.923077 Mhz

The input freqency from Flow sensor : 76.923077 Mhz

At slmulation time 2459

The input freqency from Temperature sensor : 76.923077 Mhz

The input freqency from Pressure sensor : 76.923077 Mhz

The input freqency from Flow sensor : 76.923077 Mhz

At slmulation time 2566

The input freqency from Temperature sensor : 76.923077 Mhz

The input freqency from Pressure sensor : 76.923077 Mhz

The input freqency from Flow sensor : 76.923077 Mhz

At slmulation time 2673

The input freqency from Temperature sensor : 76.923077 Mhz

The input freqency from Pressure sensor : 76.923077 Mhz

The input freqency from Flow sensor : 76.923077 Mhz

At slmulation time 2780

The input freqency from Temperature sensor : 76.923077 Mhz

The input freqency from Pressure sensor : 76.923077 Mhz

The input freqency from Flow sensor : 76.923077 Mhz

At slmulation time 2887

The input freqency from Temperature sensor : 76.923077 Mhz

The input freqency from Pressure sensor : 76.923077 Mhz

The input freqency from Flow sensor : 76.923077 Mhz

At slmulation time 2994

The input freqency from Temperature sensor : 76.923077 Mhz

The input freqency from Pressure sensor : 76.923077 Mhz

The input freqency from Flow sensor : 76.923077 Mhz

Compile time = 0.00000,

Load time = 0.00000,

Execution time = 1.05000

CHAPTER - VI

SIMULATION

6.1. INTRODUCTION

When the logic is designed in verilog HDL, it is necessary to know how to simulate and to run the project. The simulation software used for simulation is Test Bencher Pro. The functionality of the design block can be tested by using stimulus block. The stimulus block is also called a test bench. The stimulus block is also written in verilog.

6.2. COMPONENTS OF SIMULATION:

The two distinct components in a simulator are

- Stimulus block
- Design block

The connection between the components of simulation is shown in Fig.6.1. The stimulus instantiates the design block and directly drives the signals in the design block. The stimulus block is the top level module. The input to the design block will be the output of the stimulus block.

6.3. STEPS TO BUILD THE PROJECT

To run and simulate the project, it is necessary to build the project tree. The various steps involved are as follows:

- The project must be added to the project window.
- The build command is selected. This compiles the verilog file and builds the verilog tree.

This shows the hierarchical view of the modules and internal details. One module is surrounded by brackets. This indicates the top-level module. This is the highest-level instantiated component. This is also known as test bed.

In our project, the stimulus module is the test bed. All the sub modules can be viewed by descending the top-level module tree. The various sub modules in our project are

- <<< stimulus >>>
- Total
- Process
- Temp_control
- Flow_control

- Pres_control
- Counter_t
- Counter_p
- Counter_f

In order to display the signals, ports and components contained in each module, the tree is expanded. When the tree is expanded the various input, output ports, their vector declarations, internal signals are displayed.

6.4. SIMULATION OF THE PROJECT

Once the project is built, all the names of the internal signals in the top-level module are automatically added to the simulation diagram window. In our project, the various internal signals are:

- T_com_out, P_com_out, f_com_out
  - Counter output which are 8 bit wide
- T_com_out, P_com_out, f_com_out
  - Counter output which are 8 bit wide
- T_sen_in, P_sen_in, f_sen_in
  - Sensor inputs

- T_com_out, P_com_out, f_com_out
  - Comparator output which are 2 bit wide
- T_error, P_error, F_error
  - Error values which are vectors of 8 bit wide.
- T_fin, P_fin, F_fin
- T_start, P_start, f_start

When the project is run by pressing the run button, the simulation is started and it runs until the end of the simulation time. In our project, the simulation gets terminated at the end of 3000 time units. The simulation diagram for the project is shown in Fig. 5.2

6.5 SYNTHESIZING:

When we synthesize the project, the important blocks such as

- Pressure Control Block
- Temperature Control Block
- Flow Rate Control Block
- Process Block

are generated along with the I/O ports. The inter connection between these blocks are shown in Internal Block Diagram. The final report explains the total accumulated area, the number of input buffers, output buffers, ports, nets and instances of the blocks namely

- Total Block
- Counter_t Block
- Counter_p Block
- Counter_t Block

The report is shown at the end of this chapter. This report also contains the arrival time, data required time, slack time for all the blocks.

The schematic diagram shows the gate level diagram indicating various input's, input buffers, counter blocks and output ports. The register transfer level schematic diagram shows the input ports and output ports of the these main blocks, these diagram are shown at the end of this chapter.

- Press control block
- Temperature control block
- Flow control.

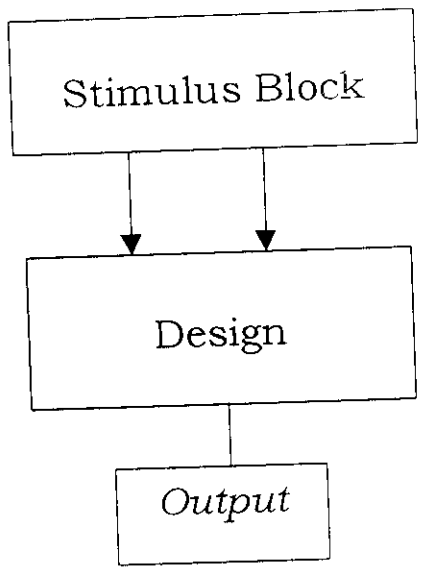
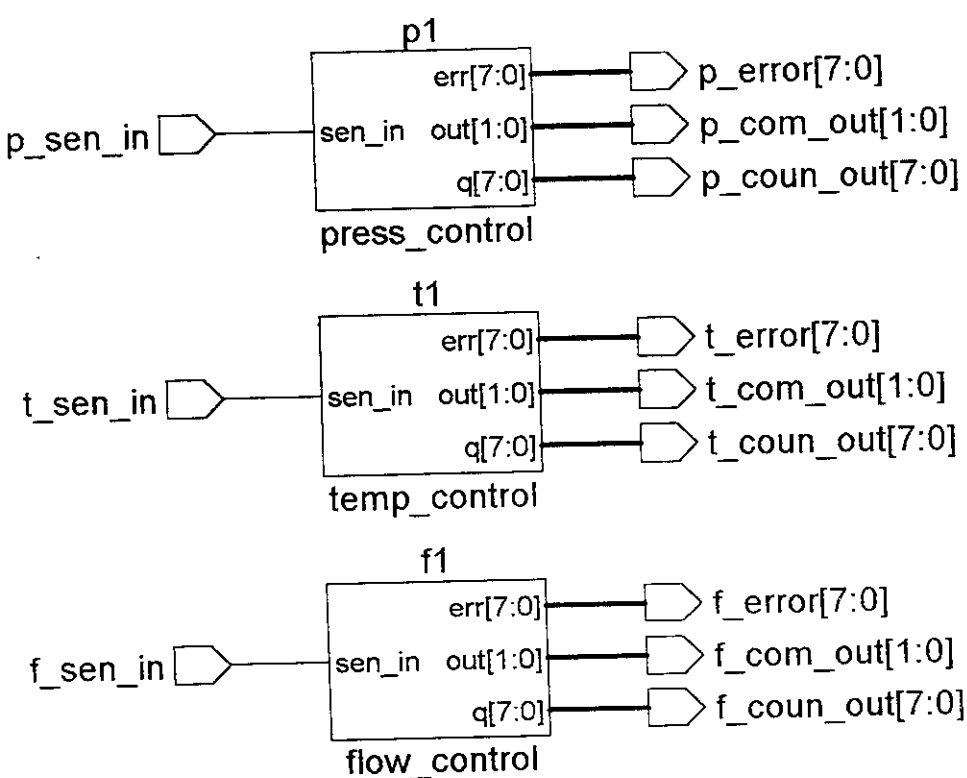
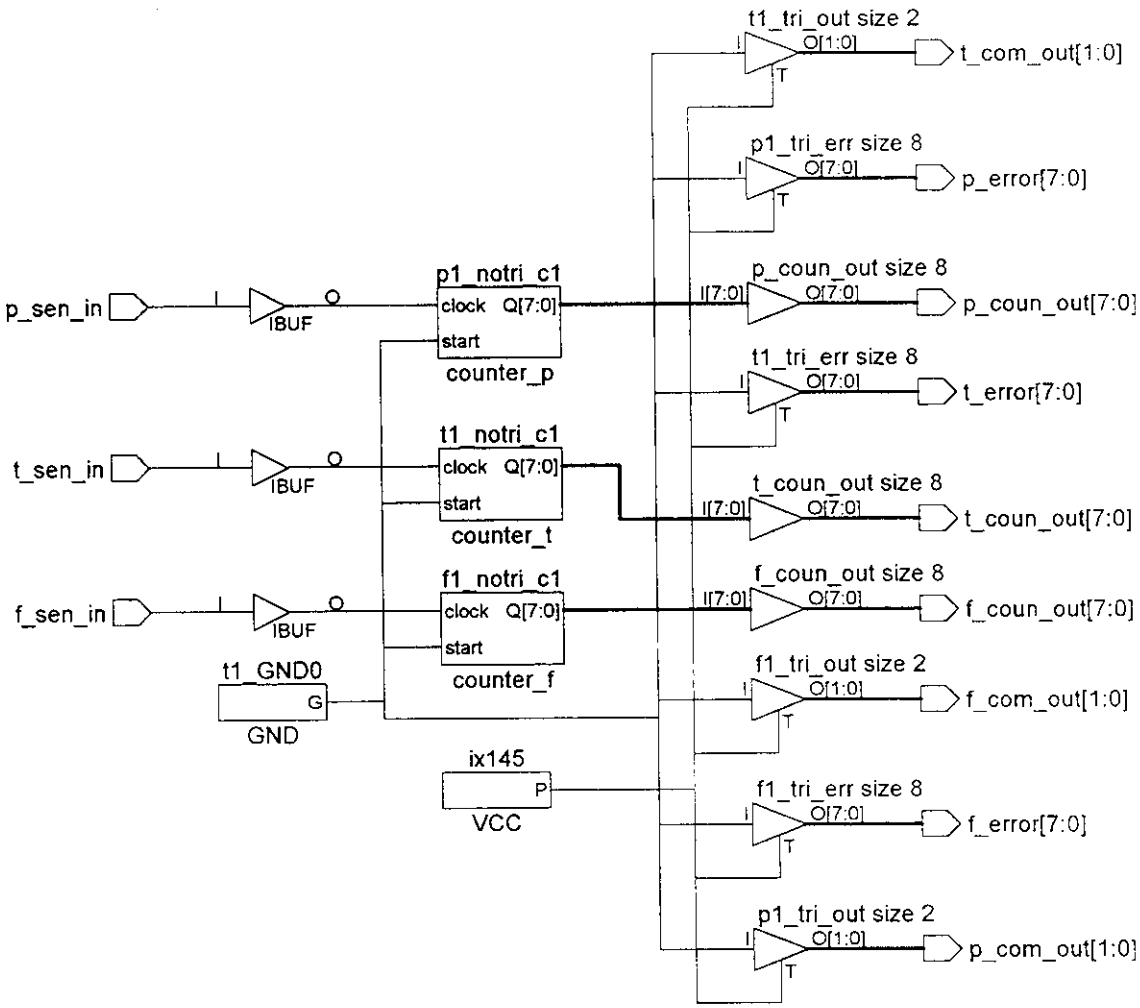


Fig.6.1. Components of Simulation

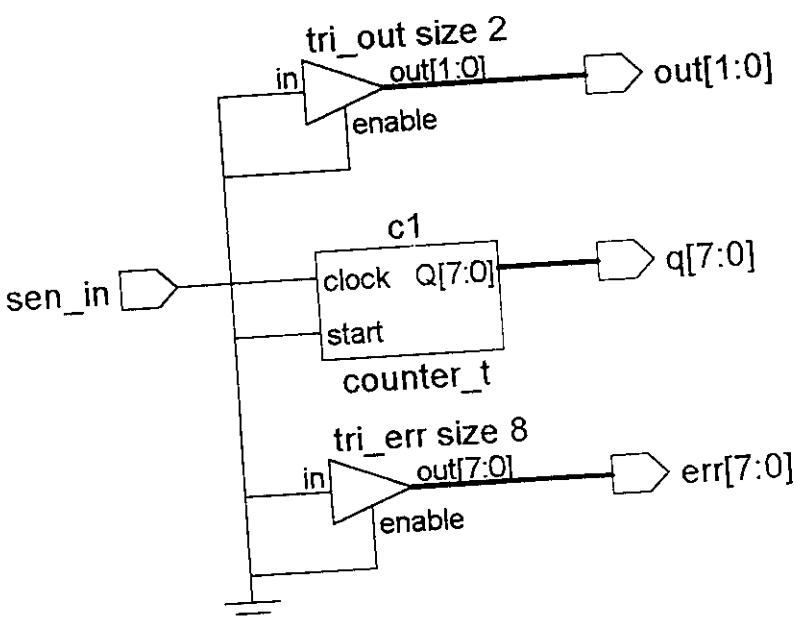
RTL SCHEMATIC



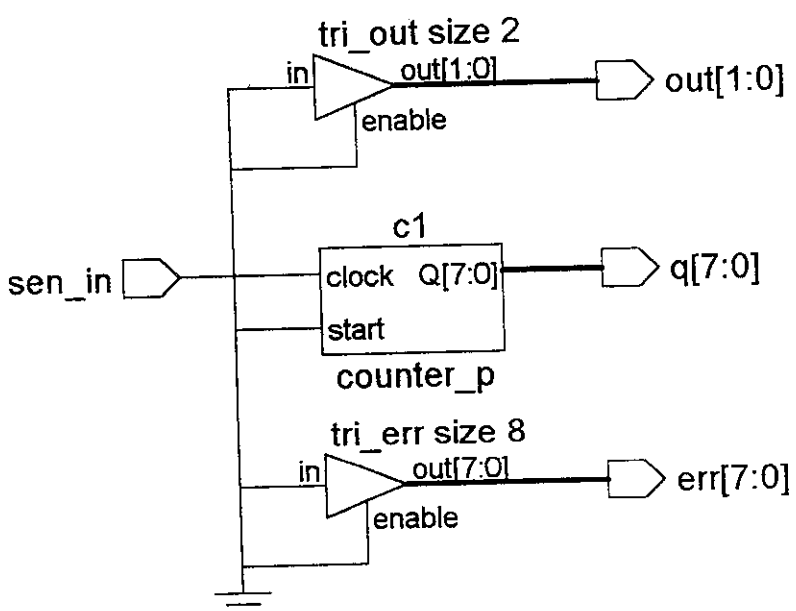
TECHNNOLOGY SCHEMATIC



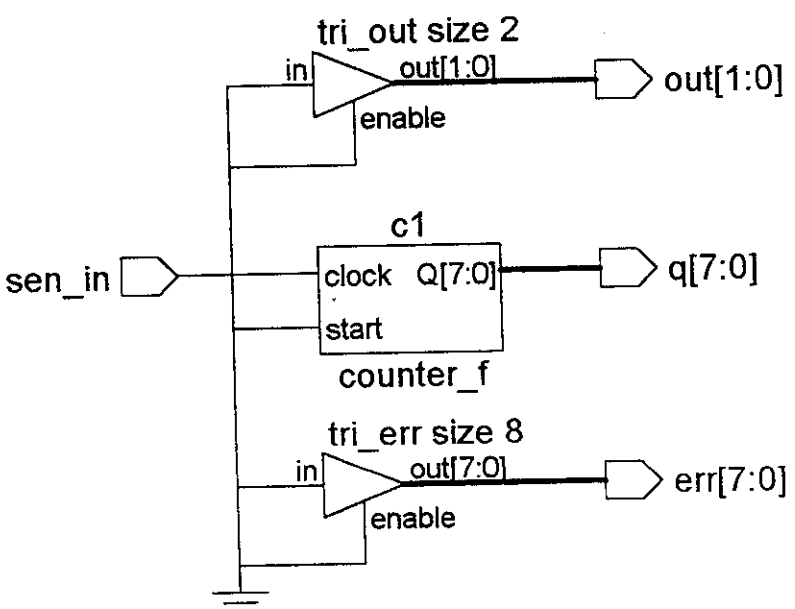
TEMPERATURE CONTROL MODULE



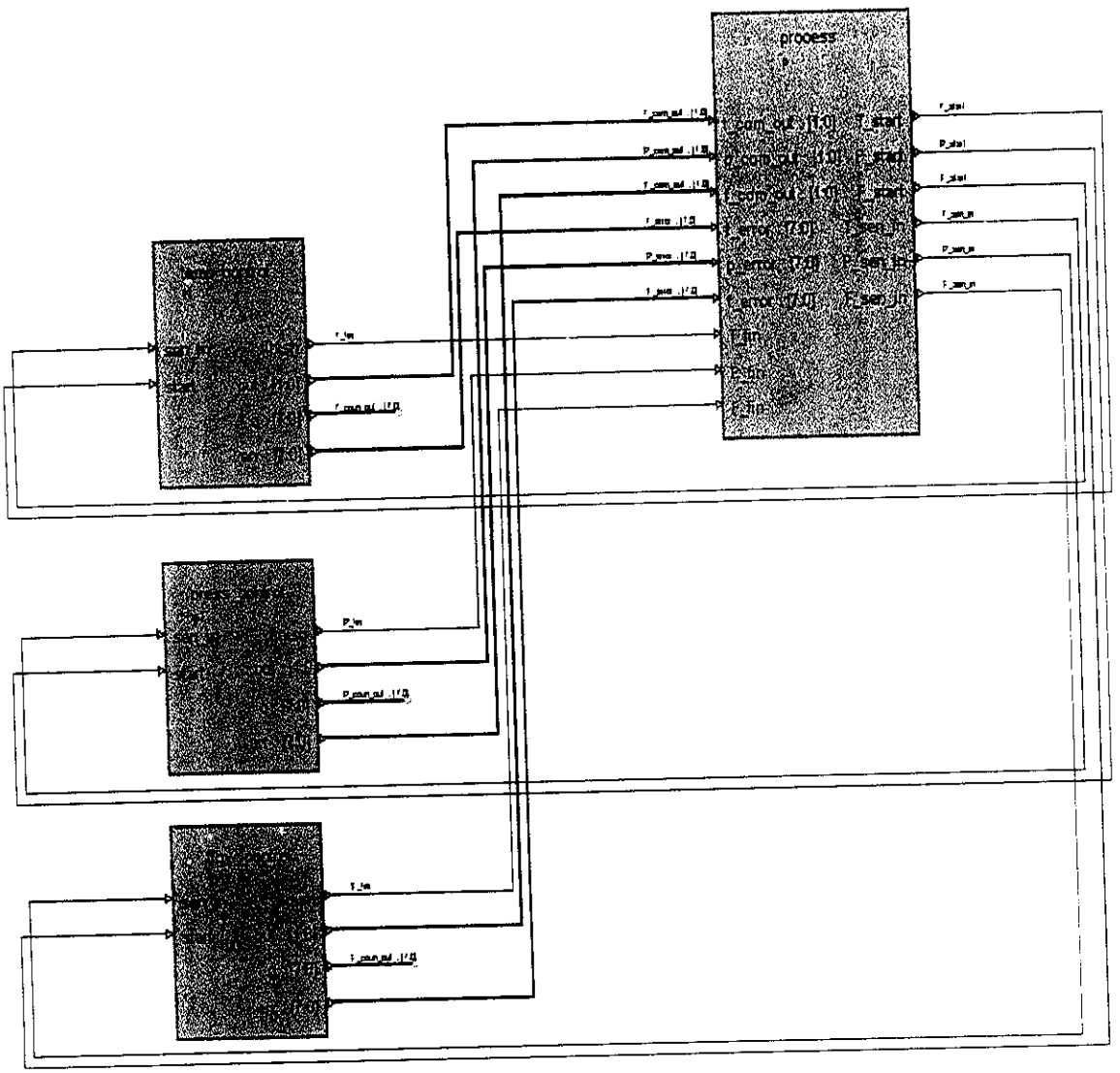
PRESSURE CONTROL MODULE



FLOW RATE CONTROL MODULE



INTERNAL BLOCKS



Cell: *Total* View: INTERFACE

Total accumulated area :

Number of IBUF	:	3
Number of OBUF	:	24
Number of OBUFT	:	30
Number of ports	:	57
Number of nets	:	86
Number of instances	:	62
Number of references to this view	:	0

Cell	Library		References	Total Area
GND	xcv	1x	1	1 GND
IBUF	xcv	3x	1	3 IBUF
OBUF	xcv	24x	1	24 OBUF
OBUFT	xcv	30x	1	30 OBUFT
VCC	xcv	1x	1	1 VCC
counter_f	work	1x	1	1 counter_f
counter_p	work	1x	1	1 counter_p
counter_t	work	1x	1	1 counter_t

Cell: counter_t View: INTERFACE

Total accumulated area :

Number of ports	:	10
Number of nets	:	0
Number of instances	:	0
Number of references to this view	:	2

Cell: counter_p View: INTERFACE

Total accumulated area :

Number of ports	:	10
Number of nets	:	0
Number of instances	:	0
Number of references to this view	:	2

Cell: counter_f View: INTERFACE

Total accumulated area :

Number of ports	:	10
Number of nets	:	0
Number of instances	:	0
Number of references to this view	:	2

Slack Table at End Points:

End Points	Slack	Arrival		Required	
		Rise	Fall	Rise	Fall
t_com_out(1)/	14.00	6.00	6.00	20.00	20.00
t_com_out(0)/	14.00	6.00	6.00	20.00	20.00
p_com_out(1)/	14.00	6.00	6.00	20.00	20.00
p_com_out(0)/	14.00	6.00	6.00	20.00	20.00
f_com_out(1)/	14.00	6.00	6.00	20.00	20.00
f_com_out(0)/	14.00	6.00	6.00	20.00	20.00
t_error(7)/	14.00	6.00	6.00	20.00	20.00
t_error(4)/	14.00	6.00	6.00	20.00	20.00
f_error(1)/	14.00	6.00	6.00	20.00	20.00
f_error(2)/	14.00	6.00	6.00	20.00	20.00

Critical Path Report

Critical path #1, (path slack = 17.8):

Name	Gate	Arrival	Load
f_sen_in/		0.00 0.00 up	2.19
f_sen_in_ibuf/O	IBUF	2.16 2.16 up	2.19
f1_notri_c1/clock	Generic_Black_Box	0.00 2.16 up	0.00

Data arrival time	2.16
Data required time (default specified - setup time)	20.00

Data required time	20.00
Data arrival time	2.16
Slack	17.83

Critical path #2, (path slack = 17.8):

Name	Gate	Arrival	Load
p_sen_in/		0.00 0.00 up	2.19
p_sen_in_ibuf/O	IBUF	2.16 2.16 up	2.19
p1_notri_c1/clock	Generic_Black_Box	0.00 2.16 up	0.00

Data arrival time	2.16
Data required time (default specified - setup time)	20.00

Data required time	20.00
Data arrival time	2.16
Slack	17.83

Critical path #3, (path slack = 17.8):

Name	Gate	Arrival	Load
t_sen_in/		0.00 0.00 up	2.19
t_sen_in_ibuf/O	IBUF	2.16 2.16 up	2.19
t1_notri_c1/clock	Generic_Black_Box	0.00 2.16 up	0.00

Data arrival time	2.16
Data required time (default specified - setup time)	20.00

Data required time	20.00
Data arrival time	2.16
Slack	17.83

CHAPTER – VII

CONCLUSION

The project entitled **"DESIGN AND SIMULATION OF CONTROL OF AUTOMATION USING VERILOG HDL"** has been successfully designed, developed and simulated.

When compared to electronic circuits and techniques like microprocessor , microcontroller, Programming Logic Controller, Computers, this design in VLSI technique using verilog HDL as the hardware language is easy, simple and single chip control.

In other techniques, due to the presence of bulky interfacing circuits, buses, etc, the inter chip time delay is more. As this design is a single chip control, the time delay is very much reduced (less than 1 nanoseconds). The need for memory ,external clock generators are eliminated in this design.

Also this technique is easy to understand, consumes very less power and does not require any maintenance. The programs written in verilog HDL can be easily debugged and implemented when compared to Printed Circuit Boards.

REFERENCES

1. Neil .H.E. Weste and Kamran Eshraghian – "Principles of CMOS VLSI design" A System Perspective. Addision – Wesley Publishing Company, California.

2. Samir Palnitkar – "Verilog HDL: A Guide To Digital Design And Synthesis" Sunsoft Press, USA, 1996.

3. Rajvir Singh, Rajeev Madhavan, E.Sternheim, Yatin Trivedi – "Digital Design And Synthesis With Verilog HDL." Automata Publishing Company, California, 1994.

4. Donald Thomas, Phil Moorby –"The Verilog Hardware Description Language" Kluwer Academic Publishers, MA 1994.

5. Stuart Sutherland –"Verilog HDL Language Reference Guide." Sutherland Consulting, California, USA, 1993.

6. Websites:
   www.syncad.com
   www.verilog.com
   www.mentor.com