P-1430

FPGA SIMULATION AND MICRO CONTROLLER IMPLEMENTATION OF POWER FACTOR CONTROLLER FOR AC TO DC CONVERTER

A PROJECT REPORT

Submitted by

MANIKANDAN.D	(71201105024)
PRAVEEN R.KASHYAP	(71201105034)
THIRUMALAI RAJAN.K	(71201105065)
VINU BALAJEE.C.R	(71201105072)

in partial fulfillment for the award of the degree

of

BACHELOR OF ENGINEERING

in

ELECTRICAL & ELECTRONICS ENGINEERING

Under the guidance of

Mr.S.Chidambaram, M.E.

KUMARAGURU COLLEGE OF TECHNOLOGY,
COIMBATORE - 641006

ANNA UNIVERSITY :: CHENNAI - 600 025

APRIL – 2005

ANNA UNIVERSITY : CHENNAI-600 025

BONAFIDE CERTIFICATE

Certified that this project report titled **"FPGA SIMULATION AND MICRO CONTROLLER IMPLEMENTATION OF POWER FACTOR CONTROLLER FOR AC TO DC CONVERTER"** is the bonafide work of

MANIKANDAN.D	**-**	**Register No. 71201105024**
PRAVEEN R.KASHYAP	**-**	**Register No. 71201105034**
THIRUMALAI RAJAN.K	**-**	**Register No. 71201105065**
VINU BALAJEE.C.R	**-**	**Register No. 71201105072**

who carried out the project work under my supervision.

[illegible]

Signature of the Head of the Department

Prof.K.Regupathy Subramanian,B.E.,M.Sc.
DEAN/EEE,
Kumaraguru College of
Technology

[illegible] 11/04/05

Signature of the guide

Mr.S.Chidambaram,M.E
Lecturer, EEE Dept.,
Kumaraguru College of
Technology

CERTIFICATE OF EVALUATION

College : **KUMARAGURU COLLEGE OF TECHNOLOGY**

Branch : **Electrical & Electronics Engineering**

Semester : **Eighth Semester**

Sl.No.	Name of the Students	Title of the Project	Name of the Supervisor with Designation
01	Manikandan.D	**"FPGA simulation and Micro Controller implementation of Power factor controller for AC to DC converter"**	Mr.S.Chidambaram,M.E Lecturer, EEE Dept.
02	Praveen R.Kashyap		
03	Thirumalai Rajan.K		
04	Vinu Balajee.C.R		

The report of the project work submitted by the above students in partial fulfillment for the award of Bachelor of Engineering degree in Electrical & Electronics Engineering of Anna University was evaluated and confirmed to be report of the work done by the above students.

[illegible]

(INTERNAL EXAMINER)

[illegible] 15/4/05

(EXTERNAL EXAMINER)

ACKNOWLEDGEMENT

The completion of our project can be attributed to the combined efforts made by us and the contribution made in one form or the other, by the individuals we hereby acknowledge.

We are highly privileged to thank **Dr. K. K. Padmanabhan**, Principal, Kumaraguru College of Technology for allowing us to do this project.

We express our heartfelt gratitude and thanks to the Dean (R&D) of Electrical and Electronics Department, **Prof. K. Regupathy Subramaniam**, for encouraging us to choose and for being with us right from the beginning of the project and guiding us at every step.

We express our sincere thanks to **Dr. T. M. Kameswaran,** former head of our department, for his kind support during the course of study.

We wish to place on record our deep sense of gratitude and profound thanks to our guide **Mr. S. Chidambaram,** Lecturer, Electrical and Electronics department, for his valuable guidance, constant encouragement, continuous support and co-operation rendered throughout the project.

We are also thankful to all teaching and non-teaching staffs of Electrical and Electronics Engineering Department for their kind help and encouragement.

Last but not least, we extend our sincere thanks to all our parents and friends who have contributed their ideas and encouraged us for completing the project.

ABSTRACT

Power factor plays an important role in almost in all power systems. It is essential to improve the power factor almost to unity of each and every system which leads to increase in efficiency and decrease the size of the components used. This project titled **"FPGA Simulation and Micro-controller Implementation of Power factor Controller for AC to DC Converter"** aims at improving the input power factor of commercial converters closer to unity.

This project uses the PWM control method of active current shaping for power factor improvement that yields input power factors in range of 0.95 to 0.99.

The simulation is done with the Field Programmable Gate Array controller in VHSIC HDL (Very High Speed Integrated Circuit Hardware Description Language) using Xilinx 1.5 tool. The hardware is implemented using (Programmable Interface Controller) PIC. The same algorithm used in the FPGA simulation was used in the PIC micro controller and the power factor improvement was analyzed.

LIST OF SYMBOLS

SYMBOL	DESCRIPTION
V_s	Source voltage
V_o	Output voltage
I_S	Source current
I_o	Output current
I_{in}	Input current
V_{in}	Input voltage
θ	Phase angle of voltage and current
Φ_n	Phase angle of the nth harmonic with respect to the input sinusoidal voltage
T_{vi}	Voltage to current transfer ratio
DF	Displacement factor
PF	Input power factor
HC	Harmonic content
THD	Total harmonic distortion
P_i	Input power
P_o	Output power
PFP	Power factor pre-regulator
G_{in}	Inverse of the equivalent input resistance
T_{vin}	Period of rectified V_{in}
R_{ii}	Current reflection ratio
T_{vv}	Voltage transfer ratio
$V_{out\text{-}nom}$	Nominal Output Voltage
I_{sh}	Short circuit current at the point of common coupling
I_{1P}	Real component of fundamental current
I_{1Q}	Reactive component of fundamental current
I_n	Harmonic current

LIST OF TABLES

LIST OF FIGURES

CONTENTS

INTRODUCTION

1. INTRODUCTION

AC to DC converters without control, are popularly known as rectifiers. The most commonly used AC source is 50 or 60 Hz voltage source, which is available from the electric utility supply, also called the line source. The uncontrolled rectifiers are designed using the diodes. The designs are inexpensive and popular in the industrial applications. In some of the rectifiers, the AC voltage from the electric utility is directly rectified without the use of an expensive and bulky transformer. In some applications, the DC voltage from the rectifier is connected to the DC bus for the distribution for several different circuit systems, sub-systems, and other converters as loads. In other applications, the rectifiers supply power to inductive-resistive (motors) and capacitive-resistive (power supplies) loads.

The rectifiers are required to supply ripple-free DC voltage or DC current to the load. In practice, the peak-to-peak output ripple is designed to be as small as possible and the ripple frequency as large as possible. The rectifiers usually draw highly non-sinusoidal current from the electric utility supply, giving rise to poor power factors and thus poor efficiency. Improving power factor is an important design objective. There are different techniques available for power factor improvement. Another design concern is the reduction of high frequency distortion in the line current, which is caused by switching loads or switch mode converters as loads.

The variation of input power factor as a function of (V_O/V_S) is shown in the figure below.

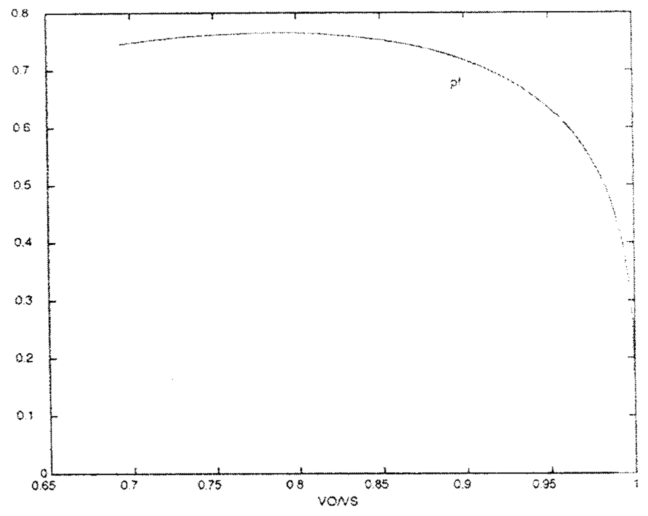


Fig: 1 Graph showing variation of input pf as a function of V_o/V_s

The plot shows that the input power factor remains constant when the output voltage is very low and decreases quadratically with the increase in output voltage when the input voltage remains constant. Also, it has to be noted that the total harmonic distortion increases with the increase in the output voltage, thus leading to very high injection of harmonics in the line, which disturbs the operation of other utilities, connected in parallel.

The main objective of this project is to suppress the harmonics and improve the input power factor of the commercial converter circuits by using the PWM control method of the Active current shaping technique. By using this method, the input current is made to trace the input voltage, which has a combined effect of reducing the harmonic distortion in the circuit and improving the input power factor of the circuit. By using this method, high input power factors almost nearing unity can be attained (0.95-0.99).

Thus this method has the following main advantages,

1. The rating of the system components reduces much greatly as a result of reduction in harmonics.
2. The reactive power consumed by the utility reduces, increasing the efficiency and reducing the cost of operation.

POWER FACTOR PROBLEMS IN CONVERTERS

2. POWER FACTOR PROBLEM IN CONVERTERS

Before getting started with analyzing the power factor problem in converters in depth, it is necessary to recollect some of the important concepts relating to the analysis of the converter circuits.

2.1. Performance Characteristics:

Converters performance consists of two parts: the transient and steady state part. The output voltage waveforms consist of high harmonic content called ripple. In steady state, all waveforms are stabilized and periodic, therefore easy to analyze.

Forward Transfer Characteristics:

The Forward Transfer characteristics describe the effect of input variables on the output variables. There are two performance parameters, which are used in the design of AC to DC converter systems.

Voltage transfer ratio:

$$T_{vv}=\frac{\bar{V}_o}{\bar{V}_s}$$

Voltage to Current transfer ratio:

$$T_{vi}=\frac{\bar{I}_o}{\bar{V}_s}$$

Reflective Characteristics:

The reflective characteristics give the effect of output conditions on the source variable.

Current Reflection Ratio:

$$R_{ii}=\frac{I_s}{I_o}$$

It is desirable to have a small change in the output variable for a large variation in the output characteristic.

Harmonic Profile:

The voltage and current waveforms at both the input and output ports are in general non-sinusoidal. The following measures are used for the undesirable Fourier components of a port variable.

Peak-to-Peak ripple in the output voltage and the output current are denoted as V_{or} and I_{or} respectively.

Ripple frequency is the frequency of the voltage or current ripple waveform. It is a simple multiple of the output source frequency. A higher multiple is a design goal. A large ripple frequency can be filtered with relative ease and with small value and small size filter components.

Output current waveform in an AC to DC converter is non-sinusoidal. The Fourier expansion of output current is given as follows:

$$i_O=I_O+\sum_{n=1,2...}^{\infty} I_{on}\sin(n.\theta-\phi_n)$$

where Φ_n is the peak amplitude of the nth harmonic component and the phase angle of the nth harmonic with respect to the output sinusoidal voltage.

The Fourier expansion of the input current is given as follows

$$i_S = \sum_{n=1,2\ldots}^{\infty} I_{Sn} \sin(n.\theta + \phi_n)$$

where Φ_n is the peak amplitude of the nth harmonic component and the phase angle of the nth harmonic with respect to the input sinusoidal voltage.

The phase angle of the fundamental component with respect to the output voltage waveform, Φ_1, the displacement factor DF, the cosine of Φ_1, and the input power factor PF are the three important performance measures.

$$DF = \cos(\phi_n)$$

$$PF = \left(\frac{\bar{I}_{SI}}{\bar{I}_S}\right)\cos(\phi_n)$$

A low value of power factor indicates that the converter is using substantial reactive power. A unity power factor is desired.

Lowest undesired Harmonic Frequency is an important design criterion.

Harmonic content of the source current is defined as the square root of the difference between the squares of total RMS source current and the RMS fundamental current.

$$HC = \sqrt{\bar{I}_S^2 - \bar{I}_{SI}^2}$$

Total Harmonic Distortion (THD) of the source current is defined as the ratio of harmonic content to the RMS fundamental current is calculated as following:

$$THD = \frac{HC}{\bar{I}_{SI}} = \sqrt{\left(\frac{\bar{I}_{SI}}{\bar{I}_S}\right)^2 - 1}$$

The voltage and current in the converter elements, especially, the switches may consist of high transients (other than the surges) which may be considerably higher than the normal ratings. These transients dictate the device selection and hence the cost.

Component Stresses are measured as the ratio of the peak voltage and current values to the respective RMS values on the elements.

2.2. Electric Utility Interface – Power factor control:

There is a growing concern regarding harmonic pollution of the power distribution system. Adoption of the International Electric Commission Standard IEC-555-3 and IEEE-519-1992 has helped greatly in the awareness for clean AC line currents and a power factor close to unity.

Electric Utility Distribution System:

The electric utility Distribution System can be represented by a Thevenin equivalent voltage source V_S in series with a source inductance L_S. The Line voltage at the point of common coupling is dependent on the load current and the source inductance,

$$\bar{V}_{1s} = \bar{V}_s - j\omega L_s \bar{I}_1$$

where subscript 1 refers to the fundamental harmonic (n=1). I_1 is the fundamental harmonic of the line current. Actually, the line current drawn from the electric utility by a power electronic system consists of several components: real and imaginary components at the line frequency and the high frequency harmonics:

$$\bar{I}_1 = \bar{I}_{1P} + j\bar{I}_{1Q} + \sum_{n>1} \bar{I}_n$$

The Imaginary (reactive) component I_{1Q} is the main contributor to the line voltage regulation. Minimizing I_{1Q} is referred to as the static VAR Control. The real component of the fundamental harmonic, I_{1p}, contributes little to the line voltage regulation. The phase angle of the fundamental harmonic current with respect to the line voltage is a very important parameter that determines the power factor.

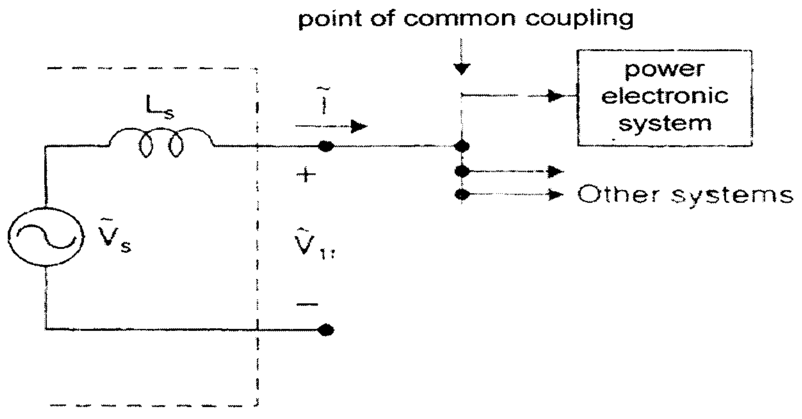


Fig: 2 Electrical utility distribution system

Recall the definition,

$$PF = \frac{\bar{I}_1}{\bar{I}_s}\cos(\theta_1)$$

The subscript s refers to the line source. The subscript 1 refers the fundamental component. The power factor is also the ratio of the real power used by the load to the power supply by the line source. A poor factor is translated in heavy expenses to the user; hence, considerable effort has been directed in improving the power factor.

Suppression of high harmonic in the electric utility is another very important requirement by Electric utility Vendors. The harmonics in a typical unfiltered single-phase line current are given below

Table 1: TYPICAL HARMONIC DISTORTION IN AN UNFILTERED SINGLE PHASE LINE CURRENT

N=	3	5	7	9	11
I_n/I_1 (%)	73.2	36.6	8.1	5.7	4.1

Compare these numbers with the percent amplitudes of high frequency harmonics as specified by IEEE519-1992 standards, as given in the below table.

TABLE 2: HARMONIC DISTORTION OF LINE CURRENT (I_N/I_1, FOR ALL HARMONICS LESS THAN 11TH)

As per IEEE-519-1992

For $I1/I_{sc}$	I_N/I_1	THD (%)
>0.05	4%	5%
0.02-0.05	7%	8%
0.01-0.02	10%	12%

Note: $I_{SC} = V_S/\Omega_e.I_{Sh}$ is the short circuit current at the point of common coupling.

Obviously, there is a great need for reduction of the harmonics. The interface design must aim at following two-fold goals:

a) To prevent all high frequency harmonics from being coupled to other systems connected to the point of common coupling, and
b) To shield the power electronics system under consideration from the transients and high frequency harmonics present in the electric utility.

The utility interface may be a passive filter or an active current shaping system.

2.3. Passive filtering:

The simplest technique is to use an inductor in series with the line source to shape the current waveform. This technique involves the smoothening of AC line current using LC filters in the line just before connecting it to a Power Electronic Converter system. One such filter is shown below;

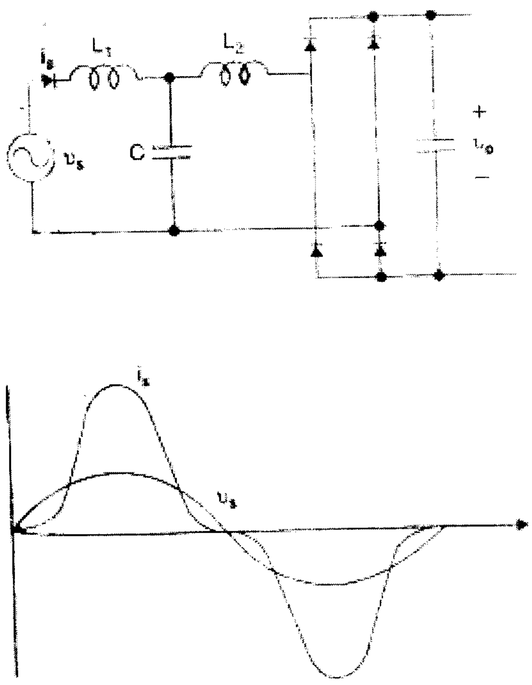


Fig: 3 Circuit using passive filtering method and Vs and I_s waveforms

The inductors basically add a source side inductance thereby reducing the high order harmonics in the input current; the output voltage ripples and also reduces the voltage and current stresses on the rectifier diodes. The capacitor suppresses the high frequency harmonics and transients from being coupled to and from the source. The power factor improves slightly; but the AC line voltage, available to the converter, becomes dependent on the load.

The disadvantage of the passive filtering is the size of the filter components at low line frequency (which are very large). The active current shaping is one technique for reducing the size of the filter components.

2.4. Active current shaping:

The AC line operated, switch mode power supplies and converters for motor drives all require an AC – DC Converter as interface to the AC line. However, the uncontrolled version, the bridge rectifier, is more popular. A large capacitor is connected to the output of this rectifier for reducing the voltage ripple. Since the Capacitor drawn line current only when the output voltage is below the line voltage, the line current consist of a highly non - sinusoidal waveform, with narrow and high peaks.

The power factor is ideal (equal to 1) when the bridge rectifier is loaded by a current sink and poorest when loaded by a voltage sink. In the case of current sink loading the rectifier diodes conduct for π line angle and the power stress on the diodes is uniformly distributed over a half line period. In the case of a voltage sink loading, the rectifier diodes conduct for a much shorter line angle and the power stress on the diodes is highly concentrated in a short duration. Similarly, the power factor is ideal when a voltage sink load is connected to a switched constant Current Wave form source and poorest when supplied by a switched constant DC voltage waveform source. A power factor correction device should act like an interface between the bridge rectifier and the voltage sink load. The interface appears as the current sink load to the bridge rectifier and a switched constant current waveform source to the voltage sink load.

There are two consequences that happen due to a poor power factor.

(1) The diodes in the bridge rectifier must be rated for higher currents that are several times more than the peak value of the fundamental components.

(2) The user pays for high reactive power from the AC line.

Improving the power factor in the AC – DC rectifier involves

(a) Shaping the line current to a sinusoidal wave form and

(b) Reducing the phase difference between the line voltage and the line current.

In essence, the AC – DC converter in the power factor correction emulates a resistor. The resistor emulator, also called the Power Factor Pre-regulator (PFP), is basically a DC – DC converter whose topology is given below;

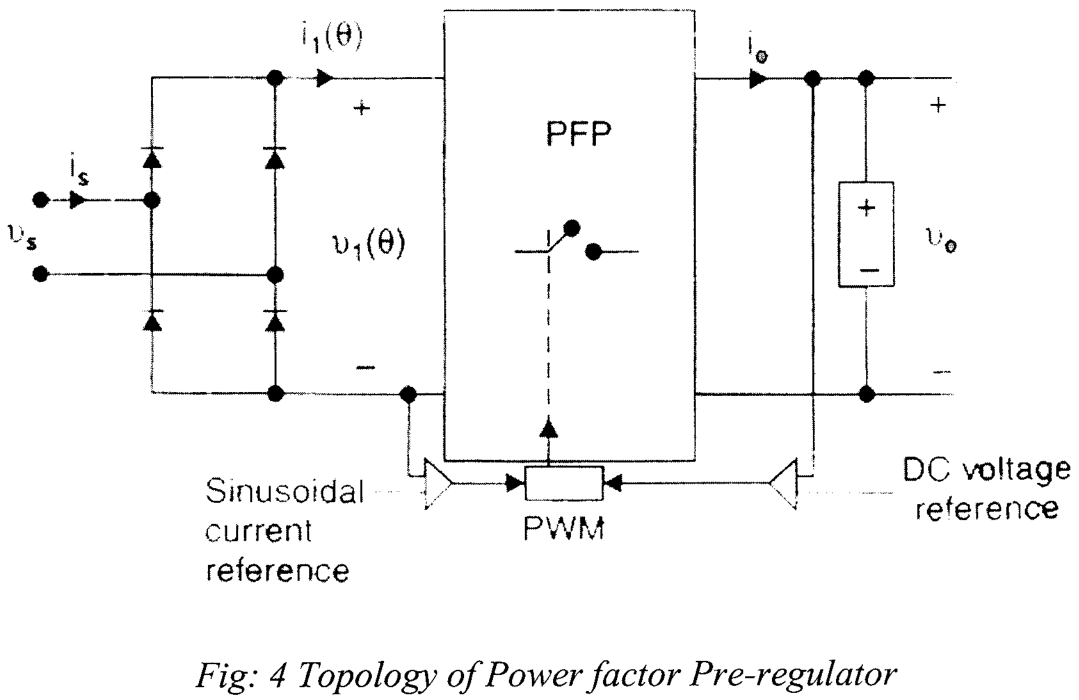


Fig: 4 Topology of Power factor Pre-regulator

The input to the PFP is a full – rectified sinusoidal voltage waveform. A voltage sink V_O represents the load in parallel with a large

C. The DC – DC converter is switched at a frequency f_S that is several times higher than the line frequency f_L. A large value of inductor would be needed to appear as a current sink load to the bridge rectifier. The current through the diode in PFP switches between 0 and the constant value I_O. The PFP acts as an ideal interface that appears as the current sink load to the bridge rectifier and a switched constant waveform source to the Voltage Sink load.

For the practical values of the inductor, the input current of the PFP, possess a triangular wave shape. The input current waveform before the diode bridge is modified to contain a strong fundamental sinusoid at the line frequency but with harmonics at several times higher frequency than the line frequency.

Since the switching frequency is very high in comparison to the line frequency, the input and output voltages of the PFP converter may be considered to be constant throughout the switching period. Thus, the PFP converter can be analysed like a regular DC – DC Converter. The line voltage and the input voltage to the PFP are given by

$$v_S = V_S \sin(\theta)$$

$$v_1 = V_S |\sin(\theta)| \qquad \theta = 2\pi f_L t$$

The voltage transfer ratio of PFP is required to vary within the angle θ in a half line period. The voltage transfer ratio of DC-DC PFP is

$$T_{vv}(\theta) = \frac{\langle V_O(\theta)\rangle}{\langle v_S(\theta)\rangle} = \frac{V_O}{V_S}|\sin(\theta)| \quad , \quad f_S \gg f_L$$

where V_O is the local average DC output voltage from PFP. The T_{VV} in a line period can be made high at the 0 and π angles by using boost, buck-boost or fly back topologies. Buck topology cannot provide high voltage transfer ratios.

2.5. For unity power Factor:

The current from the diode bridge must be identical in shape and in phase with the voltage waveform, hence,

$$i_1 = I_S |\sin(\theta)|$$

The input and output powers, averaged over a switching period, are given by:

$$p_i = v_1 i_1 = V_S I_S \sin^2(\theta)$$

$$p_O = V_O i_O$$

Assuming the conversion efficiency to be 1, ($P_i = P_O$), the output current requirement is determined as

$$i_O = \frac{(V_S I_S \sin^2(\theta))}{V_0}$$

The input and output powers, as averaged over the line period, are:

$$P_i = \frac{(V_S I_S)}{2}$$

$$P_O = V_O I_O$$

where I_O is the average DC output current from the PFP.
The output current of the PFP is then

$$i_O = 2 I_O \sin^2(\theta) = I_O(1 - \cos(2\theta))$$

The above discussion points out two important requirements for PFP

1. The voltage transfer ratio T_{VV} must be varied over a half line period.
2. The current output of the PFP should be varied according to the profile shown below

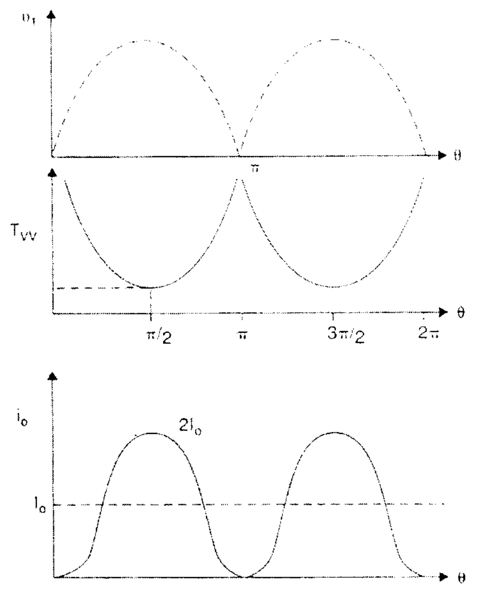


Fig: 5 Voltage transfer ratio & output current of PFP as required for unity power factor

Thus, two separate control loops are required. The control systems is given in below

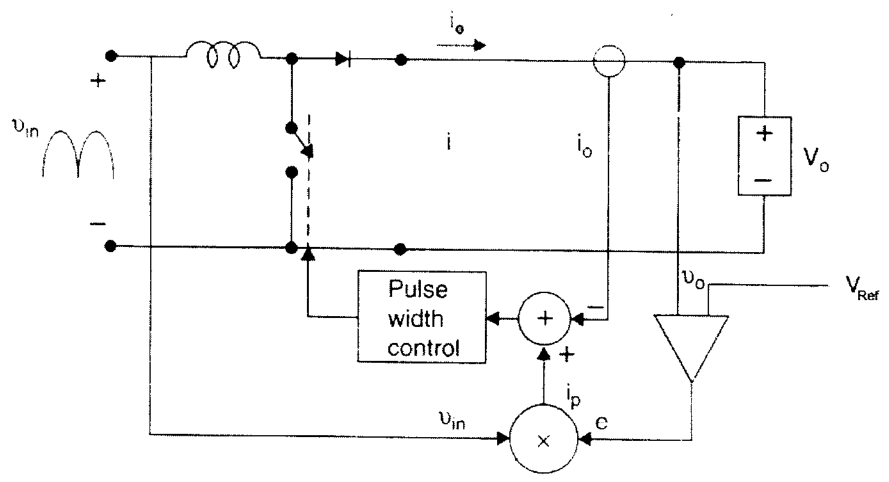


Fig: 6 Control loops in Power factor correction

In one of the loops, T_{VV} is controlled by the pulse width as the Control parameter. The output voltage is sensed, compared with the DC reference, and the error signal is used to modify the pulse width of the switching device in the PWM converter. In the other loop, the output current is sensed, compared with a sinusoidal reference, and the error is used to control the pulse width. These two loops are merged into one and called the current- mode control. In this method, the reference current waveform is made a function of the input voltage V_1 waveform and the error voltage,

$$i_{ref} = ev_1 = (V_O - V_{ref}) V_S |\sin(\theta)|$$

The error voltage is sampled only once in a switching period, T_s and held at the sampled value during the feedback to regulate the output voltage around the nominal value V_O. This method can yield power factors in the range of 0.95 to 0.99 and reduces the THD of the line current to within 3%.

2.6. PWM control of power factor:

Consider the AC- DC rectifier with a boost type PFP incorporated into it, as shown below

The voltage transfer ratio in the boost PFP is obtained by combining

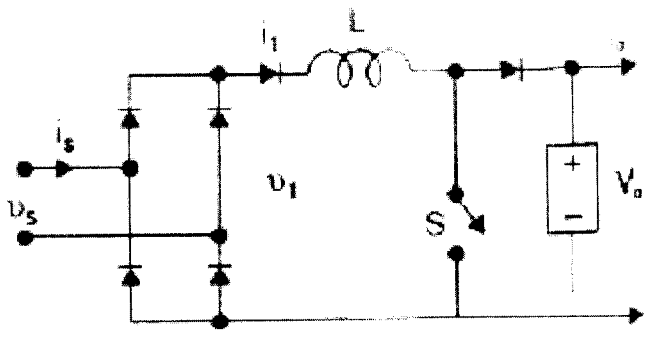
$$i_{ref} = ev_1 = (V_O - V_{ref}) V_S |\sin(\theta)|$$

$$V_s \tau + (V_s - \bar{V}_o)(T_s - \tau) = 0$$

The above two equation, we get,

$$T_{vv} = \frac{V_O}{(V_s |\sin(\theta)|)} = \frac{1}{(1-D)}$$

$$D = 1 - \left(\frac{V_S}{V_O}\right) |\sin(\theta)|$$



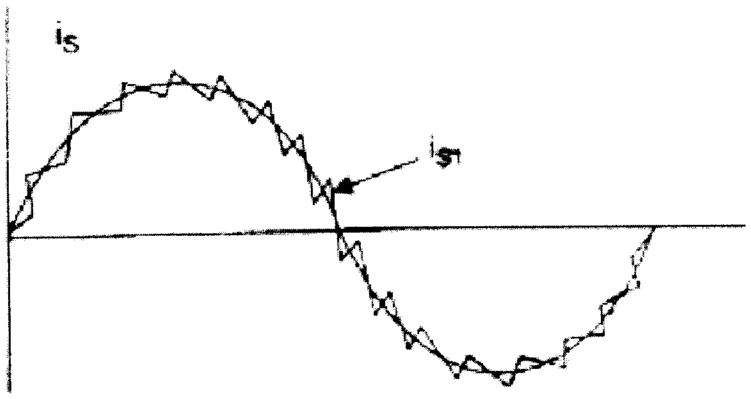


Fig: 7 Circuit diagram & waveform of Boost type Power factor Pre-regulator

The literature may be referred to for several other types of power factor correction strategies. An interesting strategy employs the constant current ripple method.

2.7. Constant Current Ripple Control of Power factor:

The input current to the PFP, i_l is maintained within a band of tolerance (+/-) ΔI with respect to the desired unity PF current waveform a given in figure above. The circuit is illustrated below

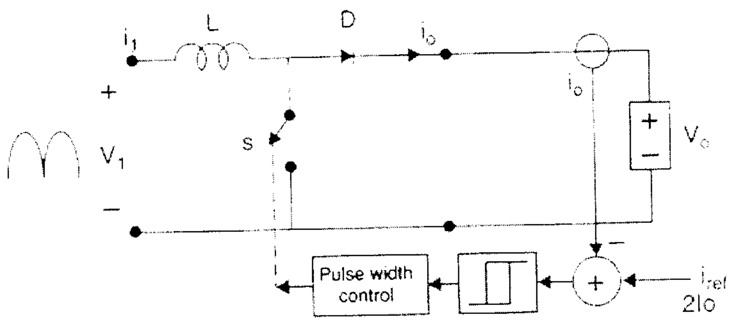


Fig: 8 Current ripple control & power factor correction

If the actual current increases more than the nominal value, plus the half of the tolerance band, the switch S1 is turned OFF and S2 is turned ON. Opposite switching occurs when the actual current decreases below the nominal value minus half of the tolerance band. The switching period is variable along the current waveform.

SIMULATION WITH FPGA

3. SIMULATION WITH FPGA

3.1. Introduction to FPGA:

The implementation of a digital system is not independent of the design style. Digital integrated circuits may be realized in different technologies depending on their size and their role in the system.

Integrated circuits are connected in such a way as to realize the design. Several masks define the locations and connectivity of the transistors. A mask corresponds to one of the silicon compound layers that form the transistors and the interconnect layers.

Circuit implementation may be grouped into two main categories, fully custom and semi custom design. The latter category itself consists of several approaches. These approaches have facilitated the design and manufacturing of Application Specific Integrated Circuits (ASICs). ASICs can be defined as integrated circuits designed for a particular application in low volumes. They can also be contrasted to standard integrated circuits such as microprocessors and memories that are used in a wide range of application and are available 'off the shelf'.

3.2. Custom Integrated Circuit:

Custom integrated circuit, are created using unique masks for all layers during the manufacturing process. The user controls chip density with high utilization. Since the design controls all stages of the chip layout, maximum design flexibility and high performance are possible. Consequently, only highly skilled and competent designers are engaged with such design flexibility and high performance are possible. Also, development time is long, and development costs are extremely high. For applications that require high volume, custom provides a low cost alternative. The high cost of design and the sting can be successfully amortized over the high volume

3.3. Mask Programmable Gate Array (MPGA)

The gate array implementation approach uses generic masks for all but the metallization layers, which are customized to the user's specifications. The generic masks create an array of modular functional block. Modules of transistors are arranged in rows that are separated by fixed-width channels. User logics implemented, by patterning these transistors into logic functions and connecting the different modules.

A *cell library*, making the designers expertise less critical than in the case of the full custom methodology, usually facilitates the design. For the same reasons, MPGAs offer shorter development time and lower development costs than do custom integrated circuits. A special class of gate array is channel-less. They are known as *sea of gates*.

3.4. Standard Cells

In this approach, as in the case of MPGAs, the design task is facilitated by the use of pre-designed modules, which are the masks for the modules. The modules standard cells, which are the same pitch size, are usually saved in a database. Designers select cells from the database to realize their design. The cells are then placed in rows and interconnected. The routing is done within channels that may be of variable width. Placement and routing are done automatically almost removing the designers from the physical design process. Compared with custom Integrated circuits, implemented in standard cells are less efficient in size and performance; however, their development cost is lower. Mixed standard cells and macros have also proliferated from the standard cell design.

3.5. Field Programmable Devices

Like gate arrays field programmable devices are pre-fabricated. However, the logic is implemented by electrically programming interconnects and personalizing the basic cells, typically in the user's laboratory instead of a factory. The field-programmable devices have a variety of architectures. Implementing the design in programmable logic devices has the advantage of *fast turn around*, but limits the design flexibility. Development time and costs are significantly *lower* that are those of any other IC implementation. According to their architecture, we distinguish two main categories of user programmable logic devices: Programmable logic Devices (PLDs) and Field Programmable Gate Arrays.

- *PLDs* consist of programmable AND arrays (product terms) and fixed fan-in programmable, OR gates that are followed by flip-flops. The outputs of the flip-flops can be fed back as input lines in the product terms. The product line can be connected to any combination of inputs. The connecting device may be a fuse as in the case of bipolar chips or transistor. The transistor can be chosen to act as an open connection or to function normally as a switch. PLDs are at the low-density end of field programmable logic devices. Their densities range from 1,000 up to 10,000 gates. Utilization varies with applications, but it is typically very low because of the rigid AND/OR architecture. Initially, PLDs used to be fabricated with bipolar technology; however, complimentary metal-oxide semiconductors (CMOSs) are now more popular.

- ***Field programmable gate arrays*** combine the architecture of gate arrays with the programmability of programmable logic devices (PLDs). An FPGA normally consists of several uncommitted logic blocks in which the design is to be encoded. The logic block consists of some universal gates. That is, gates that can be programmed to represent any function like multiplexers (MUX), random-access memories, NAND gates, transistors, etc. The connectivity between blocks is programmed via different types of devices, SRAM (static random access memory) or antifuse. The architecture of the chip depends on the fashion in which the blocks are arranged.

3.6 Architecture of XC4000E Family

3.6.1 Functional Description.

XC4000 series devices achieve *high speed* through advance semiconductor technology and improved architecture. The *XC4000E* support system clock rates of up-to 80MHz and internal performance in excess of 150MHz. Compared to a older *Xilinx* FPGA families, *XC4000 series* devices are more powerful. They offer on-chip edge triggered and dual-port RAM, clock enables on I/O flip-flops and wide-input decoders. They are more versatile in many applications, especially those involving RAM. *Design cycles* are faster due to a combination of increased routing resources and more sophisticated software tool.

3.6.2 Basic Building Blocks

Xilinx user-programmable gate arrays include two major configurable elements, Configurable logic blocks (CLBs) and Input-Output Blocks (IOBs).

- *CLBs* provide the functional elements for constructing the user's logic.
- *IOBs* provide the interface between the package pins and internal signal lines.

The functionality of each circuit blocks is customized during configuration programming internal static memory cells (SMCs) .The value stored in the memory cells determine the logic functions and interconnections implemented in the FPGA. Each of these available circuits is described in this section.

3.6.3 Configurable Logic Blocks (CLBs)

Configurable logic blocks implement most of the logic in an FPGA. Two 4 input function generators (F & G) offer unrestricted versatility. Most combinatorial logic functions need four of fewer inputs. However, a third function generator (H) is provided. The H function generator has three inputs. Zero, one or two of these inputs can be the outputs of F and G; the other input(s) are from outside the CLB. The CLB can therefore, implement certain functions of up to 9 variables. Each CLB contains two storage elements that can be used to store the function generator outputs.

Function generator outputs can also drive two outputs independent of the storage element outputs. This versatility increases logic capacity and simplifies routing. Thirteen CLB inputs and four CLB outputs provide access to the function generators and storage elements. These inputs and outputs connect to the programmable interconnect resources outside the block.

Function Generators

The function generators are implemented as memory lookup tables. The propagation delay is therefore independent of the function implemented. A third function generator, H, can implement any Boolean function of its three inputs. Two of these inputs can optionally be the F' and G' functional generator outputs.

Implementing wide functions in a single block reduces both the number of blocks required in the signal path, achieving both increased capacity and speed. The versatility of the CLB function generators significantly improves system speed. In addition the design software tools can deal with each function generator independently. This flexibility improves the cell usage.

- When 3 separate functions are generated, one of the function outputs must be captured in a flip-flop internal to the CLB, only two registered function generators outputs are available from the CLB.

- User configurable, input /output blocks, (IOBs) provide the interface between external package pins and the internal logic. Each IOB controls one package pin and can be configured for input, output or bi-directional signals .The Figure below shows a simplified block diagram of the XC4000E IOB.

IOB Input Signals

Two paths labeled I1 and I2 in the Figure bring input signal into the array. Inputs also connect to an input register that can be programmed as either an edge-triggered flip-flip or a level-sensitive latch. Placing the appropriate library symbol makes the choice variations with inverted clocks are available and some combinations of latches and flip-flops can be implemented in a single IOB. The XC4000E inputs can be globally configured for either TTL (1.2V) or (5.0Volt) CMOS thresholds.

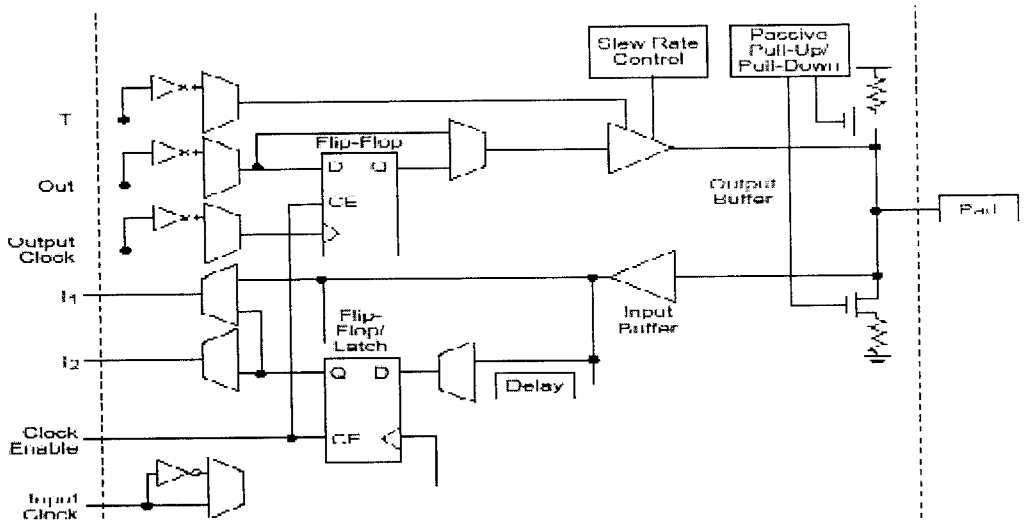


Fig: 9 Simplified block of XC4000E input and output unit

IOB output Signals

Output signals can be optionally inverted with in the IOB, and can pass to the pad directly or be stored in an edge-triggered flip-flop. The polarity of these signals is independently configured for each IOB. The high driver is an n-channel pull-up transistor, which has been pulled to a voltage of one transistor threshold below V_{CC}.

Pull-up and Pull down Resistors

Programmable pull-up and pull-down resistors are useful for tying unused pins to V_{CC} or GND, which minimize power consumption and reduce noise sensitivity. The configurable pull-up resistor is a p-channel transistor that pulls to V_{CC}. The configurable pull-down resistor is an n channel transistor that pulls to Ground.

Independent Clocks

Separate clock signals are provided for the input and output flip-flops. The clock can be independently inverted for each flip-flop with the IOB, generating either falling edge or rising-edge triggered flip-flops .The clock input for each IOB are independent.

On-Chip Oscillator

XC4000 Series devices include an internal oscillator. This oscillator is used to clock the power on time out, for configuration memory clearing, and as source of Clock in Master configuration modes. The oscillator runs at a nominal 8 MHz frequency that varies with process, V_{cc} and temperature. The output frequency falls between 4 and 10 MHz.

Programmable Interconnect

All internal connections are composed of metal segments with programmable switching points and switching matrices to implement the desired routing. There are several types of Interconnections. They are

- ***CLB routing*** is associated with each row and column of the CLB array.
- ***IOB routing*** forms a turn (called a *Versa Ring*) around the outside of the CLB array. It connects the I/O with the internal logic blocks.
- *Global routing* consists of dedicated networks that primarily designed to distribute clocks throughout the device with minimum

delay and skew Global routing can also be used for other high fan-out signals.

CLB Routing Connections

A high level diagram of the routing resources associated with one CLB is shown in the Figure below. The shaded arrows represent routing present only in XC4000X devices. CLB inputs and outputs are distributed on all four sides, providing maximum routing flexibility. In general, the entire architecture is symmetrical and regular. It is well suited to established placement and routing algorithms.

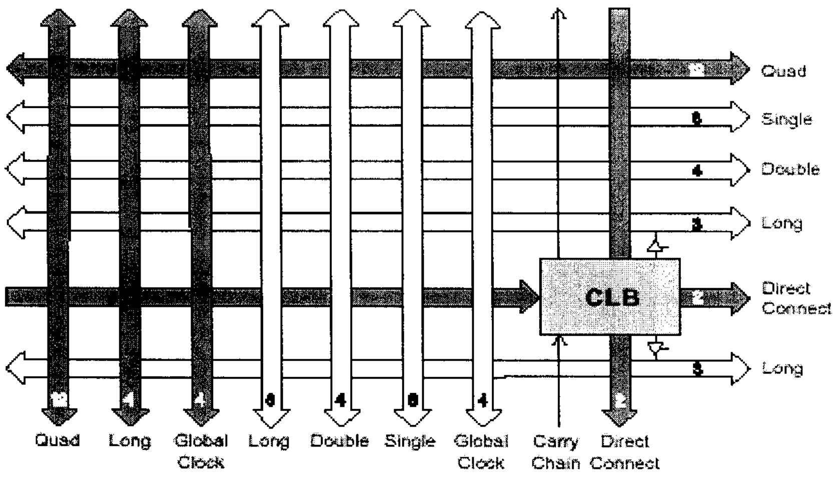


Fig: 10 High-level routing diagram of XC4000 Family

3.6.4 Programmable Switch Matrices (PSM)

The PSM is given in *Figure (a)* and *Figure (b)*. The figure (b) shows the active role of PSM in connecting the CLBs and wires. The horizontal and vertical single and double length lines intersect at a box called programmable pass transistors used to establish connections between the lines.

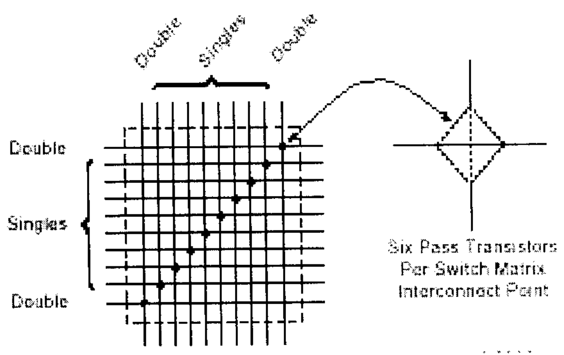
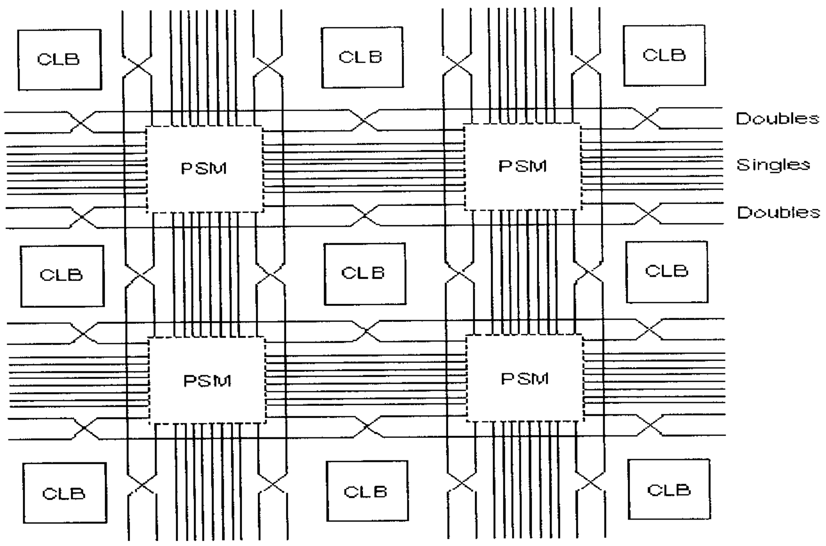


Fig: 11 Programmable Switch Matrix (PSM)



Single and Double length lines with Programmable Switch Matrices (PSM)

For example a single length signal entering on the right side of the switch matrix can be routed to a single length line on the top, left or bottom sides. If multiple branches are required, a double length signals can be routed to a double length line on any or all of the other three edges of the programmable switch matrix. Single length lines provide the greatest interconnect flexibility and other fast routing between adjacent blocks

The double length lines consist of grid of metal segments each twice as long as the single length lines: they run past two CLBs before entering a switch matrix. Long lines form a grid of metal interconnecting segments that run the entire length or width of the array.

Flip-Flops

The two edge-triggered D-type flip-flops have common clock (K) and clock enable (EC) inputs. Either or both clock inputs can also be permanently enabled. It passes the combinational output(s) to the interconnect network and store the combinatorial results.

Clock Input

Each flip-flop can be triggered on either the rising or falling clock edge. The clock pins are shared by both storage elements. However, the clock is individually invertible for each storage elements. Any inverter placed on the clock input is automatically absorbed into the CLB.

Clock Enable

The Clock enable signal (EC) is active high. The EC pin is shared by both storage elements. If left unconnected for either. The clock enable for that storage element defaults to the active state. EC is not invertible within the CLB.

Set/Reset

An asynchronous storage element input set-reset (SR) can be configured as either set or reset. This configuration option determines the state in which each flip-flop becomes operational after configuration. It also determines the effect of global set/reset pulse during normal operation and the effect of a pulse on the SR pin of the CLB. All three set/reset functions for any single stage can be disabled for either flip-flop. The set/reset state is specified by using the INIT attribute or by placing the appropriate set or reset flip-flop library symbol. SR is active high. It is not invertible within the CLB.

Global Set/Reset

A separate Global Set/Reset line sets or clears each storage element during power-up, reconfiguration or when a dedicated Reset net is driven active. This global net (GSR) does not compete with other routing resources; it uses a dedicated distribution network. Each flip-flop is configured as either globally set or reset in the same way that the local set/reset (SR) is specified. There fore, if a flip-flop is set by set-reset input, GSR also sets it. Similarly, a reset flip-flop is reset by both set-reset and global set reset (GSR). GSR can be driven from any user–programmable pin as a global reset input.

To use this global net, place an input pad and input buffer in the schematic or HDL code, driving the GSR pin of the startup symbol. A specific pin location can be assigned to this input using a loc attribute or property, just as with any other user programmable pad. An inverter can optionally be inserted after the input buffer to invert the sense of the global set/reset signal. Alternatively, GSR can be driven from any internal mode.

Data Inputs and Outputs

The Source of a storage element data input is programmable. It is driven by any of the functions F', G' and H', or by the Direct In (DIN) block input. The flip-flops or latches drive the NQ and YQ CLB outputs. Two fast feed through paths are available. A two-to-one multiplexer on each of the XQ and YQ outputs selects between a storage element output and any of the control inputs. This bypass is sometimes used by the automated router to re-power internal signals.

Control Signals of Chip

Multiplexers in the CLB map the four control inputs into the four internal control signals H1, DIN/H2, SR/H0 and EC. Any of these inputs can drive any of the four internal control signals.

Using Function Generator as RAM

Optional modes for each CLB make the memory look-up tables in the F' and G' function generators enable as an array of Read/Write memory cells. XC4000 series devices are the first programmable logic devices with synchronous edge trigger and dual port RAM accessible to the user. Edge triggered RAM simplifies system timing. Dual port RAM doubles the effective throughput of first in first out (FIFO) applications. These features can be individually programmed in any XC4000 family. The on-chip RAM is extremely fast. The read access time is the same as the logic delay. The write access time is slightly slower. Both access times are much faster than any off-chip solution, because they avoid I/O delays

3.7 Design flow & Implementation, by Xilinx1.5 tool

Design flow-Introduction:

Any design in digital circuit can be realized in using hardware descriptive language. Many vendors like Xilinx, Act, Altera, Phillips, mentor graphics and others comes out with powerful design tool. With that the realization of digital circuit can be achieved. Similarly, Exemplar (Leonardo spectrum) is a power-full tool for synthesis. Certain tool comes only with few options with specific manner. Here Xilinx1.5 is a typical tool, which supports following flow of design:

- Design Entry (text based, state machine editor, schematic editor)
- Synthesis & simulation.
- Implementation & timing and verification
- Programming and device download.

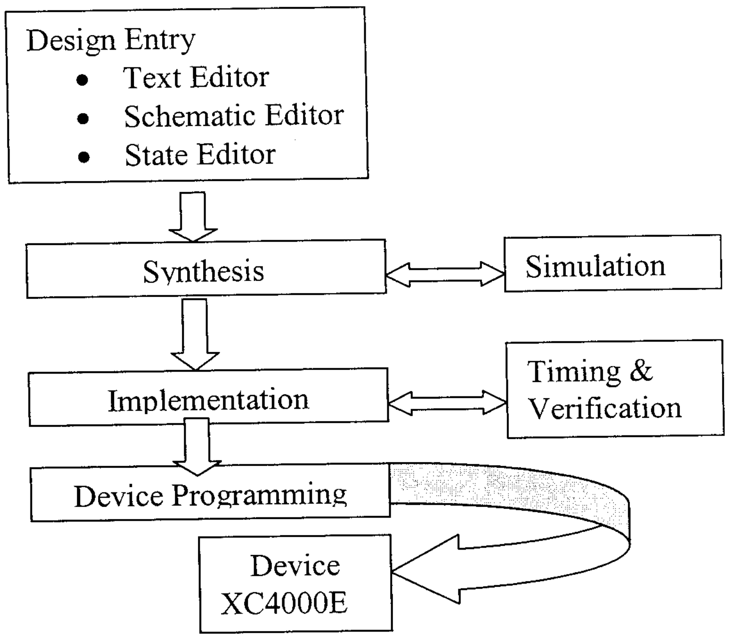


Fig: 12 Design flow

Design entry consists of three approach of design entry State machine editor. The editor gives designer; actual view of his design and this editor includes all inbuilt libraries of components of different companies, and families. This entry provides an interactive routing between different designs components.

The Xilinx 1.5 design tool provides two languages for it as VHDL, Verilog. These two languages were grouped under hardware descriptive language. Designs which are synthesizable are (i) State machine Editor's output file. (ii) Text Based Editor's output.
Further Schematic Editor doesn't need any synthesis as designer already chooses, desired component from the library.

Text Based Editors:

Xilinx 1.5 supports two Hardware descriptive languages like VHDL, and Verilog. Some features of the HDL were discussed. The hardware description language (HDLs) is used to describe the architecture and behavior of discrete electronic systems. HDLs were developed to deal with increasingly complex designs.

An analogy is often made to the development of software description languages, from the machine code to assembly language (net-lists), to high-level languages. Top-down, HDL based system design is most useful in large projects, where several designers or team of designers are working concurrently. HDLs provide structured development. After major architectural decisions have been made, and major components and their connections have been identified, work can proceed independently on subprojects.

3.8 DESCRIPTION OF THE PFC CONTROLLER:

The FPGA simulation is done using the Xilinx 1.5 design tool. The FPGA controller consists of two loops namely the Voltage loop and the current loop which are indicated in the figure below.

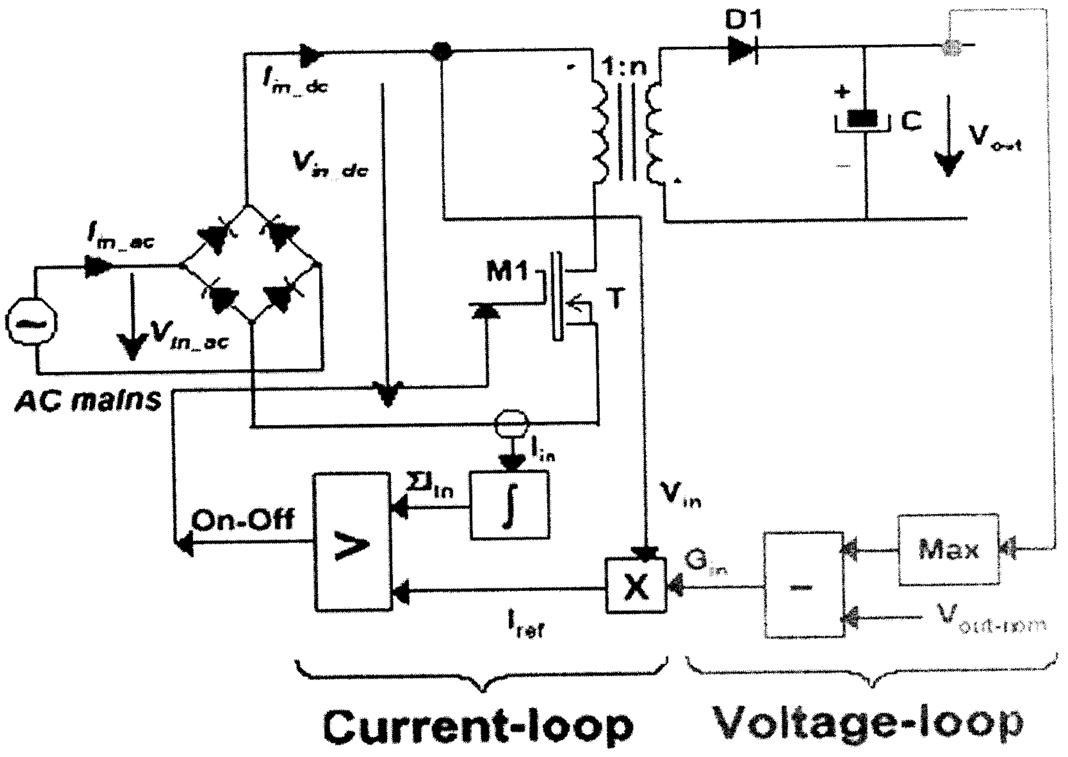


Fig: 13 circuit diagram for of PFC controller

3.8.1. CURRENT -LOOP CONTROLLER:

This loop controls the input current (I_{in}) in order to meet the PFC goal. Its purpose is to keep I_{in} proportional to V_{in}, in this case through the equivalent input conductance (G_{in}, the inverse of the equivalent input resistance R_{in}), is calculated by the voltage loop. The goal is to set the mean input current according to the following formula;

$$I_{in} = V_{in} \cdot G_{in}$$

DSP controllers perform this task varying the duty cycle that is sent to a PWM module. This means that I_{in} is controlled with a few switching cycles delay(40 to 100 μs for a 50 kHz switching frequency).The proposed controller implements the digital version of a charge control ,deciding every system clock cycle (every 50 ns for a 20 MHz FPGA clock) whether to keep the MOSFET on or off.

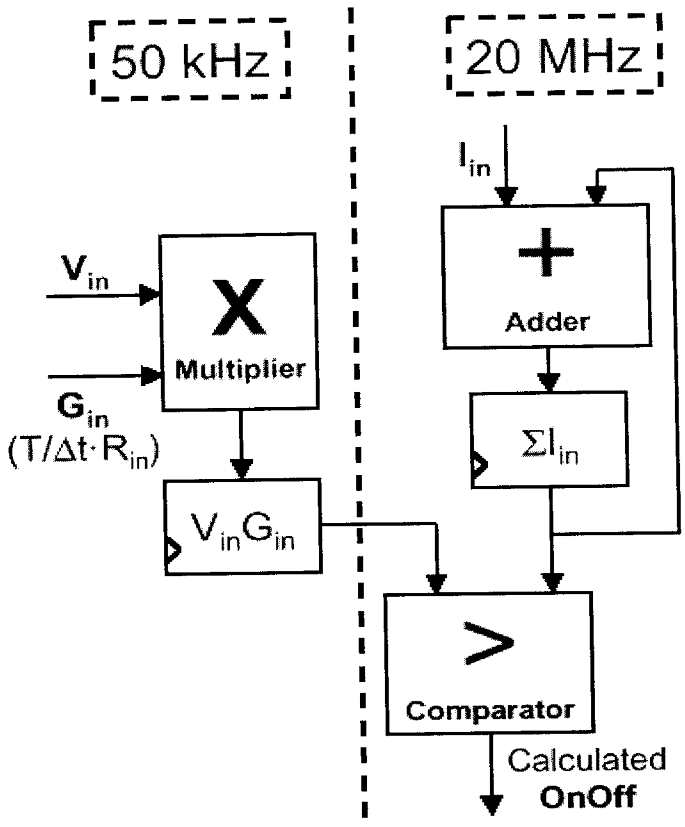


Fig: 14 Block diagram of Current loop

The working principle is to integrate I_{in} during a switching cycle (in the digital version, the integrator is substituted by a simple adder unit) until it reaches the target value, defined by the product of V_{in} by G_{in}. A high speed A/D converter is used for sampling I_{in} (a HI5805, a 5 MSPS 12-bite). The last value of I_{in} is added every FPGA clock cycle and the MOSFET is turned off when the target value is reached, controlling in this way the mean input current.

As it can be seen, both the adder and the comparator work at the system clock frequency (20 MHz). This method cannot be implemented in DSP because of its sequential nature. At least two instructions would be necessary for these two operations (add and comparison), and each instruction would consume more than a clock cycle, so that the process would be too slow using a DSP. Even more, this technique would consume almost all the DSP execution time, letting few resources for the rest of the control. However, the multiplier can work at a lower frequency because its both inputs are low frequency signals. This allows using a multiplier implemented with fewer resources because no optimization is necessary.

This loop, implemented as a digital charge control, is a good example of the FPGA advantages. It uses a high-speed algorithm in which all the resources are executed simultaneously. The hardware resources required for the algorithm implementation are quite small (just an adder, a comparator and multiplier) when compared with the traditional algorithms, many of them based on PID controllers. The main point, which makes it as accurate as the traditional algorithms, is the concurrency: not many resources, but executed simultaneously.

Some advantages of this current-loop are the PFC accuracy, the valid operation of both CCM and DCM and no need for a converter model. This method is valid independently of the inductor (L) value

3.8.2 VOLTAGE-LOOP CONTROLLER:

The previous loop makes I_{in} proportional to V_{in} through $G_{in,}$ therefore achieving power factor correction. However, that loop alone would leave the output voltage (V_{out}) uncontrolled. Therefore, the voltage-loop decides the Gin value in order to control $V_{out,}$ and consequently the input power (P_{in})

The whole control has only one output: the control signal sent to the switch in the power of converter. This signal is calculated by the current-loop as explained before. The second loop (the voltage-loop) actuates changing the G_{in} value that is sent to the current-loop. So one loop changes a parameter used in the other one. That is the way in which two different signals (I_{in} and V_{out}) are controlled with just one control signal.

The control formula has been calculated without any transfer function, just equaling the input and output power so V_{out} remains unchanged, based on the capacitor value, C. The result is a second order equation, but the implemented algorithm uses the first order equivalent equation:

$$\Delta G_{inp} = \frac{(C.V_{out-nom})}{(T_{Vin}.V_{inp-rms}^{2})}.(V_{out-nom} - V_{out})$$

T_{Vin} is the period of rectified V_{in} (10 ms), $V_{out-nom}$ is the nominal value for V_{out} (48V) and V_{in-rms} is also considered a constant (110 V), so V_{out} is the only variable. This algorithm is a proportional control that has shown a good dynamic response for controlling V_{out} recovering steady state within 3-4 cycles as shown in the experimental results. Its physical implement is reflected in figure below.

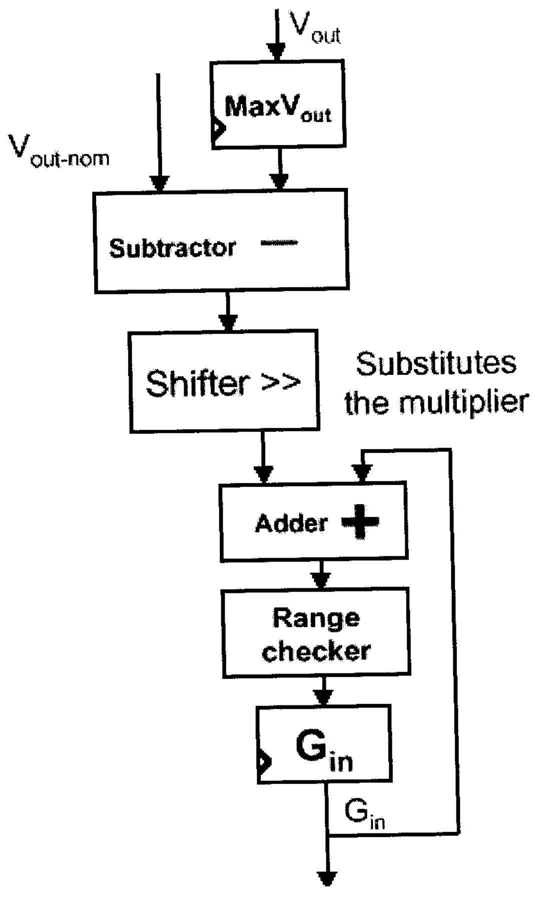


Fig: 15 Block diagram of Voltage loop

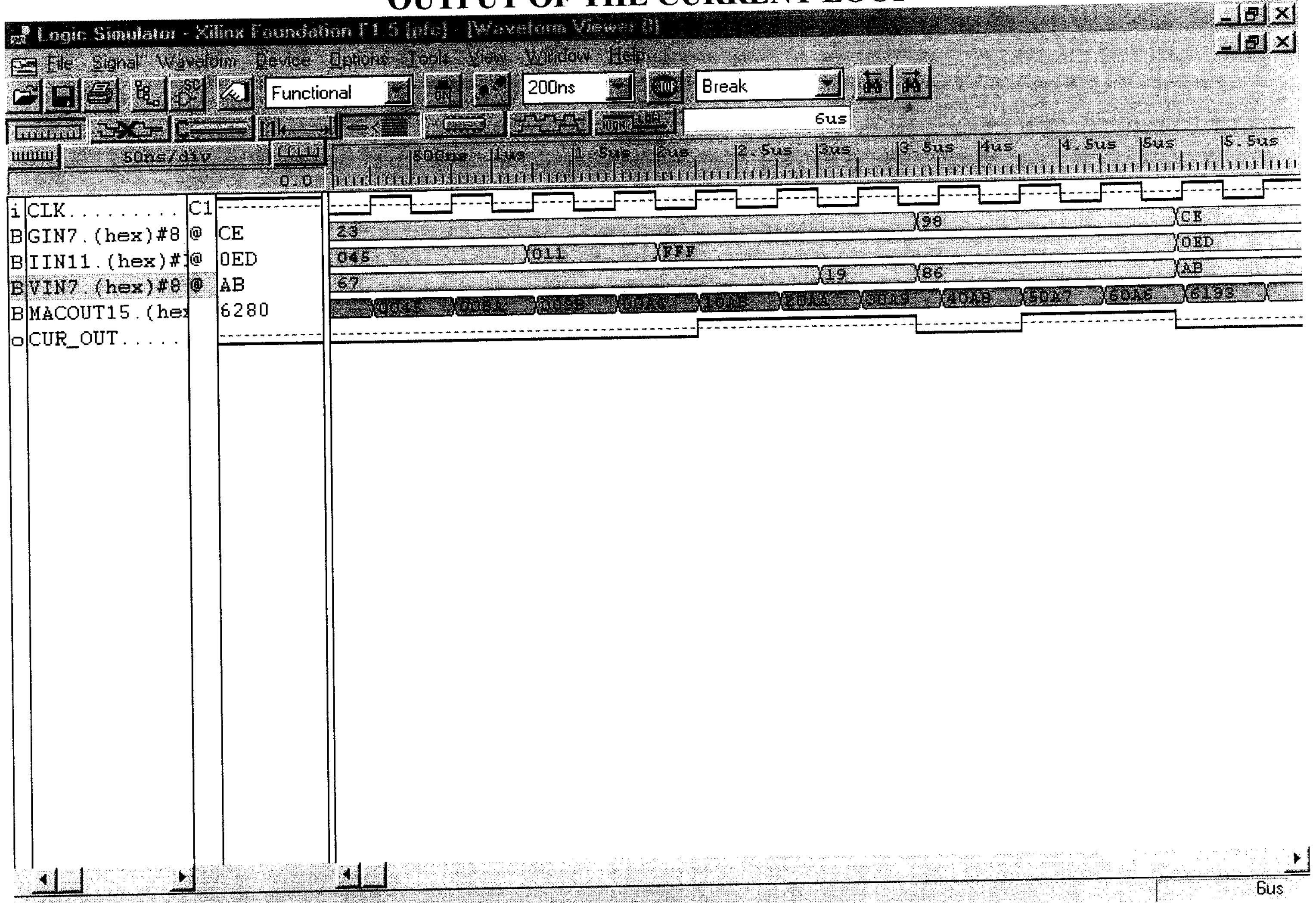
Proportional controls have a steady-state error (an offset in V_{out} from the target value). This error could be avoided using a PI control, but this would mean a big increase in the necessary resources. However, the error is not a problem in this application because the implemented proportional algorithm makes the error much smaller than the V_{out} ripple, inherent to PFC.

The implemented loop is quite simple. It has a subtractor for the difference between the nominal and measured V_{out}. The result of the subtraction has to be multiplied by a constant. However, this constant can always be adjusted to a power of two (changing the internal representation of any signal), so the multiplication can be substituted by a shifting operation, which is much simpler. Finally the value calculated represents the change from the previous value, so an additional adder is necessary. Therefore, the only necessary resources are the subtractor, a shifter and an adder. This allows keeping the whole control very simple, according to the design methodology here proposed.

In a power factor correction application, V_{in} has a ripple double in frequency then the AC mains (100 Hz). An advantage of using a digital control is avoiding the filter used for V_{out} which analog controls use for canceling that ripple. Some digital controls overcome this problem calculating the ripple and subtracting it from the measured V_{out}. However, this calculus is not a trivial one. We propose a much simpler method that consists in using the maximum value of V_{out} in each rectified V_{in} cycle .In this way, V_{out} does not need to be filtered and no further calculation is necessary. The main drawback of this method is that the changes in Gin happen only once every rectified V_{in} cycle, when a new V_{out} value is measured. In spite of it, a good dynamic response is achieved.

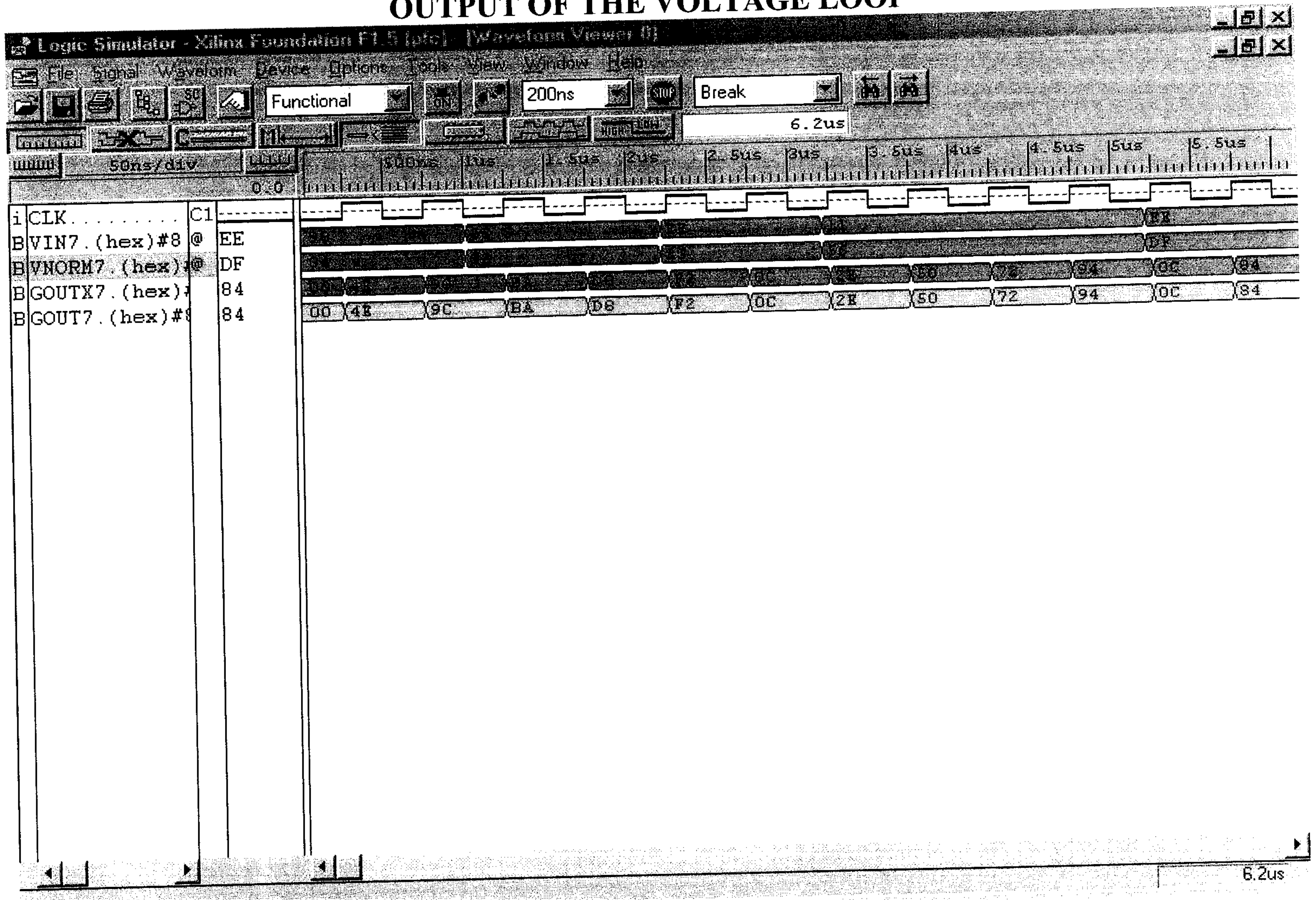
The outputs of the FPGA simulation are shown in the following pages.

OUTPUT OF THE CURRENT LOOP


Logic Simulator - Xilinx Foundation F1.5 [pfc] - [Waveform Viewer 0]
File Signal Waveform Device Options Tools View Window Help
Functional
200ns
Break
6us
50ns/div
0.0
500ns 1us 1.5us 2us 2.5us 3us 3.5us 4us 4.5us 5us 5.5us
23
98
CE
045
011
FFF
0ED
67
19
86
AB
40AB
5DA7
60A6
6193
6us

```
i CLK. . . . . . . . .  C1
B GIN7.(hex)#8      @  CE
B IIN11.(hex)#1     @  0ED
B VIN7.(hex)#8      @  AB
B MACOUT15.(hex        6280
o CUR_OUT. . . . .
```

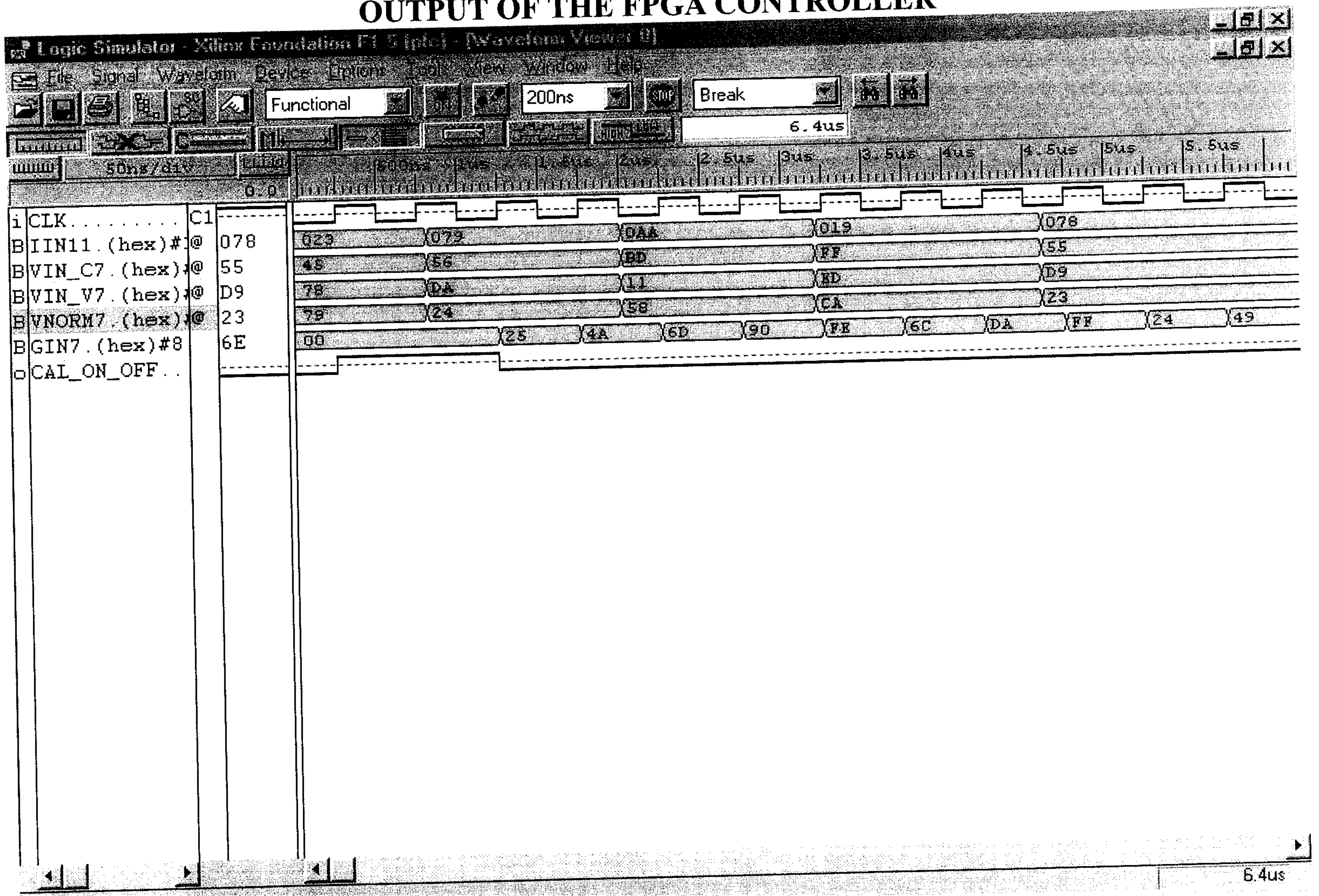
OUTPUT OF THE VOLTAGE LOOP


Logic Simulator - Xilinx Foundation F1.5 [ptc] - [Waveform Viewer 0]
File Signal Waveform Device Options Tools View Window Help
Functional
200ns
Break
6.2us
50ns/div
0.0
500ns 1us 1.5us 2us 2.5us 3us 3.5us 4us 4.5us 5us 5.5us
6.2us

```
i CLK.........  C1
B VIN7.(hex)#8  @  EE
B VNORM7.(hex)  @  DF
B GOUTX7.(hex)     84
B GOUT7.(hex)#     84
```

GOUT7: 00 4E 9C BA D8 F2 0C 2E 50 72 94 0C 84

OUTPUT OF THE FPGA CONTROLLER


Logic Simulator - Xilinx Foundation F1.5 [plc] - [Waveform Viewer 0]
File Signal Waveform Device Options Tools View Window Help
Functional
200ns
Break
6.4us
50ns/div
0.0
500ns 1us 1.5us 2us 2.5us 3us 3.5us 4us 4.5us 5us 5.5us
6.4us

```
i CLK.........   C1
B IIN11.(hex)#1@ 078   023  079  0AA  019  078
B VIN_C7.(hex)#@ 55    45   56   BD   FF   55
B VIN_V7.(hex)#@ D9    78   DA   11   BD   D9
B VNORM7.(hex)#@ 23    79   24   58   CA   23
B GIN7.(hex)#8   6E    00   25  4A  6D  90  FE  6C  DA  FF  24  49
o CAL_ON_OFF..
```

3.9 DISCUSSION ON FPGA SIMULATION:

The simulation for the FPGA controller is done in such a way that if the three input variables; the input voltage, the input current and the output voltage are digitized and given as inputs to the controller; it generates a series of pulses for switching the MOSFET ON and OFF. This can be seen clearly from the screen shot given in the previous page. The 8 bit digital values of the output voltage (VIN_V7), the input voltage (VIN_C7) and the 12 bit digital value of the input current (IIN11) are given as inputs along with CLK (clock) signal to the FPGA controller. The output of the FPGA controller can be seen from the CAL_ON_OFF output in the figure.

One important thing that is to be noted in the simulation is that the output is obtained instantaneously after specifying the inputs. The operations of the Current loop (Addition and Comparison) take place at the system clock frequency so that a good control over the variation of current is obtained. The Voltage loop (Multiplier) however functions at the lower frequency because both its inputs are low frequency signals. The only delay when using the FPGA controller is the inertial delay of the hardware. This feature in FPGA controller is known as Concurrency. The main point that makes this method so accurate is this feature of concurrency: not many resources but executed simultaneously.

The operations of Multiplication, Addition, and Comparison are executed one after the other in a micro controller and each operation takes more than one clock cycle and so the execution time would be longer than the FPGA controller.

IMPLEMENTATION WITH PIC MICROCONTROLLER

4. IMPLEMENTATION WITH PIC MICROCONTROLLER

4.1 PIC Micro controller Core Features:

4.1.1 INTRODUCTION

• High-performance RISC CPU

• Only 35 single word instructions to learn

• All single cycle instructions except for program branches which are two cycle

• Operating speed: DC - 20 MHz clock input
DC - 200 ns instruction cycle

• Up to 8K x 14 words of Flash Program Memory,
Up to 368 x 8 bytes of Data Memory (RAM)
Up to 256 x 8 bytes of EEPROM data memory

• Pin out compatible to the PIC16C73/74/76/77

• Interrupt capability (up to 14 internal/external

• Eight level deep hardware stack

• Direct, indirect, and relative addressing modes

• Power-on Reset (POR)

• Power-up Timer (PWRT) and
Oscillator Start-up Timer (OST)

• Watchdog Timer (WDT) with its own on-chip RC Oscillator for reliable operation

• Programmable code-protection

• Power saving SLEEP mode

• Selectable oscillator options

• Low-power, high-speed CMOS EPROM/EEPROM technology

• Fully static design

• In-Circuit Serial Programming (ICSP) via two pins

• Only single 5V source needed for programming capability

• In-Circuit Debugging via two pins

• Processor read/write access to program memory

• Wide operating voltage range: 2.5V to 5.5V

• High Sink/Source Current: 25 mA

• Commercial and Industrial temperature ranges

• Low-power consumption:

< 2 mA typical @ 5V, 4 MHz

20mA typical @ 3V, 32 kHz

< 1mA typical standby current

4.1.2 Peripheral Features:

• Timer0: 8-bit timer/counter with 8-bit prescaler

• Timer1: 16-bit timer/counter with prescaler can be incremented during sleep via external crystal/clock

• Timer2: 8-bit timer/counter with 8-bit period register, prescaler and postscaler

• Two Capture, Compare, PWM modules

Capture is 16-bit, max resolution is 12.5 ns,

Compare is 16-bit, max resolution is 200 ns,

PWM max. Resolution is 10-bit

• 10-bit multi-channel Analog-to-Digital converter

• Synchronous Serial Port (SSP) with SPI. (Master Mode) and I2C. (Master/Slave)

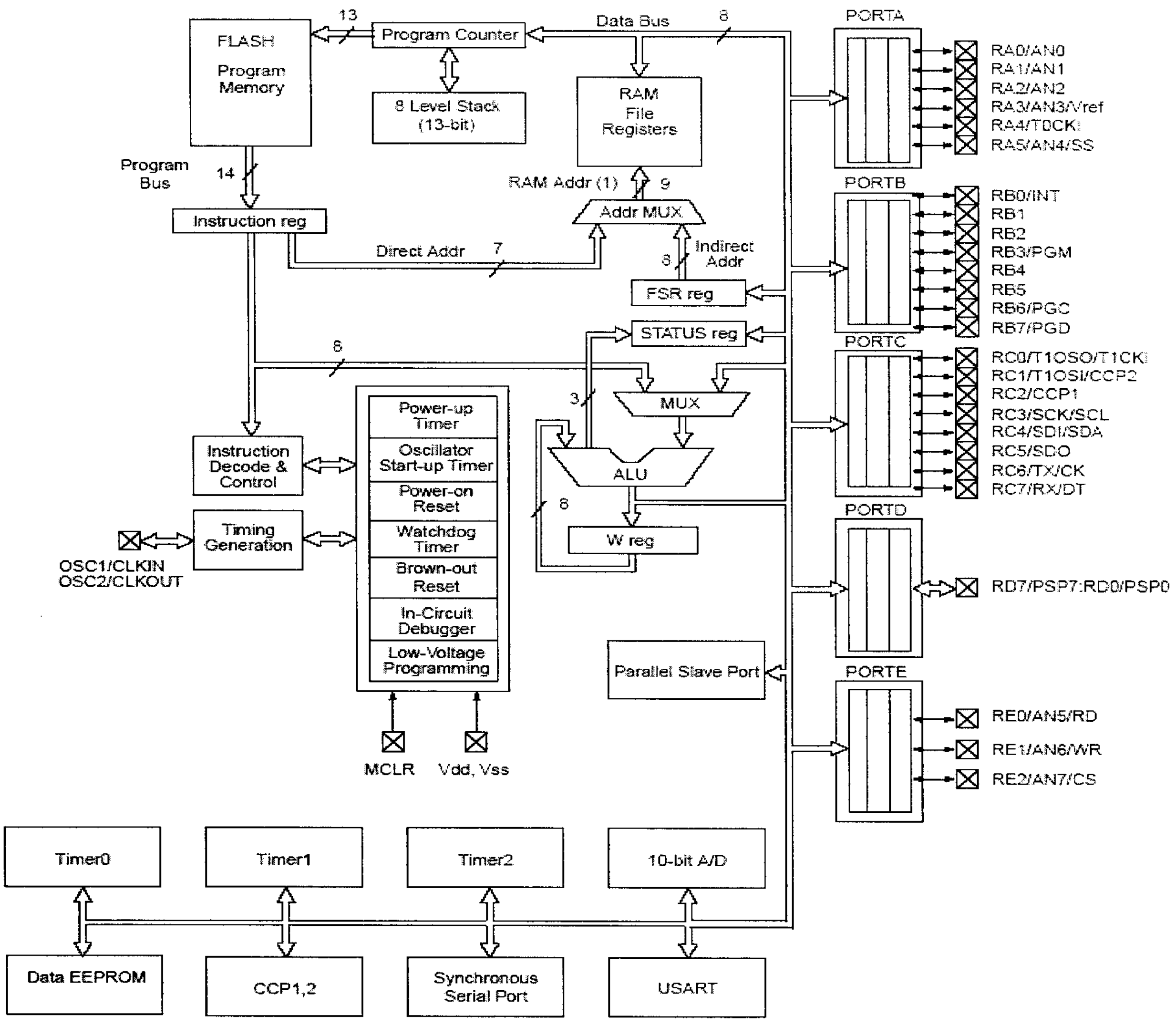
• Universal Synchronous Asynchronous Receiver Transmitter (USART/SCI) with 9-bit address detection.

• Parallel Slave Port (PSP) 8-bits wide, with external RD, WR and CS controls (40/44-pin only)

• Brownout detection circuitry **for Brownout Reset (BOR)**

4.1.3 ARCHITECTURE OF PIC 16F877 (FIG: 19)

Device	Program Flash	Data Memory	Data EEPROM
PIC16F874	4K	192 Bytes	128 Bytes
PIC16F877	8K	368 Bytes	256 Bytes



Note 1: Higher order bits are from the STATUS register.

Table 3: Memory Specification of 16F877 Microcontroller

DEVICE	PROGRAM FLASH	DATA MEMORY	DATA EEPROM
PIC 16F877	8K	368 Bytes	256 Bytes

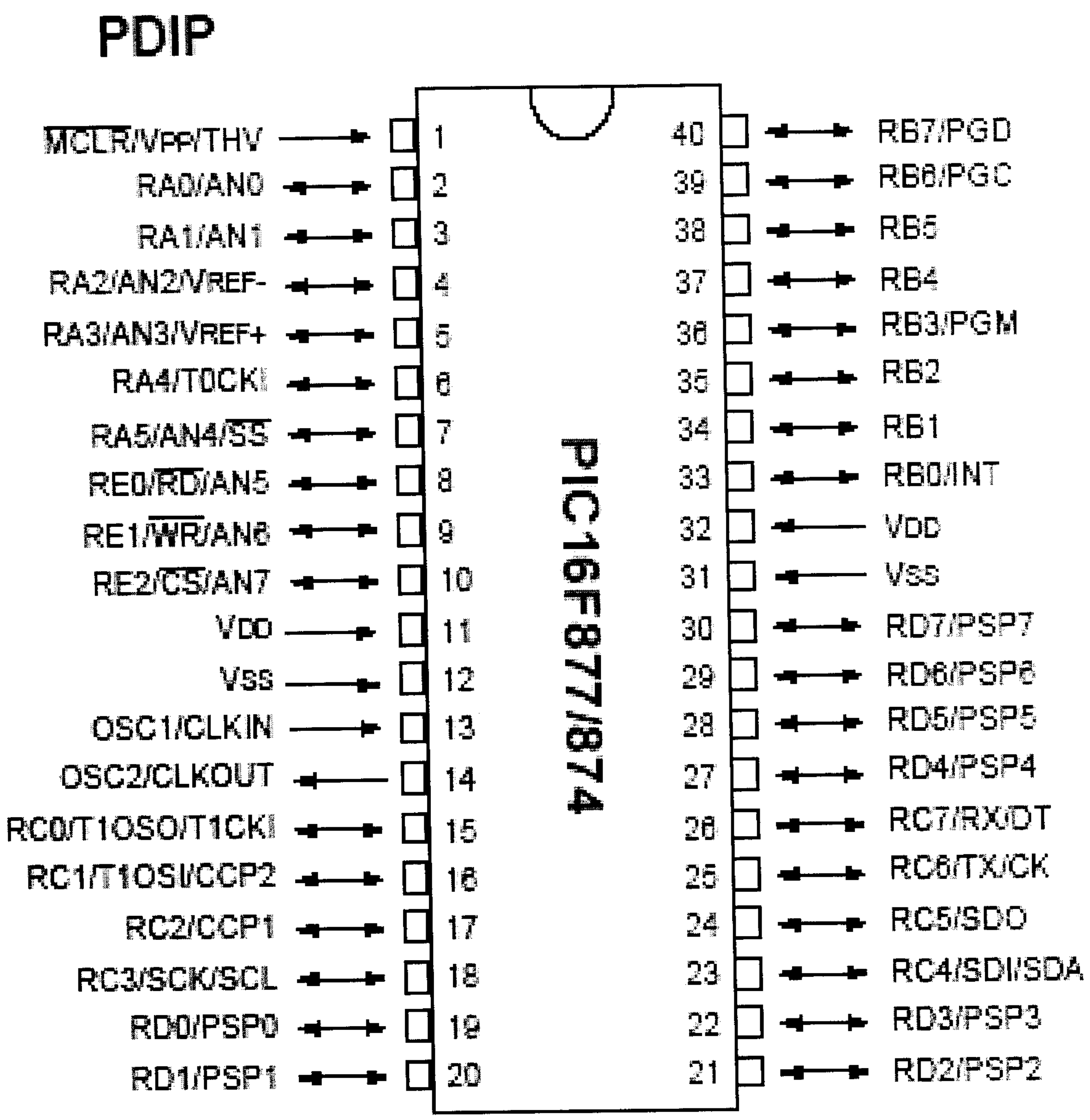


Fig: 20 Pin diagram for PIC16F877

Table 4: PIN OUT DESCRIPTION

Pin Name	DIP Pin#	PLCC Pin#	QFP Pin#	I/O/P Type	Buffer Type	Description
OSC1/CLKIN	13	14	30	I	ST/CMOS(4)	Oscillator crystal input/external clock source input.
OSC2/CLKOUT	14	15	31	O	—	Oscillator crystal output. Connects to crystal or resonator in crystal oscillator mode. In RC mode, OSC2 pin outputs CLKOUT which has 1/4 the frequency of OSC1, and denotes the instruction cycle rate.
$\overline{\text{MCLR}}$/VPP/THV	1	2	18	I/P	ST	Master clear (reset) input or programming voltage input or high voltage test mode control. This pin is an active low reset to the device.
						PORTA is a bi-directional I/O port.
RA0/AN0	2	3	19	I/O	TTL	RA0 can also be analog input0
RA1/AN1	3	4	20	I/O	TTL	RA1 can also be analog input1
RA2/AN2/VREF-	4	5	21	I/O	TTL	RA2 can also be analog input2 or negative analog reference voltage
RA3/AN3/VREF+	5	6	22	I/O	TTL	RA3 can also be analog input3 or positive analog reference voltage
RA4/T0CKI	6	7	23	I/O	ST	RA4 can also be the clock input to the Timer0 timer/counter. Output is open drain type.
RA5/$\overline{\text{SS}}$/AN4	7	8	24	I/O	TTL	RA5 can also be analog input4 or the slave select for the synchronous serial port.
						PORTB is a bi-directional I/O port. PORTB can be software programmed for internal weak pull-up on all inputs.
RB0/INT	33	36	8	I/O	TTL/ST(1)	RB0 can also be the external interrupt pin.
RB1	34	37	9	I/O	TTL	
RB2	35	38	10	I/O	TTL	
RB3/PGM	36	39	11	I/O	TTL	RB3 can also be the low voltage programming input
RB4	37	41	14	I/O	TTL	Interrupt on change pin.
RB5	38	42	15	I/O	TTL	Interrupt on change pin.
RB6/PGC	39	43	16	I/O	TTL/ST(2)	Interrupt on change pin or In-Circuit Debugger pin. Serial programming clock.
RB7/PGD	40	44	17	I/O	TTL/ST(2)	Interrupt on change pin or In-Circuit Debugger pin. Serial programming data.

Legend: I = input O = output I/O = input/output P = power

— = Not used TTL = TTL input ST = Schmitt Trigger input

Note

1. This buffer is a Schmitt Trigger input when configured as an external interrupt

2. This buffer is a Schmitt Trigger input when used in serial programming mode.

3. This buffer is a Schmitt Trigger input when configured as general purpose I/O and a TTL input when used in the Parallel Slave Port mode (for interfacing to a microprocessor bus).

4. This buffer is a Schmitt Trigger input when configured in RC oscillator mode and a CMOS input otherwise.

Pin Name	DIP Pin#	PLCC Pin#	QFP Pin#	I/O/P Type	Buffer Type	Description
						PORTC is a bi-directional I/O port.
RC0/T1OSO/T1CKI	15	16	32	I/O	ST	RC0 can also be the Timer1 oscillator output or a Timer1 clock input.
RC1/T1OSI/CCP2	16	18	35	I/O	ST	RC1 can also be the Timer1 oscillator input or Capture2 input/Compare2 output/PWM2 output.
RC2/CCP1	17	19	36	I/O	ST	RC2 can also be the Capture1 input/Compare1 output/PWM1 output.
RC3/SCK/SCL	18	20	37	I/O	ST	RC3 can also be the synchronous serial clock input/output for both SPI and I^2C modes.
RC4/SDI/SDA	23	25	42	I/O	ST	RC4 can also be the SPI Data In (SPI mode) or data I/O (I^2C mode).
RC5/SDO	24	26	43	I/O	ST	RC5 can also be the SPI Data Out (SPI mode).
RC6/TX/CK	25	27	44	I/O	ST	RC6 can also be the USART Asynchronous Transmit or Synchronous Clock.
RC7/RX/DT	26	29	1	I/O	ST	RC7 can also be the USART Asynchronous Receive or Synchronous Data.
						PORTD is a bi-directional I/O port or parallel slave port when interfacing to a microprocessor bus.
RD0/PSP0	19	21	38	I/O	ST/TTL(3)	
RD1/PSP1	20	22	39	I/O	ST/TTL(3)	
RD2/PSP2	21	23	40	I/O	ST/TTL(3)	
RD3/PSP3	22	24	41	I/O	ST/TTL(3)	
RD4/PSP4	27	30	2	I/O	ST/TTL(3)	
RD5/PSP5	28	31	3	I/O	ST/TTL(3)	
RD6/PSP6	29	32	4	I/O	ST/TTL(3)	
RD7/PSP7	30	33	5	I/O	ST/TTL(3)	
						PORTE is a bi-directional I/O port.
RE0/$\overline{\text{RD}}$/AN5	8	9	25	I/O	ST/TTL(3)	RE0 can also be read control for the parallel slave port, or analog input5.
RE1/$\overline{\text{WR}}$/AN6	9	10	26	I/O	ST/TTL(3)	RE1 can also be write control for the parallel slave port, or analog input6.
RE2/$\overline{\text{CS}}$/AN7	10	11	27	I/O	ST/TTL(3)	RE2 can also be select control for the parallel slave port, or analog input7.
Vss	12,31	13,34	6,29	P	—	Ground reference for logic and I/O pins.
VDD	11,32	12,35	7,28	P	—	Positive supply for logic and I/O pins.
NC	—	1,17,28, 40	12,13, 33,34		—	These pins are not internally connected. These pins should be left unconnected.

Legend: I = input O = output I/O = input/output P = power

— = Not used TTL = TTL input ST = Schmitt Trigger input

Note

1. This buffer is a Schmitt Trigger input when configured as an external interrupt.
2. This buffer is a Schmitt Trigger input when used in serial programming mode.
3. This buffer is a Schmitt Trigger input when configured as general purpose I/O and a TTL input when used in the Parallel Slave Port mode (for interfacing to a microprocessor bus).
4. This buffer is a Schmitt Trigger input when configured in RC oscillator mode and a CMOS input otherwise.

4.1.4. I/O PORTS

Some pins for these I/O ports are multiplexed with an alternate function for the peripheral features on the device. In general, when a peripheral is enabled, that pin may not be used as a general purpose I/O pin.

Additional Information on I/O ports may be found in the ICmicro™ Mid-Range Reference Manual,

PORTA and the TRISA Register:

PORTA is a 6-bit wide bi-directional port. The corresponding data direction register is TRISA. Setting a TRISA bit (=1) will make the corresponding PORTA pin an input, i.e., put the corresponding output driver in a Hi-impedance mode. Clearing a TRISA bit (=0) will make the corresponding PORTA pin an output, i.e., put the contents of the output latch on the selected pin. Reading the PORTA register reads the status of the pins whereas writing to it will write to the port latch. All write operations are read-modify-write operations. Therefore a write to a port implies that the port pins are read; this value is modified, and then written to the port data latch. Pin RA4 is multiplexed with the Timer0 module clock input to become the RA4/T0CKI pin. The RA4/T0CKI pin is a Schmitt Trigger input and an open drain output. All other RA port pins have TTL input levels and full CMOS output drivers. Other PORTA pins are multiplexed with analog inputs and analog VREF input. The operation of each pin is selected by clearing/setting the control bits in the ADCON1 register (A/D Control Register1).

The TRISA register controls the direction of the RA pins, even when they are being used as analog inputs. The user must ensure the bits in the TRISA register are maintained set when using them as analog inputs.

Table 5: PORT A FUNCTIONS

Name	Bit#	Buffer	Function
RA0/AN0	bit0	TTL	Input/output or analog input
RA1/AN1	bit1	TTL	Input/output or analog input
RA2/AN2	bit2	TTL	Input/output or analog input
RA3/AN3/VREF	bit3	TTL	Input/output or analog input or VREF
RA4/T0CKI	bit4	ST	Input/output or external clock input for Timer0 Output is open drain type
RA5/$\overline{SS}$/AN4	bit5	TTL	Input/output or slave select input for synchronous serial port or analog input

Legend: TTL = TTL input, ST = Schmitt Trigger input

Table 6: SUMMARY OF REGISTERS ASSOCIATED WITH PORT A

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on: POR, BOR	Value on all other resets
05h	PORTA	—	—	RA5	RA4	RA3	RA2	RA1	RA0	--0x 0000	--0u 0000
85h	TRISA	—	—	PORTA Data Direction Register						--11 1111	--11 1111
9Fh	ADCON1	—	—	ADFM	—	PCFG3	PCFG2	PCFG1	PCFG0	--0- 0000	--0- 0000

Legend: x = unknown, u = unchanged, - = unimplemented locations read as '0'. Shaded cells are not used by PORTA.

PORTB and the TRISB Register:

PORTB is an 8-bit wide bi-directional port. The corresponding data direction register is TRISB. Setting a TRISB bit (=1) will make the corresponding PORTB pin an input, i.e., put the corresponding output driver in a hi-impedance mode. Clearing a TRISB bit (=0) will make the corresponding PORTB pin an output, i.e., put the contents of the output latch on the selected pin. Three pins of PORTB are multiplexed with the Low Voltage Programming function; RB3/PGM, RB6/PGC and RB7/PGD. The alternate functions of these pins are described in the Special Features Section. Each of the PORTB pins has a weak internal pull-up. A single control bit can turn on all the pull-ups. Clearing bit performs this RBPU (OPTION_REG<7>). The weak pull-up is automatically turned off when the port pin is configured as an output. The pull-ups are disabled on a Power-on Reset.

Four of PORTB's pins, RB7:RB4, have an interrupt on change feature. Only pins configured as inputs can cause this interrupt to occur (i.e. any RB7:RB4 pin configured as an output is excluded from the interrupt on change comparison). The input pins (of RB7:RB4) are compared with the old value latched on the last read of PORTB. The "mismatch" outputs of RB7:RB4 are OR'ed together to generate the RB Port Change Interrupt with flag bit RBIF (INTCON<0>). This interrupt can wake the device from SLEEP. The user, in the interrupt service routine, can clear the interrupt in the following manner:

a) Any read or write of PORTB. This will end the mismatch condition.

b) Clear flag bit RBIF. A mismatch condition will continue to set flag bit RBIF. Reading PORTB will end the mismatch condition, and allow flag bit RBIF to be cleared. The interrupt on change feature is recommended for wake-up on key depression operation and operations where PORTB is only used for the interrupt on change feature. Polling of PORTB is not recommended while using the interrupt on change feature. This interrupt on mismatch feature, together with software configurable pull-ups on these four pins, allow easy interface to a keypad and make it possible for wake-up on key depression

Table 7: PORT B FUNCTIONS

Name	Bit#	Buffer	Function
RB0/INT	bit0	TTL/ST(1)	Input/output pin or external interrupt input. Internal software programmable weak pull-up.
RB1	bit1	TTL	Input/output pin. Internal software programmable weak pull-up.
RB2	bit2	TTL	Input/output pin. Internal software programmable weak pull-up.
RB3/PGM	bit3	TTL	Input/output pin or programming pin in LVP mode. Internal software programmable weak pull-up.
RB4	bit4	TTL	Input/output pin (with interrupt on change). Internal software programmable weak pull-up.
RB5	bit5	TTL	Input/output pin (with interrupt on change). Internal software programmable weak pull-up.
RB6/PGC	bit6	TTL/ST(2)	Input/output pin (with interrupt on change) or In-Circuit Debugger pin. Internal software programmable weak pull-up. Serial programming clock.
RB7/PGD	bit7	TTL/ST(2)	Input/output pin (with interrupt on change) or In-Circuit Debugger pin. Internal software programmable weak pull-up. Serial programming data.

Legend: TTL = TTL input, ST = Schmitt Trigger input
Note 1: This buffer is a Schmitt Trigger input when configured as the external interrupt.
2: This buffer is a Schmitt Trigger input when used in serial programming mode.

Table 8: SUMMARY OF REGISTERS ASSOCIATED WITH PORT B

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on: POR, BOR	Value on all other resets
06h, 106h	PORTB	RB7	RB6	RB5	RB4	RB3	RB2	RB1	RB0	xxxx xxxx	uuuu uuuu
86h, 186h	TRISB	PORTB Data Direction Register								1111 1111	1111 1111
81h, 181h	OPTION_REG	RBPU	INTEDG	T0CS	T0SE	PSA	PS2	PS1	PS0	1111 1111	1111 1111

Legend: x = unknown, u = unchanged. Shaded cells are not used by PORTB.

PORTC and the TRISC Register:

PORTC is an 8-bit wide bi-directional port. The corresponding data direction register is TRISC. Setting a TRISC bit (=1) will make the corresponding PORTC pin an input, i.e., put the corresponding output driver in a hi-impedance mode. Clearing a TRISC bit (=0) will make the corresponding PORTC pin an output, i.e., put the contents of the output latch on the selected pin. PORTC is multiplexed with several peripheral functions
(Table-3.5). PORTC pins have Schmitt Trigger input buffers.

When the I^2C module is enabled, the PORTC (3:4) pins can be configured with normal I^2C levels or with SMBUS levels by using the CKE bit (SSPSTAT <6>).

When enabling peripheral functions, care should be taken in defining TRIS bits for each PORTC pin. Some peripherals override the TRIS bit to make a pin an output, while other peripherals override the TRIS bit to make a pin an input. Since the TRIS bit override is in effect while the peripheral is enabled, read-modify write instructions (BSF, BCF, XORWF) with TRISC, as destination should be avoided. The user should refer to the corresponding peripheral section for the correct TRIS bit settings.

Table 9: PORT C FUNCTIONS

Name	Bit#	Buffer Type	Function
RC0/T1OSO/T1CKI	bit0	ST	Input/output port pin or Timer1 oscillator output/Timer1 clock input
RC1/T1OSI/CCP2	bit1	ST	Input/output port pin or Timer1 oscillator input or Capture2 input/ Compare2 output/PWM2 output
RC2/CCP1	bit2	ST	Input/output port pin or Capture1 input/Compare1 output/PWM1 output
RC3/SCK/SCL	bit3	ST	RC3 can also be the synchronous serial clock for both SPI and I^2C modes.
RC4/SDI/SDA	bit4	ST	RC4 can also be the SPI Data In (SPI mode) or data I/O (I^2C mode).
RC5/SDO	bit5	ST	Input/output port pin or Synchronous Serial Port data output
RC6/TX/CK	bit6	ST	Input/output port pin or USART Asynchronous Transmit or Synchronous Clock
RC7/RX/DT	bit7	ST	Input/output port pin or USART Asynchronous Receive or Synchronous Data

Legend: ST = Schmitt Trigger input

Table 10: SUMMARY OF REGISTERS ASSOCIATED WITH PORT C

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on: POR, BOR	Value on all other resets
07h	PORTC	RC7	RC6	RC5	RC4	RC3	RC2	RC1	RC0	xxxx xxxx	uuuu uuuu
87h	TRISC	PORTC Data Direction Register								1111 1111	1111 1111

Legend: x = unknown, u = unchanged.

PORTD and TRISD Registers:

This section is not applicable to the 28-pin devices. PORTD is an 8-bit port with Schmitt Trigger input buffers. Each pin is individually configurable as an input or output.

PORTD can be configured as an 8-bit wide microprocessor Port (parallel slave port) by setting control bit PSPMODE (TRISE<4>). In this mode, the input buffers are TTL.

Table 11: PORT D FUNCTIONS

Name	Bit#	Buffer Type	Function
RD0/PSP0	bit0	ST/TTL[1]	Input/output port pin or parallel slave port bit0
RD1/PSP1	bit1	ST/TTL[1]	Input/output port pin or parallel slave port bit1
RD2/PSP2	bit2	ST/TTL[1]	Input/output port pin or parallel slave port bit2
RD3/PSP3	bit3	ST/TTL[1]	Input/output port pin or parallel slave port bit3
RD4/PSP4	bit4	ST/TTL[1]	Input/output port pin or parallel slave port bit4
RD5/PSP5	bit5	ST/TTL[1]	Input/output port pin or parallel slave port bit5
RD6/PSP6	bit6	ST/TTL[1]	Input/output port pin or parallel slave port bit6
RD7/PSP7	bit7	ST/TTL[1]	Input/output port pin or parallel slave port bit7

Legend: ST = Schmitt Trigger input TTL = TTL input

Note 1: Input buffers are Schmitt Triggers when in I/O mode and TTL buffer when in Parallel Slave Port Mode.

Table 12: SUMMARY OF REGISTERS ASSOCIATED WITH PORT D

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on: POR, BOR	Value on all other resets
08h	PORTD	RD7	RD6	RD5	RD4	RD3	RD2	RD1	RD0	xxxx xxxx	uuuu uuuu
88h	TRISD	PORTD Data Direction Register								1111 1111	1111 1111
89h	TRISE	IBF	OBF	IBOV	PSPMODE	—	PORTE Data Direction Bits			0000 -111	0000 -111

Legend: x = unknown, u = unchanged, - = unimplemented read as '0'. Shaded cells are not used by PORTD.

PORTE and TRISE Registers:

PORTE has three pins RE0/RD/AN5, RE1/WR/AN6 and RE2/CS/AN7, which are individually configurable as inputs or outputs. These pins have Schmitt Trigger input buffers.

The PORTE pins become control inputs for the microprocessor port when bit PSPMODE (TRISE<4>) is set. In this mode, the user must make sure that the TRISE<2:0> bits are set (pins are configured as digital inputs). Ensure ADCON1 is configured for digital I/O. In this mode the input buffers are TTL.

PORTE pins are multiplexed with analog inputs. When selected as an analog input, these pins will read as '0's. TRISE controls the direction of the RE pins, even when they are being used as analog inputs. The user must make sure to keep the pins configured as inputs when using them as analog inputs.

Table 13: PORT E FUNCTIONS

Name	Bit#	Buffer Type	Function
RE0/$\overline{RD}$/AN5	bit0	ST/TTL(1)	Input/output port pin or read control input in parallel slave port mode or analog input: $\overline{RD}$ 1 = Not a read operation 0 = Read operation. Reads PORTD register (if chip selected)
RE1/$\overline{WR}$/AN6	bit1	ST/TTL(1)	Input/output port pin or write control input in parallel slave port mode or analog input: $\overline{WR}$ 1 =Not a write operation 0 =Write operation. Writes PORTD register (if chip selected)
RE2/$\overline{CS}$/AN7	bit2	ST/TTL(1)	Input/output port pin or chip select control input in parallel slave port mode or analog input: $\overline{CS}$ 1 = Device is not selected 0 = Device is selected

Legend: ST = Schmitt Trigger input TTL = TTL input
Note 1: Input buffers are Schmitt Triggers when in I/O mode and TTL buffers when in Parallel Slave Port Mode.

Table 14: SUMMARY OF REGISTERS ASSOCIATED WITH PORT E

Addr	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on: POR, BOR	Value on all other resets
09h	PORTE	—	—	—	—	—	RE2	RE1	RE0	---- -xxx	---- -uuu
89h	TRISE	IBF	OBF	IBOV	PSPMODE	—	PORTE Data Direction Bits			0000 -111	0000 -111
9Fh	ADCON1	—	—	ADFM	—	PCFG3	PCFG2	PCFG1	PCFG0	--0- 0000	--0- 0000

Legend: x = unknown, u = unchanged, - = unimplemented read as '0'. Shaded cells are not used by PORTE.

4.1.5. MEMORY ORGANISATION:

There are three memory blocks in each of the PIC16f877 MUCs. The program memory and Data Memory have separate buses so that concurrent access can occur.

PROGRAM MEMORY ORGANISATION

The PIC16f877 devices have a 13-bit program counter capable of addressing 8K *14 words of FLASH program memory. Accessing a location above the physically implemented address will cause a wraparound.

The RESET vector is at 0000h and the interrupt vector is at 0004h.

DATA MEMORY ORGANISATION

The data memory is partitioned into multiple banks that contain the General Purpose Registers and the special functions Registers. Bits RP1 (STATUS<6) and RP0 (STATYUS<5>) are the bank selected bits.

Table 15: Data Memory Organization

RP1:RP0	Banks
00	0
01	1
10	2
11	3

Each bank extends up to 7Fh (1238 bytes). The lower locations of each bank are reserved for the Special Function Registers. Above the Special Function Registers are General Purpose Registers, implemented as static RAM. All implemented banks contain special function registers. Some frequently used special function registers from one bank may be mirrored in another bank for code reduction and quicker access.

GENERAL PURPOSE REGISTER FILE

The register file can be accessed either directly or indirectly through the File Selected Register (FSR). The following figure shows the PIC16F877 REGISTER FILE MAP

Bank 0	File Address	Bank 1	File Address	Bank 2	File Address	Bank 3	File Address
Indirect addr.(*)	00h	Indirect addr.(*)	80h	Indirect addr.(*)	100h	Indirect addr.(*)	180h
TMR0	01h	OPTION_REG	81h	TMR0	101h	OPTION_REG	181h
PCL	02h	PCL	82h	PCL	102h	PCL	182h
STATUS	03h	STATUS	83h	STATUS	103h	STATUS	183h
FSR	04h	FSR	84h	FSR	104h	FSR	184h
PORTA	05h	TRISA	85h	▨	105h	▨	185h
PORTB	06h	TRISB	86h	PORTB	106h	TRISB	186h
PORTC	07h	TRISC	87h	▨	107h	▨	187h
PORTD (1)	08h	TRISD (1)	88h	▨	108h	▨	188h
PORTE (1)	09h	TRISE (1)	89h	▨	109h	▨	189h
PCLATH	0Ah	PCLATH	8Ah	PCLATH	10Ah	PCLATH	18Ah
INTCON	0Bh	INTCON	8Bh	INTCON	10Bh	INTCON	18Bh
PIR1	0Ch	PIE1	8Ch	EEDATA	10Ch	EECON1	18Ch
PIR2	0Dh	PIE2	8Dh	EEADR	10Dh	EECON2	18Dh
TMR1L	0Eh	PCON	8Eh	EEDATH	10Eh	Reserved(2)	18Eh
TMR1H	0Fh	▨	8Fh	EEADRH	10Fh	Reserved(2)	18Fh
T1CON	10h	▨	90h	General Purpose Register 16 Bytes	110h	General Purpose Register 16 Bytes	190h
TMR2	11h	SSPCON2	91h		111h		191h
T2CON	12h	PR2	92h		112h		192h
SSPBUF	13h	SSPADD	93h		113h		193h
SSPCON	14h	SSPSTAT	94h		114h		194h
CCPR1L	15h	▨	95h		115h		195h
CCPR1H	16h	▨	96h		116h		196h
CCP1CON	17h	▨	97h		117h		197h
RCSTA	18h	TXSTA	98h		118h		198h
TXREG	19h	SPBRG	99h		119h		199h
RCREG	1Ah	▨	9Ah		11Ah		19Ah
CCPR2L	1Bh	▨	9Bh		11Bh		19Bh
CCPR2H	1Ch	▨	9Ch		11Ch		19Ch
CCP2CON	1Dh	▨	9Dh		11Dh		19Dh
ADRESH	1Eh	ADRESL	9Eh		11Eh		19Eh
ADCON0	1Fh	ADCON1	9Fh		11Fh		19Fh
General Purpose Register 96 Bytes	20h	General Purpose Register 80 Bytes	A0h	General Purpose Register 80 Bytes	120h	General Purpose Register 80 Bytes	1A0h
			EFh		16Fh		1EFh
		accesses 70h-7Fh	F0h	accesses 70h-7Fh	170h	accesses 70h - 7Fh	1F0h
	7Fh		FFh		17Fh		1FFh

▨ Unimplemented data memory locations, read as '0'.

* Not a physical register.

Note 1: These registers are not implemented on 28-pin devices.

2: These registers are reserved, maintain these registers clear.

4.1.6. INSTRUCTION SET SUMMARY

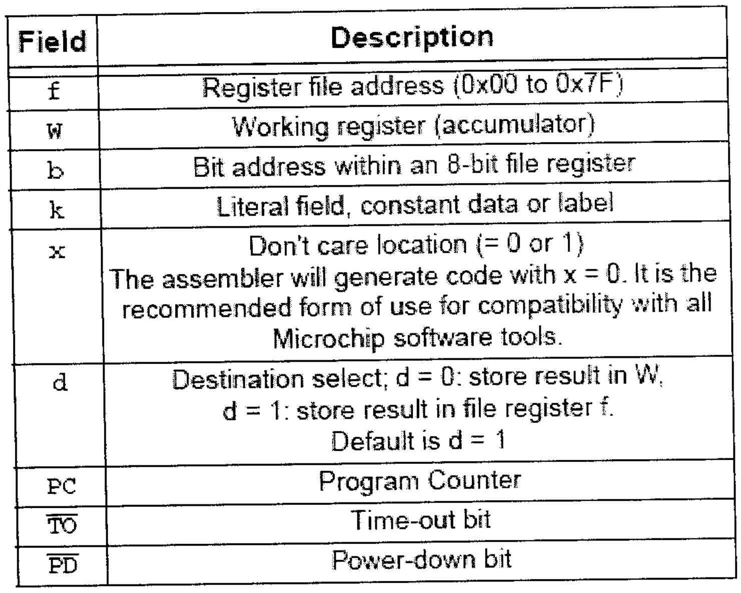
Each PIC 16F877 instruction is a 14-bit word, divided into an OPCODE that specifies the instruction type and one or more operand which further specify the operation of the instruction. The PIC16F877 instruction set summary in table 12 lists **byte-oriented, bit-oriented**, and **literal and control** operations. Table11 shows the opcode Field descriptions.

For byte-oriented instructions, 'f;' represents a file register designator and'd' represents a destination designator. The file register designator speciri4s that file register is to be used by the instruction. The destination designator specified where the result of the operation is to be placed. If'd' is zero, the result is placed in the w register. If'd' is one, the result is placed in the file register specified in the instruction.

For **bit-oriented** instructions, 'b' represents a bit field designator which selects the number of the bit affected by the operation, which 'f' represents the address of the file in which the bits is located.

For literal and control operations, 'k' represents an eight or eleven bit constant or literal value.

Table 16: OPCODE FIELD DESCRIPTIONS



Field	Description
f	Register file address (0x00 to 0x7F)
W	Working register (accumulator)
b	Bit address within an 8-bit file register
k	Literal field, constant data or label
x	Don't care location (= 0 or 1) The assembler will generate code with x = 0. It is the recommended form of use for compatibility with all Microchip software tools.
d	Destination select; d = 0: store result in W, d = 1: store result in file register f. Default is d = 1
PC	Program Counter
$\overline{\text{TO}}$	Time-out bit
$\overline{\text{PD}}$	Power-down bit

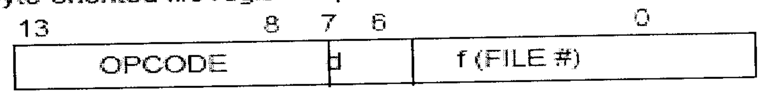
The instruction set is highly orthogonal and is grouped into three basic categories:

- **Byte-oriented** operations
- **Bit-oriented** operations
- **Literal and control** operations

All instructions are executed within one single instruction cycle, unless a conditional test is true or the program counter is changed as a result of an instruction. In this case, the execution takes two instruction cycles with the second cycle executed as a NOP. One instruction cycle consists of four oscillator periods. Thus, for an oscillator frequency of 4 MHz, the normal instruction execution time is 1 ms .If a conditional test is true or the program counter is changed as a result of an instruction, the instruction execution time is 2 ms.

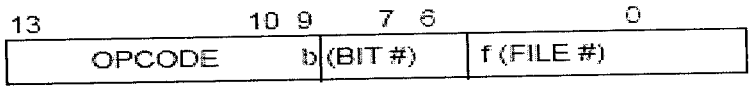
GENERAL FORMAT FOR INSTRUCTIONS

Byte-oriented file register operations



d = 0 for destination W
d = 1 for destination f
f = 7-bit file register address

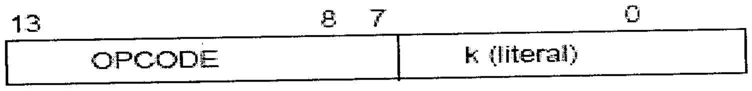
Bit-oriented file register operations



b = 3-bit bit address
f = 7-bit file register address

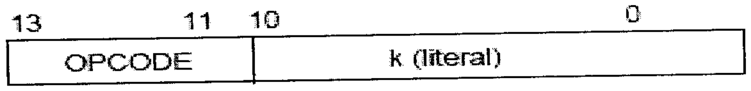
Literal and control operations

General



k = 8-bit immediate value

CALL and GOTO instructions only



k = 11-bit immediate value

16F877 INSTRUCTION SET

The PIC 16F877 instruction set is given below.

Mnemonic, Operands		Description	Cycles	14-Bit Opcode				Status Affected	Notes
				MSb			LSb		
BYTE-ORIENTED FILE REGISTER OPERATIONS									
ADDWF	f, d	Add W and f	1	00	0111	dfff	ffff	C,DC,Z	1,2
ANDWF	f, d	AND W with f	1	00	0101	dfff	ffff	Z	1,2
CLRF	f	Clear f	1	00	0001	1fff	ffff	Z	2
CLRW	-	Clear W	1	00	0001	0xxx	xxxx	Z	
COMF	f, d	Complement f	1	00	1001	dfff	ffff	Z	1,2
DECF	f, d	Decrement f	1	00	0011	dfff	ffff	Z	1,2
DECFSZ	f, d	Decrement f, Skip if 0	1(2)	00	1011	dfff	ffff		1,2,3
INCF	f, d	Increment f	1	00	1010	dfff	ffff	Z	1,2
INCFSZ	f, d	Increment f, Skip if 0	1(2)	00	1111	dfff	ffff		1,2,3
IORWF	f, d	Inclusive OR W with f	1	00	0100	dfff	ffff	Z	1,2
MOVF	f, d	Move f	1	00	1000	dfff	ffff	Z	1,2
MOVWF	f	Move W to f	1	00	0000	1fff	ffff		
NOP	-	No Operation	1	00	0000	0xx0	0000		
RLF	f, d	Rotate Left f through Carry	1	00	1101	dfff	ffff	C	1,2
RRF	f, d	Rotate Right f through Carry	1	00	1100	dfff	ffff	C	1,2
SUBWF	f, d	Subtract W from f	1	00	0010	dfff	ffff	C,DC,Z	1,2
SWAPF	f, d	Swap nibbles in f	1	00	1110	dfff	ffff		1,2
XORWF	f, d	Exclusive OR W with f	1	00	0110	dfff	ffff	Z	1,2
BIT-ORIENTED FILE REGISTER OPERATIONS									
BCF	f, b	Bit Clear f	1	01	00bb	bfff	ffff		1,2
BSF	f, b	Bit Set f	1	01	01bb	bfff	ffff		1,2
BTFSC	f, b	Bit Test f, Skip if Clear	1 (2)	01	10bb	bfff	ffff		3
BTFSS	f, b	Bit Test f, Skip if Set	1 (2)	01	11bb	bfff	ffff		3
LITERAL AND CONTROL OPERATIONS									
ADDLW	k	Add literal and W	1	11	111x	kkkk	kkkk	C,DC,Z	
ANDLW	k	AND literal with W	1	11	1001	kkkk	kkkk	Z	
CALL	k	Call subroutine	2	10	0kkk	kkkk	kkkk		
CLRWDT	-	Clear Watchdog Timer	1	00	0000	0110	0100	$\overline{TO}$,$\overline{PD}$	
GOTO	k	Go to address	2	10	1kkk	kkkk	kkkk		
IORLW	k	Inclusive OR literal with W	1	11	1000	kkkk	kkkk	Z	
MOVLW	k	Move literal to W	1	11	00xx	kkkk	kkkk		
RETFIE	-	Return from interrupt	2	00	0000	0000	1001		
RETLW	k	Return with literal in W	2	11	01xx	kkkk	kkkk		
RETURN	-	Return from Subroutine	2	00	0000	0000	1000		
SLEEP	-	Go into standby mode	1	00	0000	0110	0011	$\overline{TO}$,$\overline{PD}$	
SUBLW	k	Subtract W from literal	1	11	110x	kkkk	kkkk	C,DC,Z	
XORLW	k	Exclusive OR literal with W	1	11	1010	kkkk	kkkk	Z	

Note 1: When an I/O register is modified as a function of itself (e.g., `MOVF PORTB, 1`), the value used will be that value present on the pins themselves. For example, if the data latch is '1' for a pin configured as input and is driven low by an external device, the data will be written back with a '0'.

2: If this instruction is executed on the TMR0 register (and, where applicable, d = 1), the prescaler will be cleared if assigned to the Timer0 Module.

3: If Program Counter (PC) is modified or a conditional test is true, the instruction requires two cycles. The second cycle is executed as a NOP.

4.2. SYSTEM COMPONENTS

4.2.1. FLY-BACK CONVERTER CIRCUIT:

A normal fly back converter is used in this project. The essential necessity for a fly back converter was already explained in the second chapter and is reiterated here. The input to power factor pre-regulator (PFP) is a full-rectified sinusoidal waveform and the output of the power factor pre-regulator is a constant DC voltage. Therefore, it becomes essential that the voltage transfer ratio has to be maximum at the angles 0 and π relatively to what it has to be at $\pi/2$. So, as a result there has to be a continuous variation in voltage transfer ratio throughout a half line period. The high voltage transfer ratio at the zero and π angles can be achieved by using boost, buck-boost or fly back topologies. Buck topology cannot provide high voltage transfer ratio

A fly back converter provides a variety of voltage by chopping the input DC into a high frequency rectangular wave that can be passed through the pulse transformer to several secondary windings. This transformer also provides isolation, replacing the large, heavy, and expensive line frequency transformer.

The high voltage sine is the most common waveform. This may be converted immediately to DC and then stepped to the variety of regulated voltage provided by a fly back converter.

The buck regulator produces a single voltage lower than its input. The boost's single output is always above its input. If the input voltage begins above the output voltage, but falls below the output voltage (such as with a battery discharge neither alone works)

The fly-back converter replaces the inductor of the boost with a transformer primary. There may be as many secondaries as you need and for each you need only a diode and capacitor. There are no inductors.

Now that there is a transformer in the system, the large, heavy, expensive line transformer can be rectified and filtered, sending a high voltage DC to the fly-back transformer. This transformer provides the isolation between the line and your regulated voltage. Since the fly-back converter is running above audio frequencies, that transformer is small, light, available as a surface mount component, and less expensive.

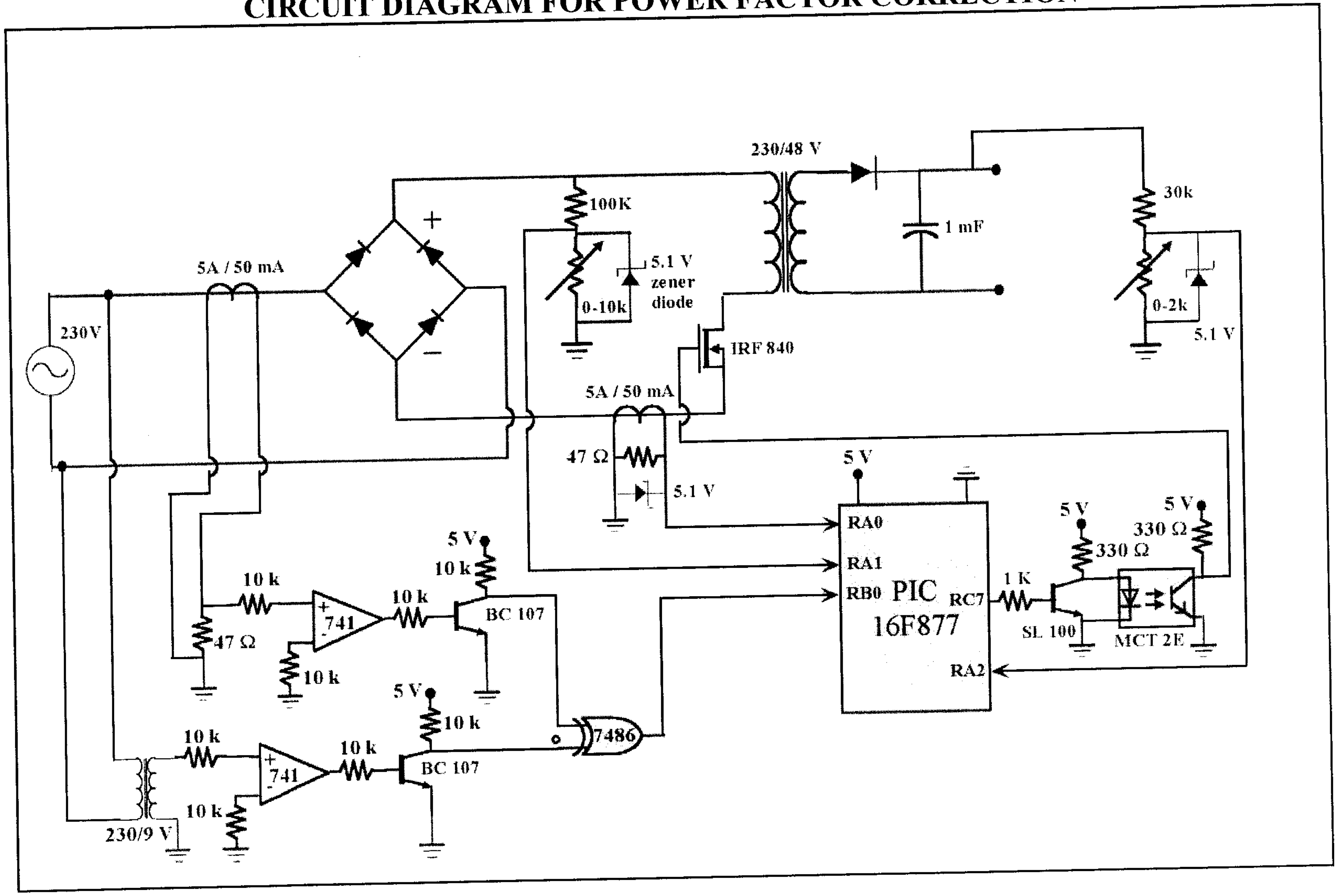
The key to a fly-back regulator's operation is that the primary and the secondaries are out of phase. When the switch is ON the phase difference will be opposite in the windings and the diode will be reverse biased and hence it remains OFF.

Since no current flows in any of the secondaries, the primary acts as a simple inductor. Current ramps in it, just as it does in the Boost regulator. When the switch turns OFF, the field in the core reverses direction, and begins to fall. This reversal of direction changes the polarities of the voltage at all of the secondary windings. The dots are induced positive. The diodes turn on and energy is transferred to the output capacitors and loads. The loads are powered on the fly-back part of each cycle, thus the converters name.

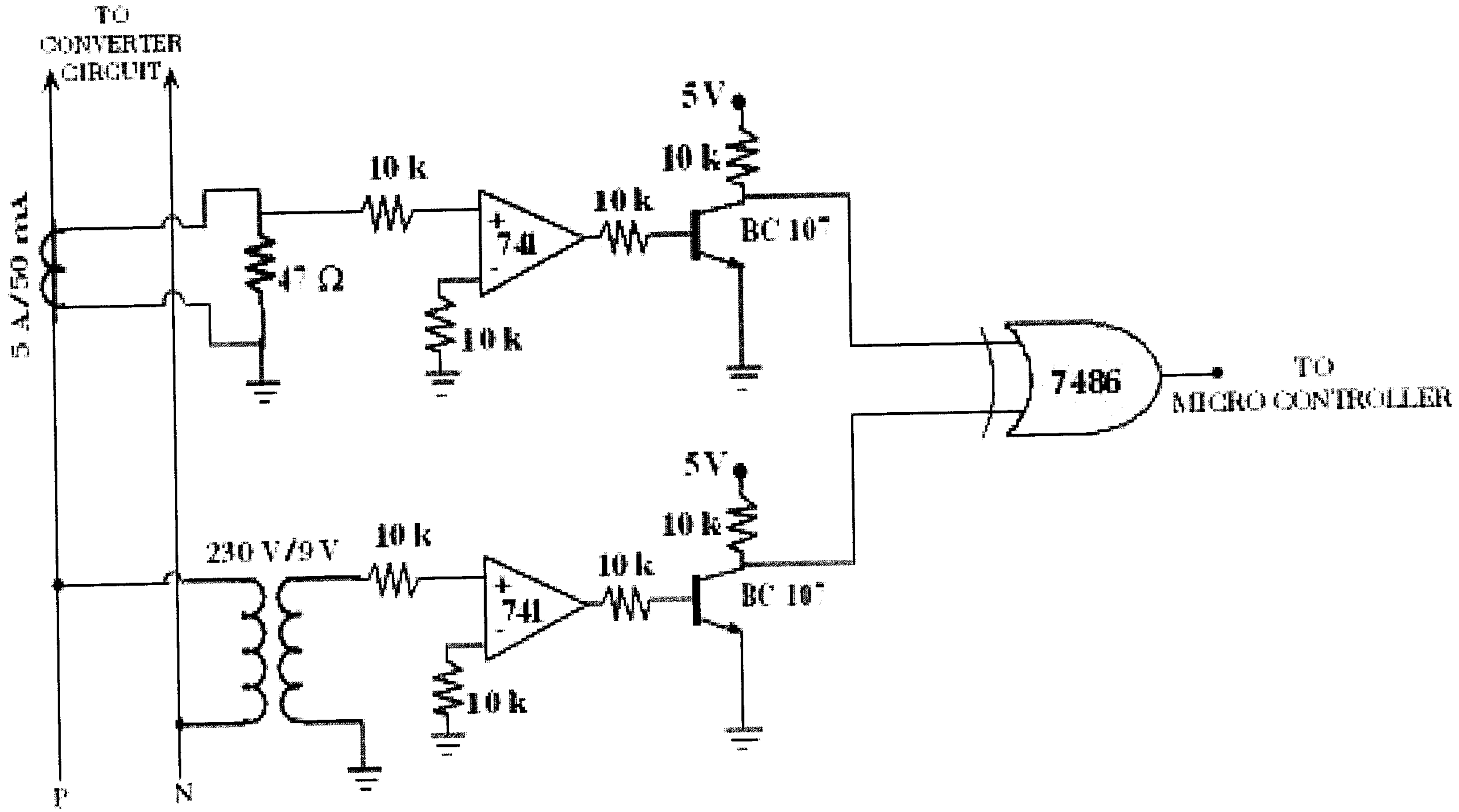
4.2.2. POWER FACTOR DETECTION CIRCUIT:

The Voltage and the current through the rectifier circuit are sensed using the potential and current transformers, the output of which is applied as an input to the zero-crossing detector. This produces two square wave outputs which correspond to the voltage across and the current through the bridge circuit. Both the square waves are applied as input to the XOR gate which produces a positive pulse output, the width of which corresponds to the input power factor of the circuit.

CIRCUIT DIAGRAM FOR POWER FACTOR CORRECTION



CIRCUIT DIAGRAM FOR POWER FACTOR DETECTION



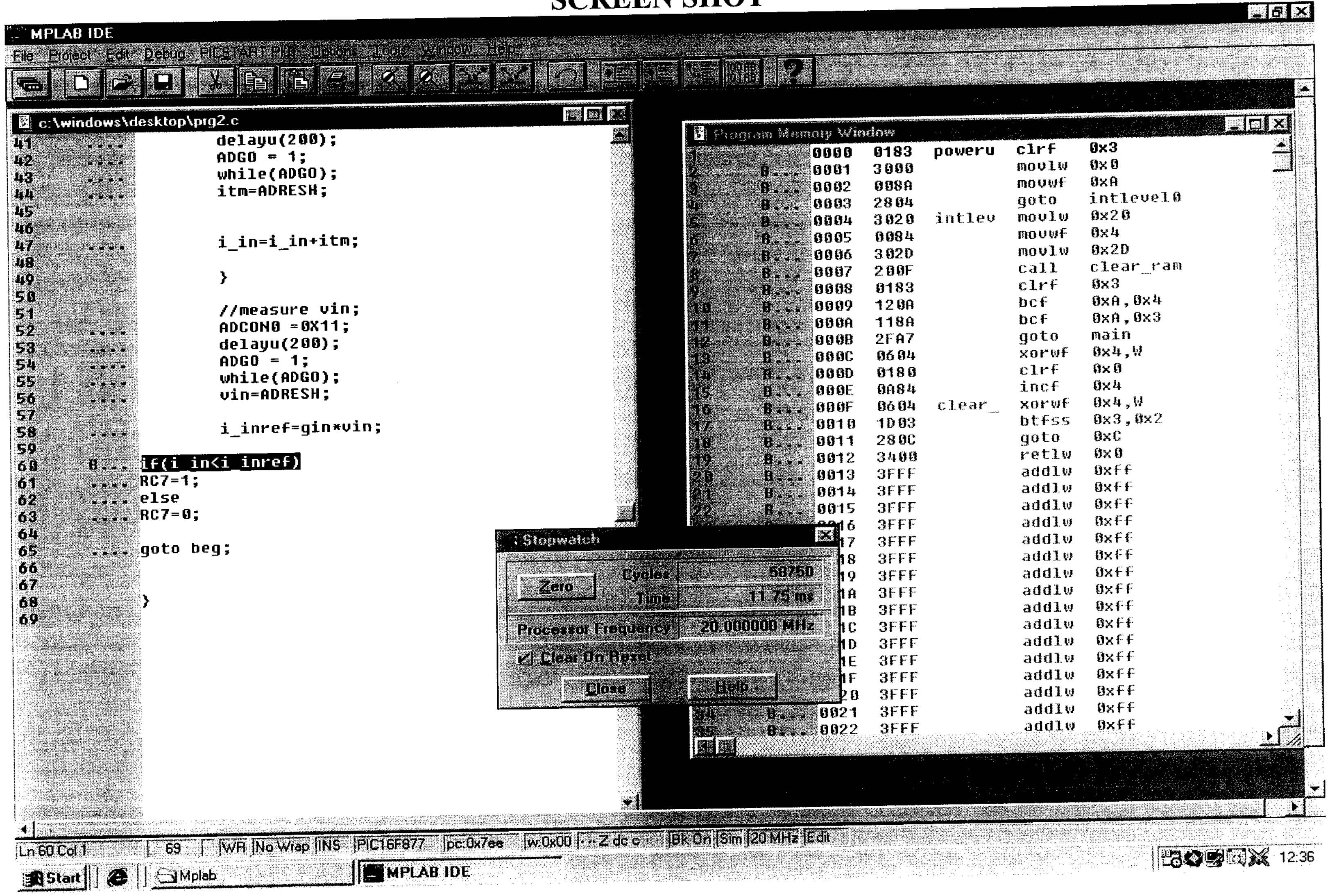
4.2.3 MICROCONTROLLER PROGRAMMING:

The microcontroller is programmed and is tested with the hardware circuit. The program is given in Appendix II. The digitized inputs are connected to the microcontroller parts and the output from the microcontroller is used to drive the MOSFET.

4.2.4 TEST RESULTS:

Unlike FPGA, the microcontroller is a sequential device and therefore processes do not take place concurrently but one after the other. This deteriorates the switching speed of MOSFET from kHz to Hz. As a result, When the MOSFET is switched at a frequency very less compared to the frequency it had to be switched; the power factor does not improve. The Conversion efficiency becomes very less. This leads to the very low improvement in power factor which cannot be noticed. The screen-shot displaying the time taken by the microcontroller for performing this operation is shown in the next leaf.

SCREEN SHOT



MPLAB IDE

c:\windows\desktop\prg2.c

```
        delayu(200);
        ADGO = 1;
        while(ADGO);
        itm=ADRESH;


        i_in=i_in+itm;

        }

        //measure vin;
        ADCON0 =0X11;
        delayu(200);
        ADGO = 1;
        while(ADGO);
        vin=ADRESH;

        i_inref=gin*vin;

if(i_in<i_inref)
RC7=1;
else
RC7=0;

goto beg;


}
```

Program Memory Window

```
0000  0183  poweru  clrf   0x3
0001  3000          movlw  0x0
0002  008A          movwf  0xA
0003  2804          goto   intlevel0
0004  3020  intlev  movlw  0x20
0005  0084          movwf  0x4
0006  302D          movlw  0x2D
0007  200F          call   clear_ram
0008  0183          clrf   0x3
0009  120A          bcf    0xA,0x4
000A  118A          bcf    0xA,0x3
000B  2FA7          goto   main
000C  0604          xorwf  0x4,W
000D  0180          clrf   0x0
000E  0A84          incf   0x4
000F  0604  clear_  xorwf  0x4,W
0010  1D03          btfss  0x3,0x2
0011  280C          goto   0xC
0012  3400          retlw  0x0
0013  3FFF          addlw  0xff
0014  3FFF          addlw  0xff
0015  3FFF          addlw  0xff
   6  3FFF          addlw  0xff
  17  3FFF          addlw  0xff
  18  3FFF          addlw  0xff
  19  3FFF          addlw  0xff
  1A  3FFF          addlw  0xff
  1B  3FFF          addlw  0xff
  1C  3FFF          addlw  0xff
  1D  3FFF          addlw  0xff
  1E  3FFF          addlw  0xff
  1F  3FFF          addlw  0xff
  20  3FFF          addlw  0xff
0021  3FFF          addlw  0xff
0022  3FFF          addlw  0xff
```

Stopwatch

Cycles 58750
Time 11.75 ms
Processor Frequency 20.000000 MHz
✓ Clear On Reset

File Project Edit Debug PICSTART Plus Options Tools Window Help
Zero
Close
Help
Ln 60 Col 1 | 69 | WR | No Wrap | INS | PIC16F877 | pc:0x7ee | w:0x00 | --Z dc c | Bk On | Sim | 20 MHz | Edit
Start
Mplab
MPLAB IDE
12:36

Since the switching speed of MOSFET is very low, it does not lead to any viewable improvement in the power factor. The algorithm was changed and tested for results with little modifications.

According to the above algorithm, the change in conductance for the change in error voltage was calculated using the formula,

$$\Delta G_{inp} = \frac{(C.V_{out-nom})}{(T_{Vin}.V_{inp-rms}^2)}.(V_{out-nom} - V_{out})$$

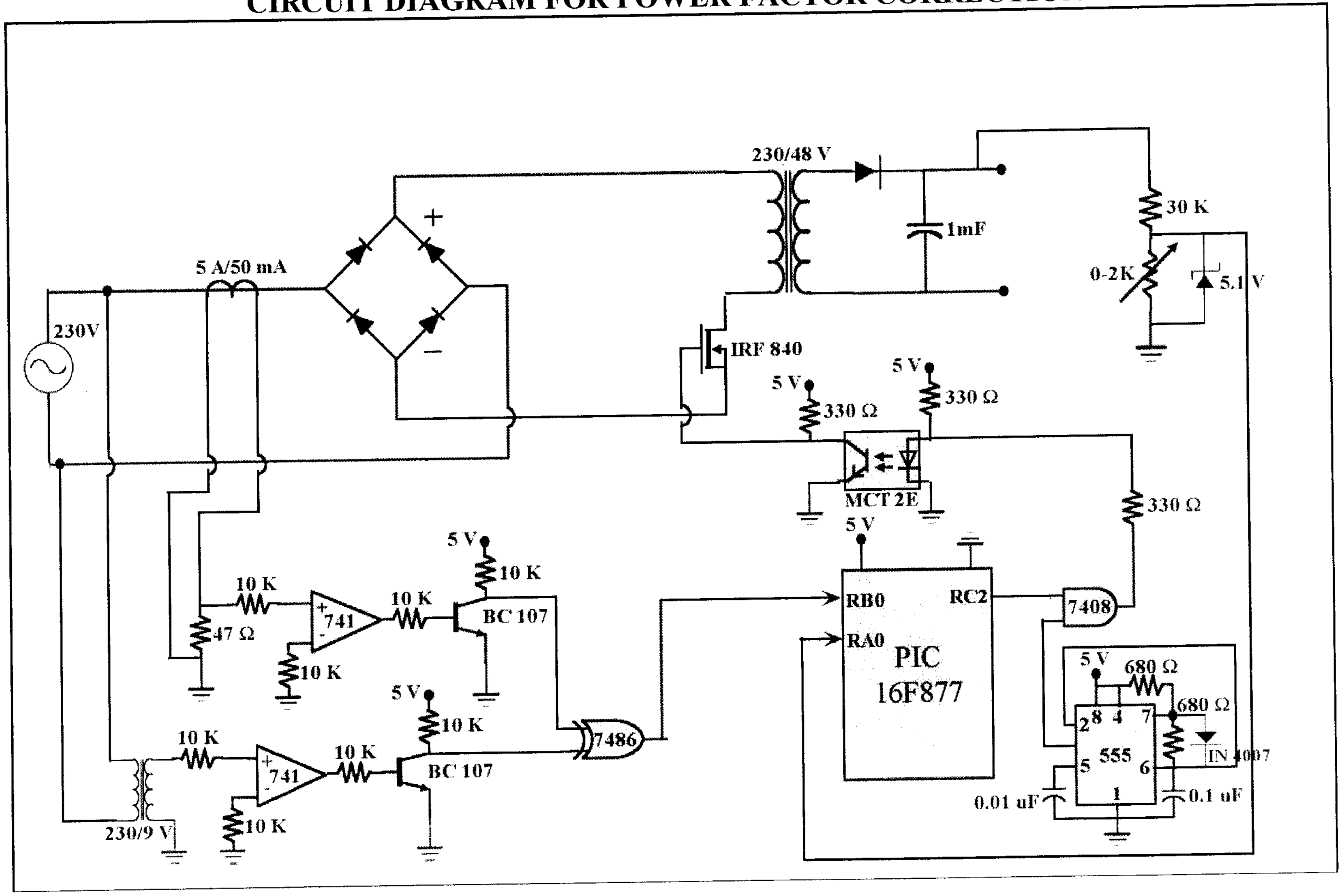
and added with the previous value of conductance which gives the new value of conductance corresponding to the error voltage. This conductance is then multiplied with the input voltage which gives the reference input current as the output.

$$I_{inp-ref} = G_{inp}.V_{inp}$$

This output is compared with the actual current and the gating pulse is given to turn the MOSFET ON, if the actual current is less than the reference current and if the actual current is more than the reference current, the MOSFET is turned OFF, so that the input current decreases and tends to follow the input voltage, thus improving the power factor.

According to the first formula, the conductance is described as a function of error voltage. So, it was anticipated that by decreasing the error voltage, the conductance can be changed and the input power factor can be improved. The circuit for testing the second method is shown below.

CIRCUIT DIAGRAM FOR POWER FACTOR CORRECTION



The microcontroller is programmed as shown in Appendix II (program 2). The circuit was designed and tested with the inputs. In this circuit, the microcontroller is programmed so as to change the duty ratio of the PWM output of the microcontroller, in order to reduce the error voltage and to maintain a constant voltage at the output.

4.2.5. TEST RESULTS:

The input power factor remained the same and there was no improvement in the power factor. It affirmed that the control system that is required to maintain high input power factors in the range of 0.95 to 0.99 must comprise of two loops namely the

- Voltage Loop
- Current Loop

which independently perform the following two actions.

- The Voltage loop maintains the output voltage at a constant value providing the required voltage transfer ratio during the half line period.
- The Current loop shapes the line current to a sinusoidal waveform.

The two loops can be combined by expressing the input current as a function of input voltage as

Input current = K (input voltage)

where the constant K has to be a function relating the input current and the error voltage.

This is explained in the previous chapters and is reiterated here for convenience.

4.3 DISCUSSIONS ON THE HARDWARE IMPLEMENTATION:

The problems faced when the Microcontroller is used instead of FPGA are

- Unlike the FPGA, the Microcontroller is not a concurrent controller and hence the operations take place one after the other leading to the delay in this operation.
- The multiplication operation and the cumulative addition operation (which simulates integration in the analog circuit), takes innumerous cycles that decrease the switching speed of MOSFET by a very large factor.
- The algorithm was made in the simplest possible way, by which a single loop performs the two processes of maintaining the output voltage constant as well as maintaining a sinusoidal line current. The algorithm could not be further simplified.
- In order to improve the power factor in this method using any sequential controller, the clock frequency has to be very high than available at present.

We can implement the power factor controller, if the problems listed above are rectified.

CONCLUSION

CONCLUSION

Since the power factor plays an important role to improve the efficiency and miniaturization of the whole system with cost effectiveness, the PWM control method was implemented in this project for current shaping in order to improve the input power factor in the range of 0.95 to 0.99. Field Programmable Gate Array controller was simulated based on PWM technique in VHSIC HDL using Xilinx 1.5 tool. Simulation results for Voltage loop, Current loop and the FPGA controller are obtained, which show the improvement in input power factor.

The hardware implementation part was carried out with PIC micro controller as the heart module. The micro controller was programmed with the same algorithm of PWM control method and tested with hardware. The PWM output of the micro controller was not matching with the switching speed of MOSFET which lead to unexpected results in shaping of current waveforms. So it is concluded that concurrency in operation is an important feature for improving the power factor by this method. The simulated codes in VHSIC HDL can be implemented in FPGA chip in which the matching between the PWM output and MOSFET switching speed will enhance the project and bring the expected results in power factor improvement.

APPENDIX I

APPENDIX I

PROGRAM FOR VOLTAGE LOOP

```
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.std_logic_unsigned.all;
use IEEE.std_logic_arith.all;

entity voltageloop is
port(vin,vnorm:in std_logic_vector(7 downto 0);
    clk:in std_logic;
    gout:inout std_logic_vector(7 downto 0));
end voltageloop;

architecture voltageloop_arch of voltageloop is
component mac is
port(Ain : in std_logic_vector(7 downto 0);
    clk : in std_logic ;
    macout : inout  std_logic_vector(11 downto 0));
end component;

signal sout,shfout:std_logic_vector(7 downto 0);
signal goutx:std_logic_vector(11 downto 0);
signal gouti:integer;

begin
sout <= vnorm - vin;
shfout <= '0' & sout(7 downto 1);
p1:mac port map (shfout,clk,goutx);
gouti <= conv_integer(goutx);
gout  <= conv_std_logic_vector(gouti,8);
end voltageloop_arch;
```

PROGRAM FOR MAC COMPONENT IN VOLTAGE LOOP

```
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.std_logic_unsigned.all;
use IEEE.std_logic_arith.all;

entity mac is
port(Ain : in std_logic_vector(7 downto 0);
    clk : in std_logic ;
    macout : inout  std_logic_vector(11 downto 0));
end mac;

architecture mac_arch of mac is
component adder8_12 is
port(A : in std_logic_vector(7 downto 0);
    B : in std_logic_vector(11 downto 0);
  SUM : out  std_logic_vector(11 downto 0));
end component ;

component reg12 is
port(datain:in std_logic_vector(11 downto 0);
    clk:in std_logic;
    dataout:out std_logic_vector(11 downto 0));
end component;

signal addout  : std_logic_vector(11 downto 0);

begin

m1 : adder8_12 port map (ain,macout,addout);
r3 : reg12 port map (addout,clk,macout);
end mac_arch;
```

PROGRAM FOR 12 BIT REGISTER COMPONENT

```
library IEEE;
use IEEE.std_logic_1164.all;

entity reg12 is
port(datain:in std_logic_vector(11 downto 0);
    clk:in std_logic;
    dataout:out std_logic_vector(11 downto 0));
end reg12;

architecture reg12_arch of reg12 is
begin
process(clk)
begin

if clk='1' and clk'event then
dataout <= datain;
end if;

end process;
end reg12_arch;
```

PROGRAM FOR CURRENT LOOP

```
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.std_logic_unsigned.all;
use IEEE.std_logic_arith.all;

entity currentloop is
port(vin,gin:inout std_logic_vector(7 downto 0);
    iin:inout std_logic_vector(11 downto 0);
    clk:in std_logic;
    cur_out:out std_logic);
end currentloop;

architecture currentloop_arch of currentloop is
component mul8c is
port(a    : inout  std_logic_vector(7 downto 0);
    b    : inout  std_logic_vector(7 downto 0);
    prod : inout std_logic_vector(15 downto 0));
end component;

component mac1 is
port(Ain : in std_logic_vector(11 downto 0);
    clk : in std_logic ;
    macout : inout  std_logic_vector(15 downto 0));
end component;

signal macout,mout:std_logic_vector(15 downto 0);

begin
p1:mul8c port map (vin,gin,mout);
p2:mac1   port map (iin,clk,macout);
process(mout,macout)
begin

if macout > mout then
  cur_out <= '1';
else
  cur_out <= '0';
end if;
end process;
end currentloop_arch;
```

PROGRAM FOR MAC1 COMPONENT IN CURRENT LOOP

```
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.std_logic_unsigned.all;
use IEEE.std_logic_arith.all;

entity mac1 is
port(Ain : in std_logic_vector(11 downto 0);
    clk : in std_logic ;
    macout : inout  std_logic_vector(15 downto 0));
end mac1;

architecture mac1_arch of mac1 is
component adder12_16 is
port(A : in std_logic_vector(11 downto 0);
    B : in std_logic_vector(15 downto 0);
    sUM : out  std_logic_vector(15 downto 0));
end component ;

component reg16 is
port(datain:in std_logic_vector(15 downto 0);
    clk:in std_logic;
    dataout:out std_logic_vector(15 downto 0));
end component;

signal addout  : std_logic_vector(15 downto 0);

begin
m1 : adder12_16 port map (ain,macout,addout);
r3 : reg16 port map (addout,clk,macout);
end mac1_arch;
```

PROGRAM FOR ADDER12_16 COMPONENT IN MAC1

```
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.std_logic_arith.all;
use IEEE.std_logic_unsigned.all;

entity adder12_16 is
port(a : in std_logic_vector(11 downto 0);
    b : in std_logic_vector(15 downto 0);
    sum : out  std_logic_vector(15 downto 0));
end adder12_16 ;


architecture adder12_16_arch of adder12_16 is
signal s,x:std_logic_vector(15 downto 0);

begin
s<="0000" & a;
x<=a + b;
sum <=x;
end adder12_16_arch;
```

PROGRAM FOR 16 BIT REGISTER

```
library IEEE;
use IEEE.std_logic_1164.all;

entity reg16 is
port(datain:in std_logic_vector(15 downto 0);
    clk:in std_logic;
    dataout:out std_logic_vector(15 downto 0));
end reg16;

architecture reg16_arch of reg16 is
begin
process(clk)

begin

if clk='1' and clk'event then
dataout <= datain;
end if;

end process;
end reg16_arch;
```

PROGRAM FOR MUL8C COMPONENT IN CURRENT LOOP

```
library IEEE;
use IEEE.std_logic_1164.all;

entity mul8c is
port(a    : inout  std_logic_vector(7 downto 0);
    b    : inout  std_logic_vector(7 downto 0);
    prod : inout std_logic_vector(15 downto 0));
end mul8c;

architecture circuits of mul8c is
  signal zero : std_logic_vector(7 downto 0);
  signal nc1  : std_logic;
  type arr8 is array(0 to 7) of std_logic_vector(7 downto 0);
  signal s    : arr8;
  signal c    : arr8;
  signal ss   : arr8;
  signal x :std_logic;

  component add8csa is
   port(b       : in  std_logic;
       a       : in  std_logic_vector(7 downto 0);
       sum_in  : in  std_logic_vector(7 downto 0);
       cin     : in  std_logic_vector(7 downto 0);
       sum_out : out std_logic_vector(7 downto 0);
       cout    : out std_logic_vector(7 downto 0));
  end component add8csa;

  component add8 is
   port(a    : in  std_logic_vector(7 downto 0);
       b    : in  std_logic_vector(7 downto 0);
       cin  : in  std_logic;
       sum  : out std_logic_vector(7 downto 0);
       cout : out std_logic);
  end component add8;

 begin
 zero <= "00000000";
 x <= '0';
  st0: add8csa port map(b(0), a, zero , zero, s(0), c(0));
  ss(0) <= '0'&s(0)(7 downto 1) ;
  prod(0) <= s(0)(0) ;
  stage: for I in 1 to 7 generate
    st: add8csa port map(b(I), a, ss(I-1) , c(I-1), s(I), c(I));
    ss(I) <= '0'&s(I)(7 downto 1) ;
    prod(I) <= s(I)(0);
  end generate stage;

  add: add8 port map(ss(7), c(7), x , prod(15 downto 8), nc1);

end architecture circuits;
```

PROGRAM FOR ADD8CSA COMPONENT IN MUL8C

```
library IEEE;
use IEEE.std_logic_1164.all;

entity add8csa is
 port(
  b      : in  std_logic;
  a      : in  std_logic_vector(7 downto 0);
  sum_in  : in  std_logic_vector(7 downto 0);
  cin     : in  std_logic_vector(7 downto 0);
  sum_out : out std_logic_vector(7 downto 0);
  cout    : out std_logic_vector(7 downto 0));
end add8csa;

architecture circuits of add8csa is
  signal zero : std_logic_vector(7 downto 0) ;
  signal aa : std_logic_vector(7 downto 0) ;

  component fadd1
   port(a    : in  std_logic;
        b    : in  std_logic;
        cin  : in  std_logic;
        s    : out std_logic;
        cout : out std_logic);
  end component fadd1;

 begin
 zero <= "00000000";
 aa <= a when b='1' else zero ;
 stage: for I in 0 to 7 generate
 sta: fadd1 port map(aa(I), sum_in(I), cin(I) , sum_out(I), cout(I));
 end generate stage;
 end architecture circuits;
```

PROGRAM FOR DELAY64 COMPONENT

```
library IEEE;
use IEEE.std_logic_1164.all;

entity delay64 is
port(din:in std_logic_vector(63 downto 0);
    clk,en:in std_logic;
   dout:out std_logic_vector(63 downto 0));
end delay64;

architecture delay64_arch of delay64 is
begin
process(din,clk,en)
begin

if clk = '1' and clk'event then
if en = '1' then
 dout <= din;
else
 dout <=
"0000000000000000000000000000000000000000000000000000000
000000000";
end if;
end if;

end process;
end delay64_arch;
```

PROGRAM FOR FADD3 COMPONENT

```
library IEEE;
use IEEE.std_logic_1164.all;

entity fadd3 is
port(a,b,c:in std_logic;
   sum,carry:out std_logic);
end fadd3;

architecture fadd_arch of fadd3 is

begin
sum <=a xor b xor c;
carry<=(a and b)or(b and c)or (c AND A);

end fadd_arch;
```

PROGRAM FOR ADD8_GEN

```
library IEEE;
use IEEE.std_logic_1164.all;

entity add8 is
port(a,b:in std_logic_vector(7 downto 0);
    cin:inout std_logic;
    sum:out std_logic_vector(7 downto 0);
    cout:out std_logic);
end add8;

architecture add8_arch of add8 is

component fadd3 is
port(a,b,c:in std_logic;
   sum,carry:out std_logic);
end component;

signal c:std_logic_vector(8 downto 0);
begin

cin <= '0';
c(0)<= cin;

g1:for i in 0 to 7 generate
a1:fadd3 port map(a(i),b(i),c(i),sum(i),c(i+1));
end generate;

cout<=c(8);
end add8_arch;
```

PROGRAM FOR ADDER8_12 COMPONENT

```
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.std_logic_arith.all;
use IEEE.std_logic_unsigned.all;

entity adder8_12 is
port(a : in std_logic_vector(7 downto 0);
    b : in std_logic_vector(11 downto 0);
    sum : out  std_logic_vector(11 downto 0));
end entity ;

architecture adder8_12_arch of adder8_12 is
signal s,x:std_logic_vector(11 downto 0);

begin
s<="0000" & a;
x<=a + b;
sum <=x;
end adder8_12_arch;
```

PROGRAM FOR FADD1 COMPONENT

```
library IEEE;
use IEEE.std_logic_1164.all;

entity fadd1 is
port(a,b,cin:in std_logic;
    s,cout:out std_logic);
end fadd1;

architecture fadd_arch of fadd1 is

begin
s <=a xor b xor cin;
cout<=(a and b)or(b and cin)or (cin and A);
end fadd_arch;
```

PROGRAM FOR SHIFTER COMPONENT

```
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.std_logic_unsigned.all;
use IEEE.std_logic_arith.all;

entity shifter is
port(s:in std_logic_vector(2 downto 0);
    datain:in std_logic_vector(7 downto 0);
    shiftout:out std_logic_vector(7 downto 0));
end shifter;

architecture shifter_arch of shifter is

begin
process(s,datain)
begin

if   s = "000" then
  shiftout <= datain;
elsif s = "001" then
  shiftout <= '0' & datain(7 downto 1);
elsif s = "010" then
  shiftout <= "00" & datain(7 downto 2);
elsif s = "011" then
  shiftout <= "000" & datain(7 downto 3);
elsif s = "100" then
  shiftout <= "0000" & datain(7 downto 4);
elsif s = "101" then
  shiftout <= "00000" & datain(7 downto 5);
elsif s = "110" then
  shiftout <= "000000" & datain(7 downto 6);
elsif s = "111" then
  shiftout <= "0000000" & datain(7);
end if;

end process;
end shifter_arch;
```

PROGRAM FOR COMBINING THE TWO LOOPS (LOOP COMPONENT)

```
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.std_logic_unsigned.all;
use IEEE.std_logic_arith.all;

entity loops is
port(vin_v,vnorm,vin_c:inout std_logic_vector(7 downto 0);
    iin:inout std_logic_vector(11 downto 0);
    clk:in std_logic;
    gin:inout std_logic_vector(7 downto 0);
    cal_on_off:out std_logic);
end loops;

architecture loops_arch of loops is

component voltageloop is
port(vin,vnorm:in std_logic_vector(7 downto 0);
    clk:in std_logic;
    gout:inout std_logic_vector(7 downto 0));
end component;

component currentloop is
port(vin,gin:inout std_logic_vector(7 downto 0);
    iin:inout std_logic_vector(11 downto 0);
    clk:in std_logic;
    cur_out:out std_logic);
end component;

begin

p1:voltageloop port map (vin_v,vnorm,clk,gin);
p2:currentloop port map (vin_c,gin,iin,clk,cal_on_off);

end loops_arch;
```

APPENDIX II

APPENDIX II

PROGRAM 1

```
#include"pic1687x.h"
#include"lcd.h"
#include"math.h"

unsigned char limit,i;
unsigned char vin,vout,gin,i_in,delg,itm,vonom,i_inref;
float pf;
static bit  rs  @((unsigned) &PORTC*8+0);
static bit  rw  @((unsigned) &PORTC*8+1);
static bit  en  @((unsigned) &PORTC*8+2);
void lcd_init( );
void lcd_disp(unsigned char);
void lcd_condis(const unsigned char*,unsigned int);
void delayu(unsigned char d)
{unsigned char i;
for(i=d;d>0;d--);
}

void main()
{
TRISB=0X01;
TRISA=0XFF;
ADCON1=0X02;
ADCON0=0X01;
OPTION=0XD7;
INTCON=0XF0;
```

```
lcd_init( );
vonom=50;
beg:
gin=0;
i_in=0;
limit=18;

for(i=1;i<=limit;i++)
    {
    delayu(200);                  //measure vout
    ADCON0 =0X01;
    delayu(200);
    ADGO = 1;
    while(ADGO);
    vout=ADRESH;
    delayu(22);

     delg=vonom-vout;
     gin=gin+delg;
                                  //measure itm
     ADCON0 =0X09;
     delayu(200);
     ADGO = 1;
     while(ADGO);
     itm=ADRESH;


     i_in=i_in+itm;
     }
```

```
                    //measure vin
ADCON0 =0X11;
delayu(200);
ADGO = 1;
while(ADGO);
vin=ADRESH;
i_inref=gin*vin;
if(i_in<i_inref)
RC7=1;
else
RC7=0;
goto beg;
}

void interrupt isr()
{
if(INTF==1)
{
INTF=O;
if(INTEDG==1)
{TMR0=0;
OPTION=0X97;
}
else
{
t1=TMR0;
OPTION=0XD7;
}
}
```

```
 lcd_condis(" pwr factor ",15);
delayu(200);
lcd_disp(pf);
}
```

PROGRAM II

```
#include"pic1687x.h"
#include"lcd.h"
#include"math.h"

unsigned int vout,vin,cin;
unsigned char t1,t2;
float pf;
static bit  rs  @((unsigned) &PORTC*8+0);
static bit  rw  @((unsigned) &PORTC*8+1);
static bit  en  @((unsigned) &PORTC*8+2);
void lcd_init( );
void lcd_disp(unsigned char);
void lcd_condis(const unsigned char*,unsigned int);
void delayu(unsigned char d)
{unsigned char i;
for(i=d;d>0;d--);
}

void main( )
{
TRISC=0X04;
TRISB=0X01;
TRISA=0XFF;
ADCON1=0X82;
ADCON0=0X01;
OPTION=0XD7;
PIE1=0X40;
INTCON=0XF0;
```

```
                //timer0 and external inrrpt are on
PR2=0X63;
CCPR1L=0X32;
T2CON=0X04;
CCP1CON=0X0F;
vout=40;
lcd_init( );
while(1)
{
ADCON0 =0X10;
delayu(200);
ADGO = 1;
while(ADGO);
vout=ADRESH;
vout=vout<<8;
vout=vout+ADRESL;
delayu(22);
ADCON0 =0X08;
delayu(200);
ADGO = 1;
while(ADGO);
vin=ADRESH;
vin=vin<<8;
vin=vin+ADRESL;
delayu(22);
ADCON0 =0X00;
delayu(200);
ADGO = 1;
while(ADGO);
cin=ADRESH;
cin=cin<<8;
cin=cin+ADRESL;
delayu(22);
pf=t1*0.2*128/(55);
pf=cos(pf*3.1419/180);
}
}

void interrupt isr()
{
if(INTF==1)
{
INTF=O;
if(INTEDG==1)
{TMR0=0;
OPTION=0X97;
}
else
{
 t1=TMR0;
OPTION=0XD7;
}
}
 lcd_condis(" pwr factor ",15);
delayu(200);
 lcd_disp(pf);
}
```

REFERENCES

REFERENCES

1. B.K.Bose 'Modern Power Electronics Evolution,Technology and Applications'
2. Bimal.K.Bose 'Modern Power Electronics and AC Drives'
3. Bhaskar 'VHDL Primer'
4. Jacob 'Power Electronics Principles and Applications'
5. Jai.P.Agrawal 'Power Electronic System Theory and Design'
6. John .B. Peatman 'Design with PIC Micro controllers', Pearson Education 2004.

WEBSITES

1. www.microchip.com
2. www.national.com