IMPLEMENTATION OF MICRO-CHIP FOR IOT APPLICATIONS

ARUMUGAM Marimuthu 60303
CHINNIAH vinodhkumar 60307
JOSEPH Jojo 60299
OKORO Oghenerume Cyril 60481
SENGUTTUVAN Dhinesh Kumar 11000
VENKATESAN Shalini- 60301

College:Institut supérieur d'électronique de Paris

Guided by : Dr.Lionel Trojman, PhD (ISEP)

***Abstract*— Implementation and simulation of SoC using a BT connection integrated for a wireless communication with a terminal and CMOS based sensors to interact with its environment. This SoC was implemented using the platform MicroWind for a CMOS technology of 130nm.**

I. INTRODUCTION

An Integrated circuit is an association or connection of various electronic devices such as resistors, capacitors and transistors etched (or) fabricated to a semiconductor material such as silicon or germanium. IC's are present everywhere in today's world from a small calculator, cellular phone to Rocket launching systems. We are designing a System on Chip using Microwind which is a very famous and much needed technology nowadays.

II. SYSTEM ON CHIP

The entire system is designed in a single chip with required electric circuits, GPU, CPU, RAM and so on. SoC includes both hardware and software with less power, better performance, more reliable and efficient. The SoC is designed instead of a system which requires more chips more electric works and connection works. The SoC fabricates all necessary circuits in one unit. Nowadays, SoC is a challenging and most expected task as it needs good designing and smart work. Many SoC's are designed using various tools and software. Our SoC is designed with Microwind following the designing rules and structures of it.

III. MICROWIND

Microwind is a software for IC designers to design beyond imagination. Microwind integrates traditionally separated front-end and back-end chip design into one flow, accelerating the design cycle and reduces design complexities. It tightly integrates mixed-signal implementation with digital implementation, circuit simulation, transistor-level extraction and verification – providing an innovative education initiative to help individuals to develop the skills needed for design positions in virtually every domain of IC industry. We started with the architecture of Very simple Microprocessor (VSM) and did step by step and finally concluded it.

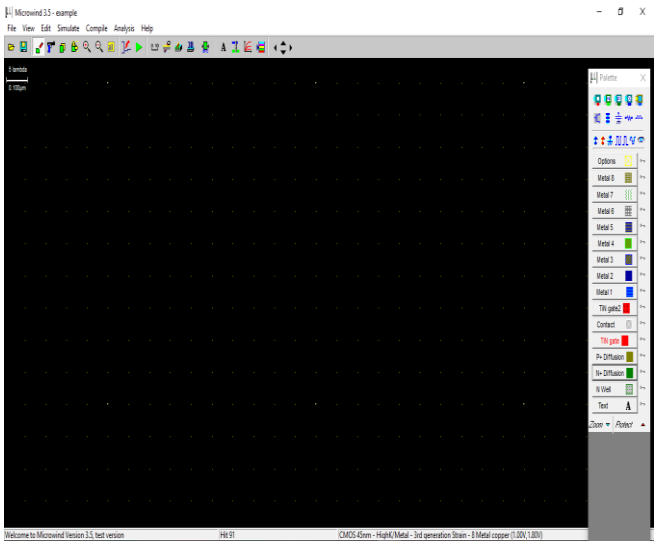


Fig 3.1: MICROWIND WINDOW

IV. VSM ARCHITECTURE

The VSM computer introduces the basic concepts of Microprocessor architecture in the simplest possible way. VSM architecture includes

1. Program counter
2. Program Memory
3. Accumulator A, Accumulator B
4. Arithmetic and Logic Unit
5. Input Register, Output Register

Each and every block has its own functions. Their functions are explained below:

- Program Counter(8bits)- Monitors address of active instruction.
- Program Memory (8 bits)- 4 most significant (Instruction) and 4 least significant (Data).
- Accumulator A, B (8+8 bits)- Store the operands for an Arithmetic operation.
- ALU (8 Bits)-Arithmetic operations.
- Input and Output Register (8+8 bits)-Involved in transferring data.

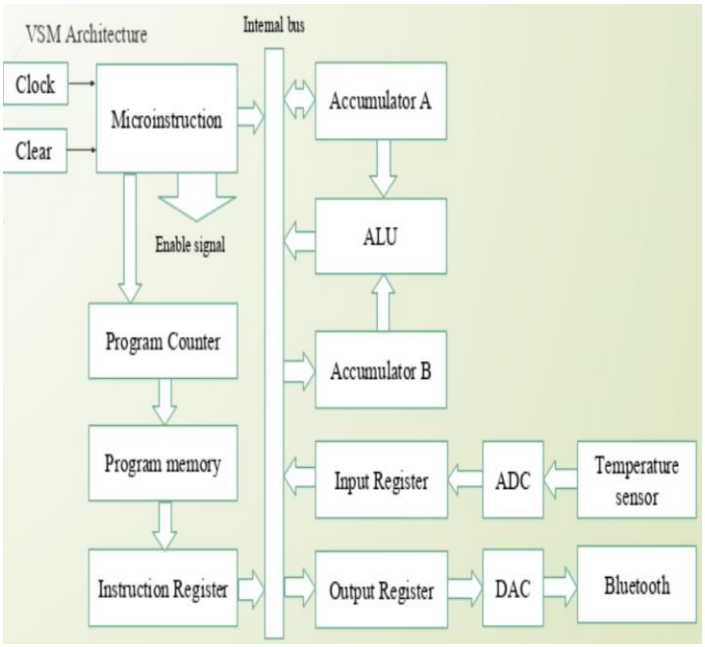


Fig 4.1. ARCHITECTURE OF THE CHIP

V. ENABLE SIGNALS

VSM circuit is a based on a bus. Each block may take control of bus using a specific signal f "Enable". for example, Accumulator A has a enable signal Enable A. When Enable signal is high, the four bits of accumulator A go to internal bus. The list of Enable signals are listed below. The control of these signals is provided by the microinstruction block, which plays a fundamental role in the control microprocessor.

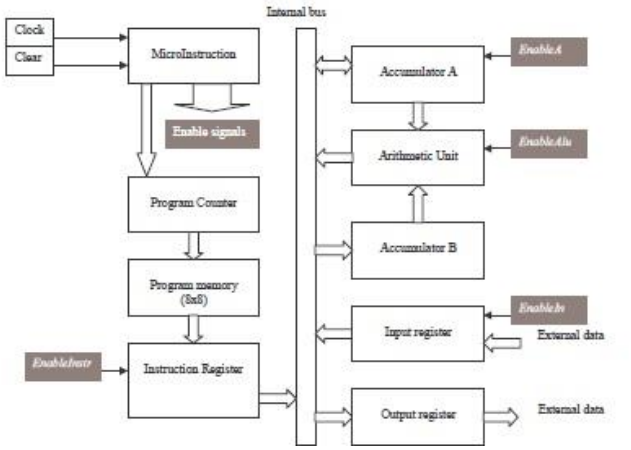


Fig 3: The controller generates enable signals that allow one block to take control of the bus.

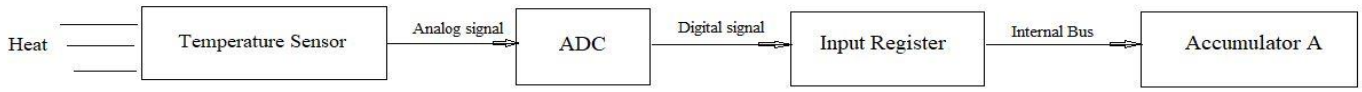
List of Enable Signals

Enable Signal	Description
EnableA	Authorizes A to take control of the bus.
EnableAlu	Places the result of the arithmetic operation (ADD or SUB) on the bus
EnableInstr	Places the data part of the instruction (Four least significant bits) on the bus.
EnableIn	Transfers the contents of the external input on the internal bus.

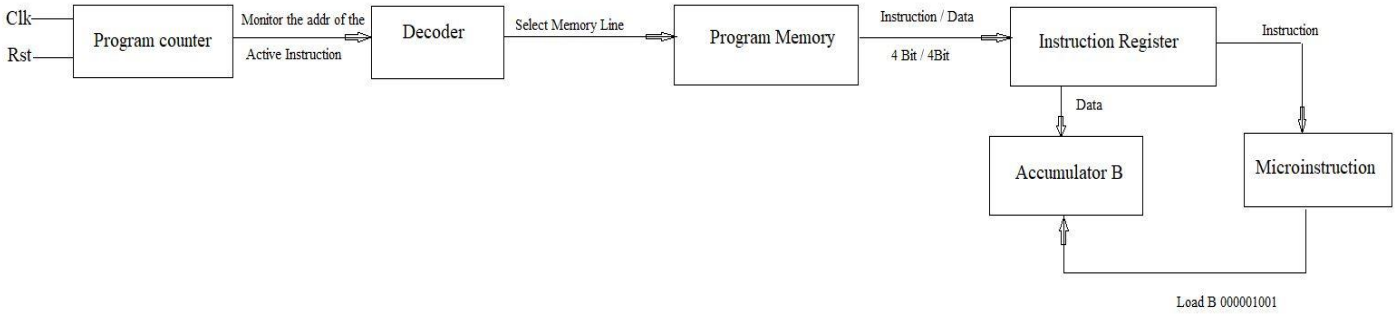
Fig 4: Four blocks may take to control of internal bus.

VI. FLOOR PLAN

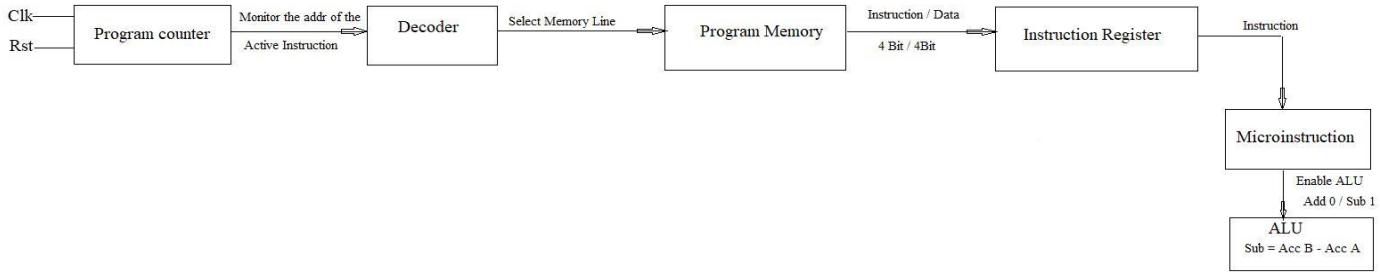
Step 1 : Get sensor value



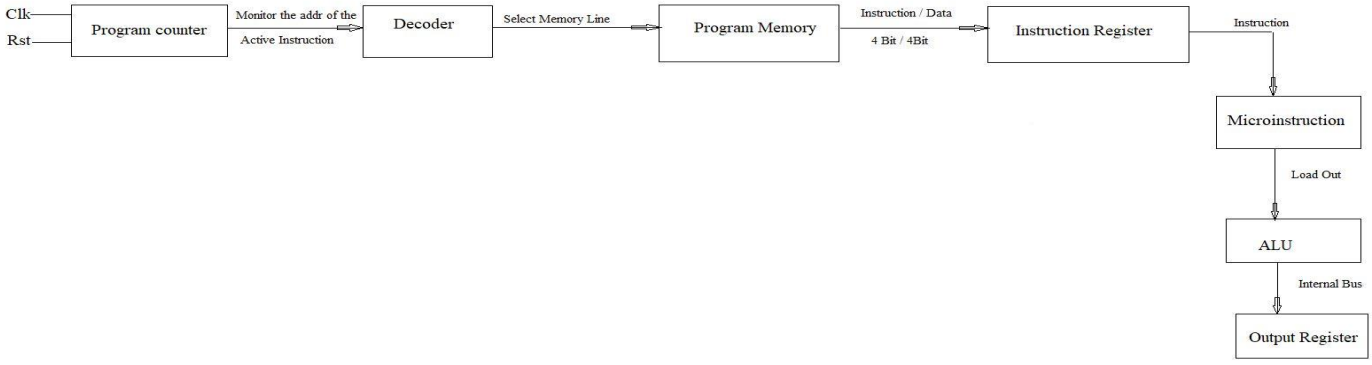
Step 2 : Load Default Value



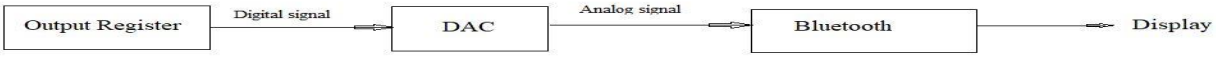
Step 3 : Subtract Two Values



Step 4 : Get output

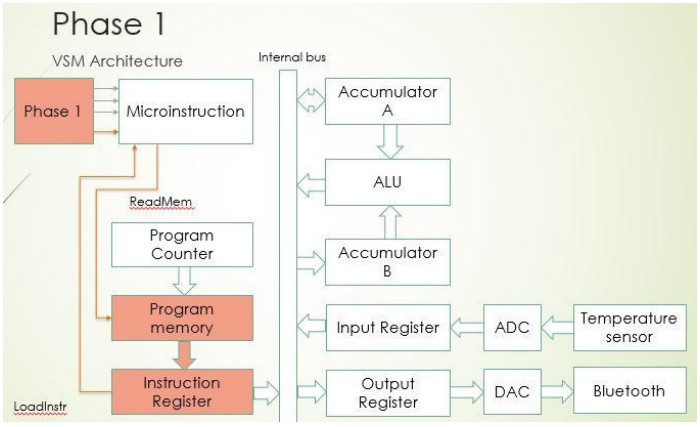


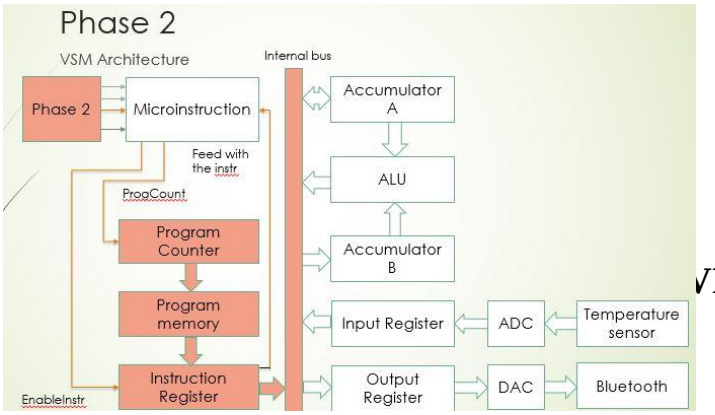
Step 5 : Display

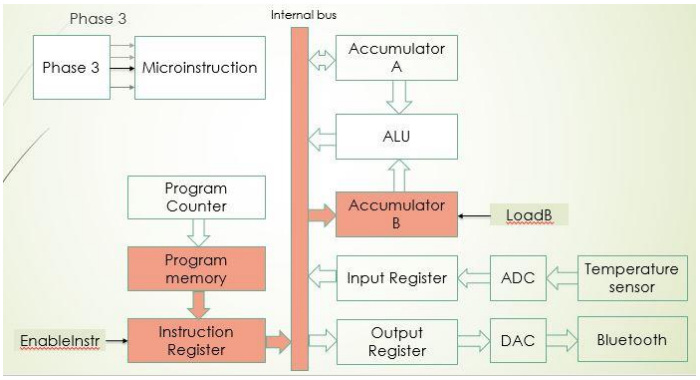


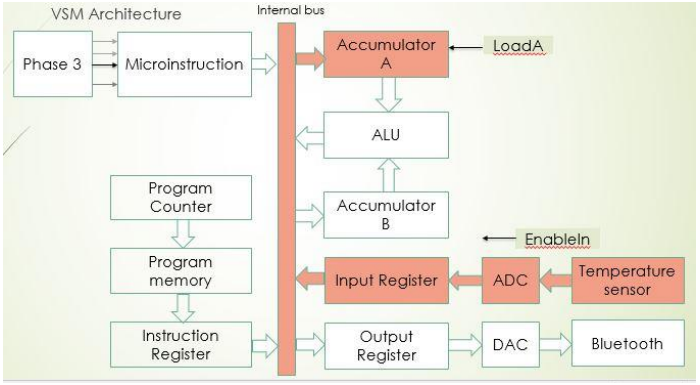
❖ PHASE

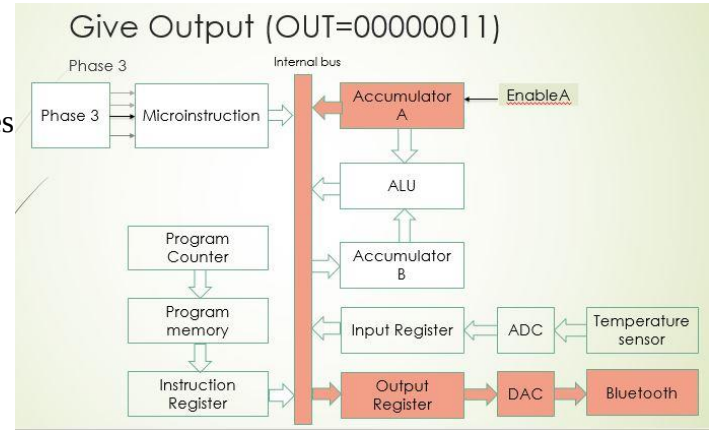
Execution of one instruction is based on four times phases

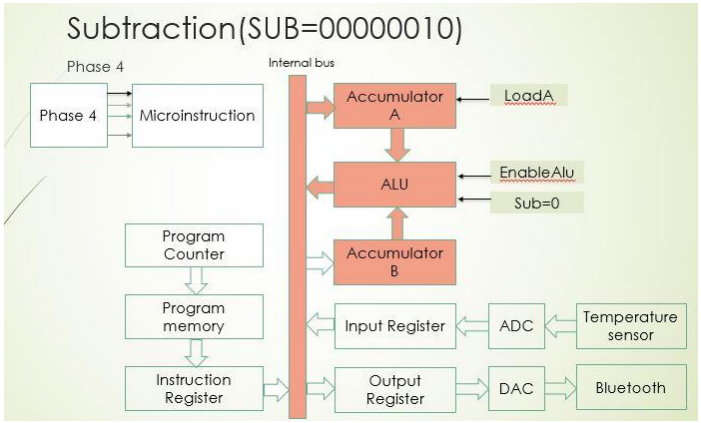










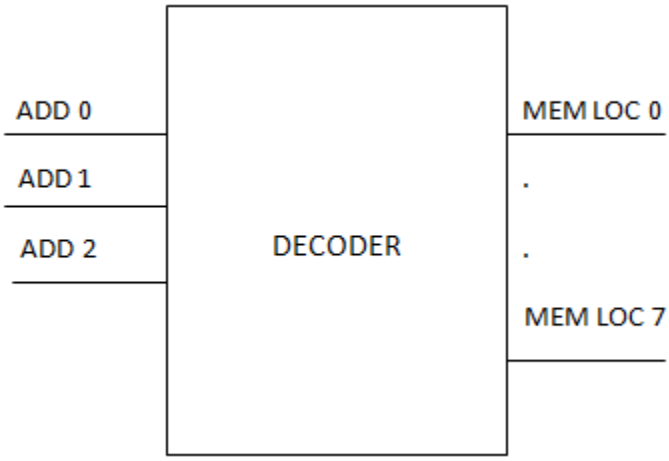


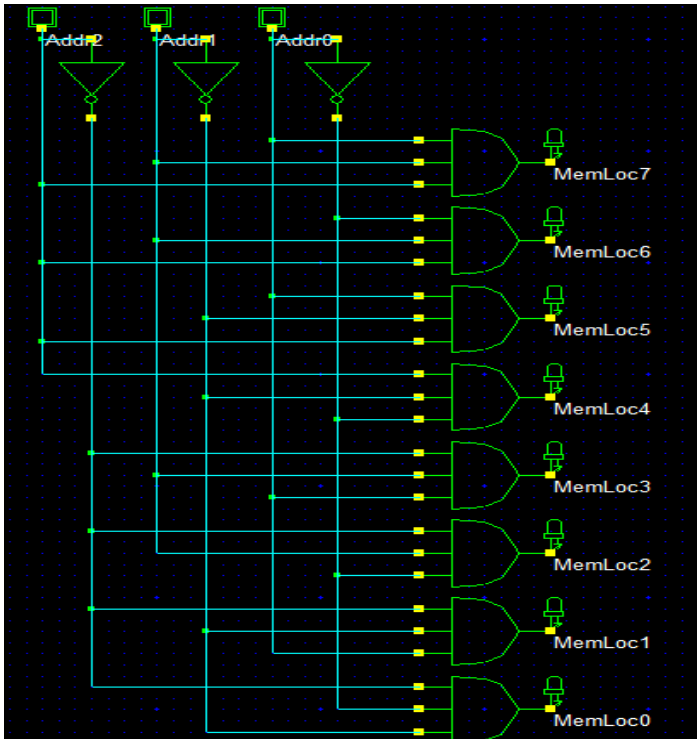
VII. INSTRUCTIONS

The basic instructions are listed below. As the instruction is 8 bits, we can use till 256 instructions.
No operation (NOP-00000000)
Addition (00000001)
Subtraction (00000010)
Get input (00000011)
Give output (00000100)
Load instruction (00000101)
We can use remaining instruction for the future enhancement of the chip by adding multiple sensors and different operations.

❖ DECODER

Decoder is used to specify the memory address for storing the instructions. Here we are using 3 to 8 decoder that gives 8 logic outputs for 3 input and has an enable pin. It is also known as Binary to an octal decoder. The decoder circuit works whenever the enable pin is high. The address and the instruction is stored in the program memory. The processor gives instruction during execution. So, with the help of addresses that are given by the program counter, the instruction is identified in the program memory during processor execution.





VIII. PROGRAM MEMORY

The program memory contains up to 16 bytes, where we store the instructions to be executed and the value to be compared with the sensor data. Both instruction and value are 8-bits long. As displayed in Fig. 8.1 is an example for an 8-bit instruction. Each instruction is split into two parts: the four most significant bits represent the instruction code, while the four least significant bits represent the data. As shown in Table 8.1, the program loads accumulator A with the value '2', then adds '1', and places the result in the output register.

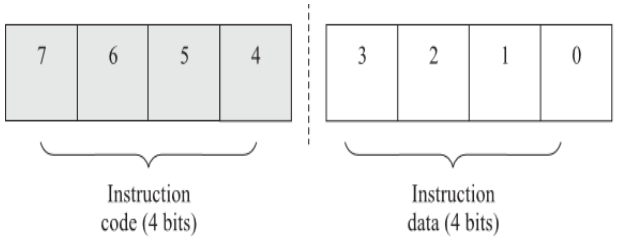


Fig 8.1 Each instruction is split into four-bit microinstruction code and four-bit data fields

Mnemonic	OpCode (binary)	OpCode (hexa)
LDA 2	0101 | 0010	0 x 52
ADD 1	0001 | 0001	0 x 11
OUT	0011 | 0000	0 x 30
NOP	0000 | 0000	0 x 00

Table 8.1 A simple program for adding two four-bit numbers

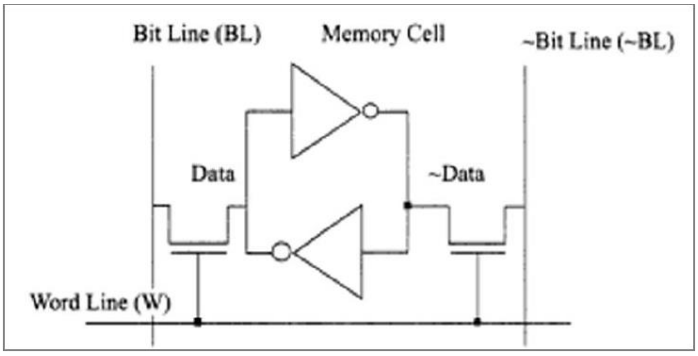


Fig 8.2 A single memory cell

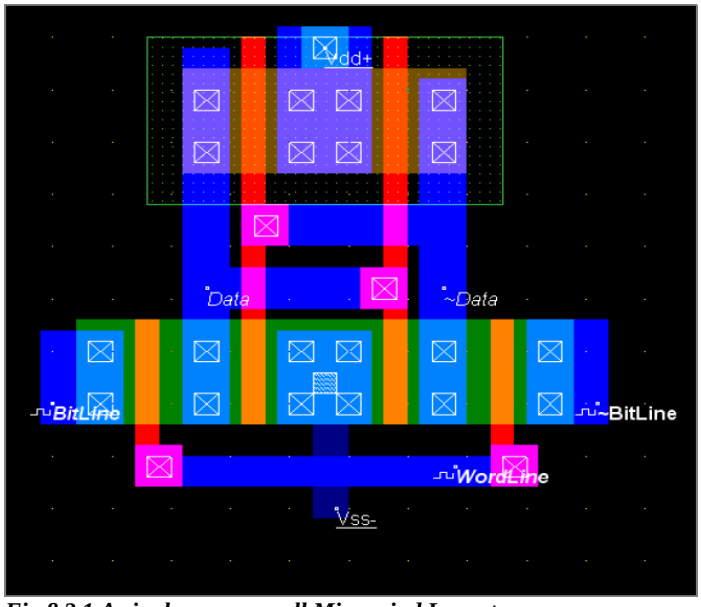


Fig 8.2.1 A single memory cell Microwind Layout

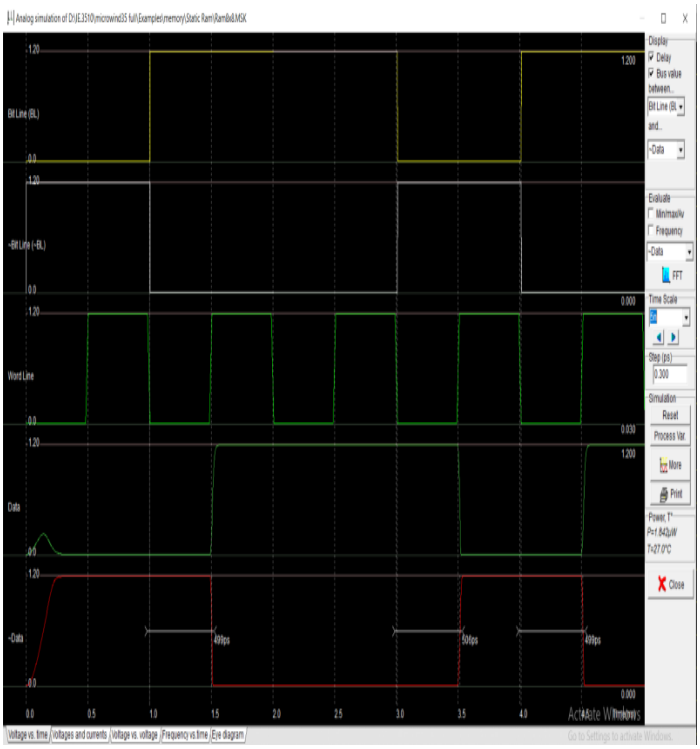


Fig 8.3 A single memory cell Microwind Simulation

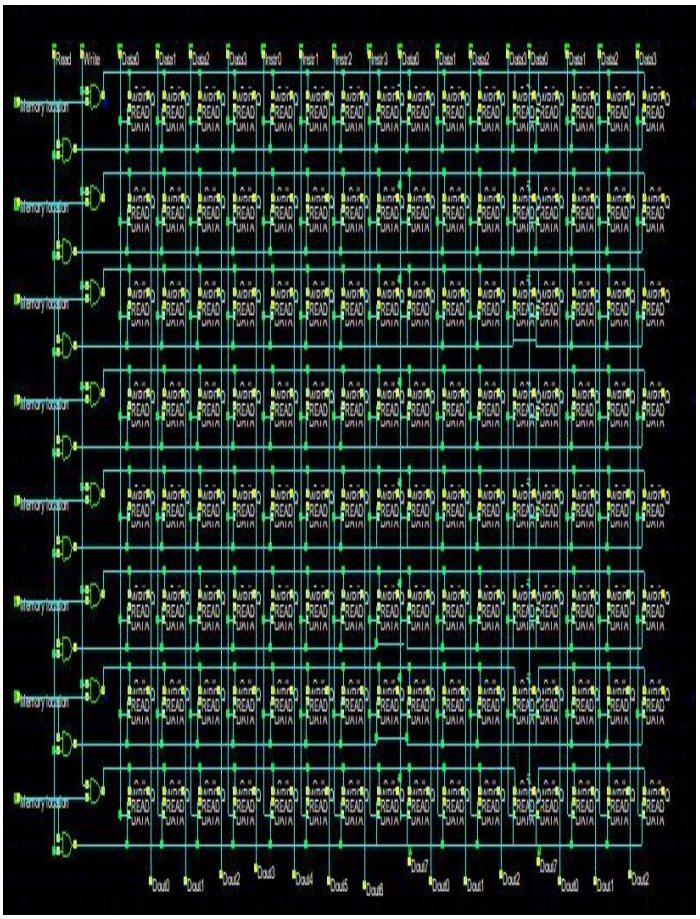


Fig 8.4 8*8 memory Layout

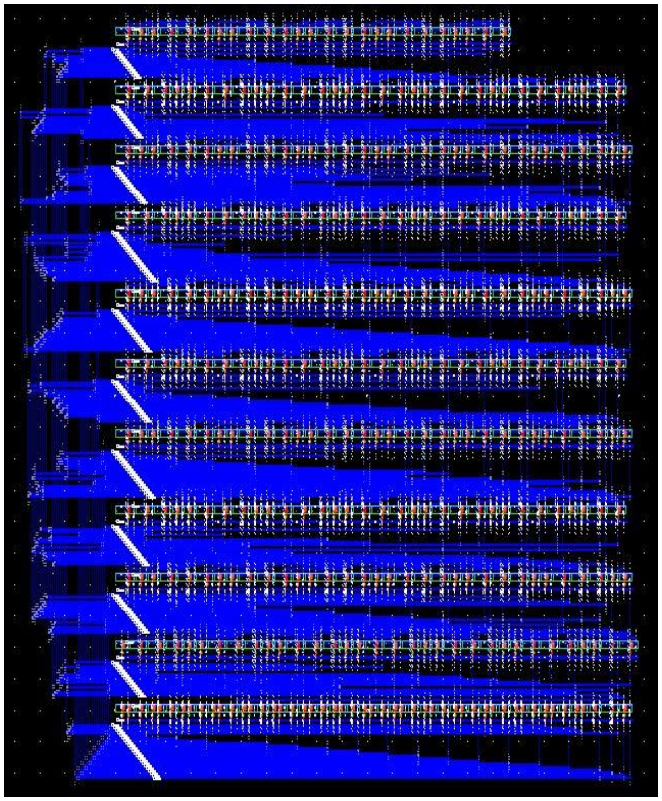


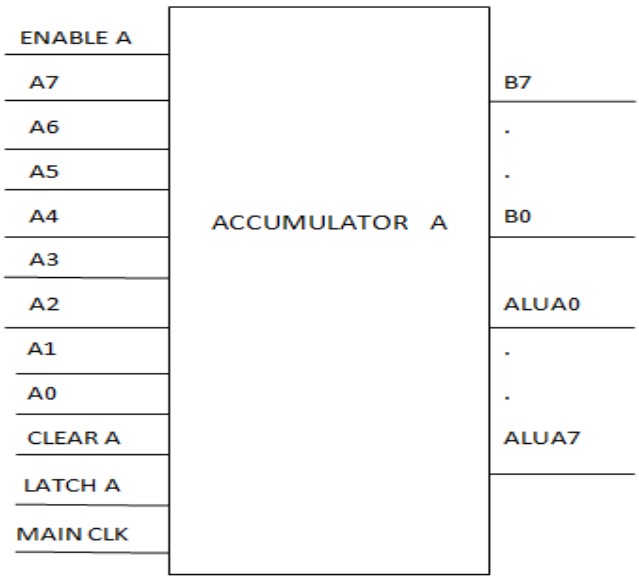
Fig 8.5 8*8 memory Microwind Layout

The above fig 7.5 is 8*8 memory. We are using two 8*8 memory so we can send the 8-bit stored data to Accumulator B to compare it with Sensor data.

IX. ACCUMULATOR A

The Accumulator is an 8 Bit register that is a part of Arithmetic and logic unit (ALU). This register is used to store 8-bit data and to perform Arithmetic and logical operations. The result of the ALU is stored in the Accumulator. The size of the Accumulator in terms of bits is used as a measure of the data unit capability of the Microprocessor. The Accumulator A stores the intermediate results computed by the microprocessor. When Enable A signal is asserted, it stores the accumulator results on the internal bus. Enable A authorizes A to take control of the bus. The signal of Latch A authorizes the transfer of Input data to Accumulator A at the falling edge of main clock.

The accumulator A is designed using 8 D-registers. Commonly registers are used to stores the data that uses edge-triggered latches. It transfers input data to the Q on clock rising or falling edge. The below shows the schematic diagram of Accumulator A.



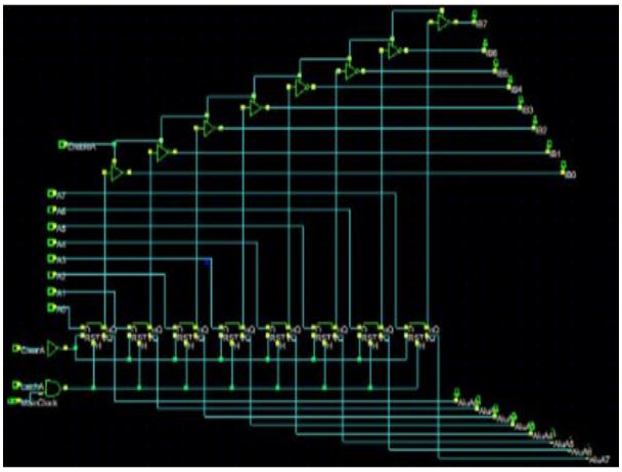


Fig 9.1 Accumulator A layout

Addition or subtraction operation is performed in the Accumulator A with the help of 8 bit data. Likewise, same happens in Accumulator B, then the Arithmetic and Logic operation is done using ALU. After this operation is finished on the rise edge of the clock, the result is stored in the Accumulator A. This process is done during Phase-3.

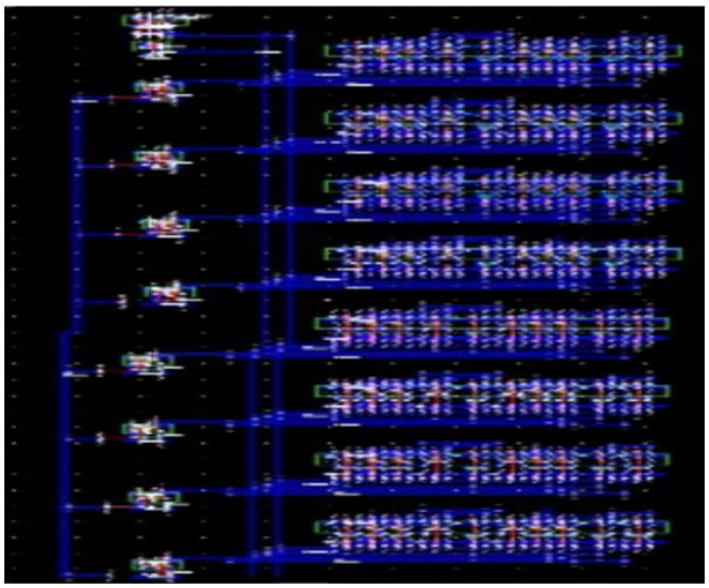


Fig 9.2 ACCUMULATOR A DESIGN USING MICROWIND

Initially, the clear button is set as 0 to 1. After reverting it changes from 1 to 0. The input of synchronised clock is controlled by latch A. The 8 NAND gates are connected through which the main clock exports. By following the design rules of microwind the Accumulator A is designed finally.

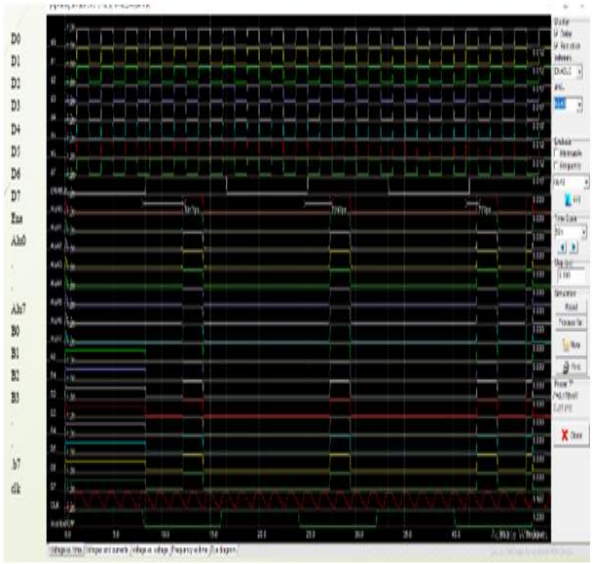


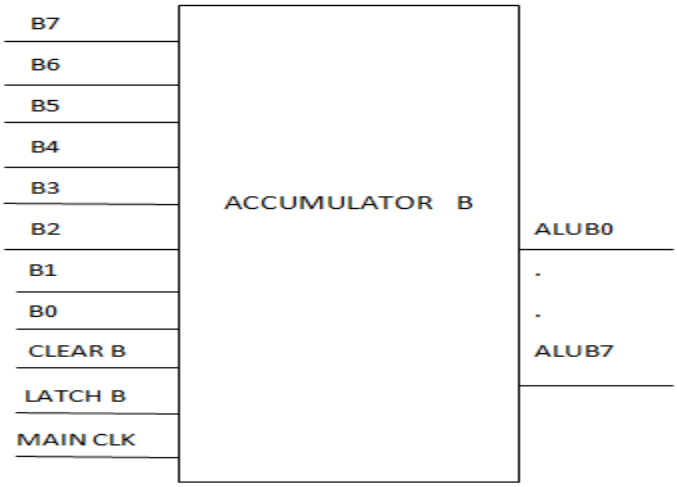
Fig 9.3 ACCUMULATOR A SIMULATION

X. ACCUMULATOR B

Accumulator B is similar to Accumulator A which has 8 D-registers. Here the Latch B signal authorises the transfer of Input data to the D-register at the fall edge of main clock. The main work of Accumulator B is it serve the operands that is needed for addition

and subtraction process. So, it adds/subtracts with Accumulator A and the result is done in ALU.

It is also designed using 8 D-registers. The register output is available from AluB0...AluB3 for add and sub operations



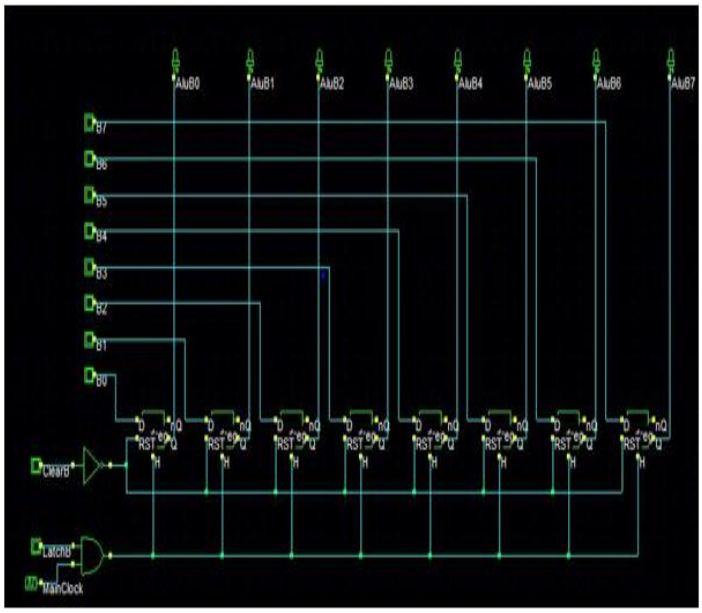


Fig 10.1 Accumulator B layout

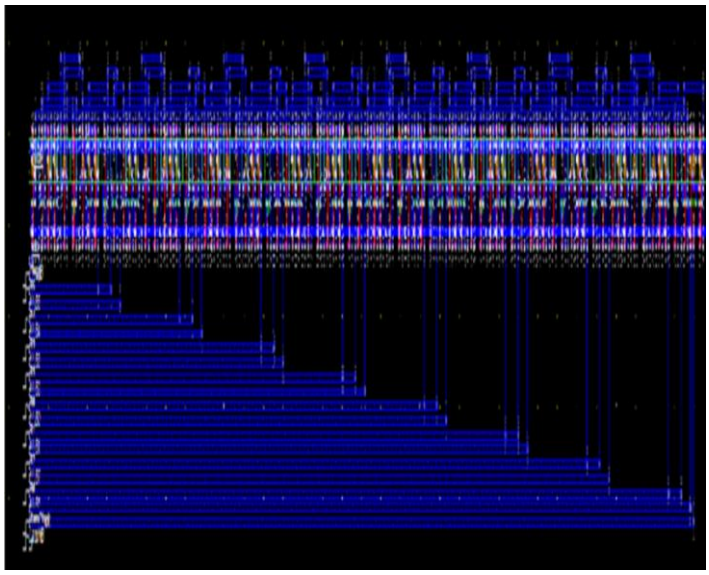


Fig 10.2 ACCUMULATOR A DESIGN USING MICROWIND

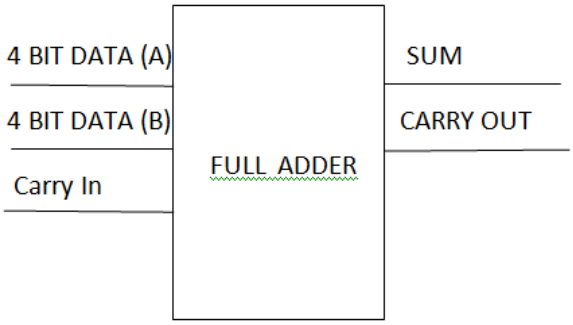
The operation is done between D-registers and ALU. The simulation result is shown below.

Fig 10.3 ACCUMULATOR A SIMULATION

XI. ARITHMETIC AND LOGIC UNIT

The addition and subtraction process is done in this part with the help of Accumulator A and B. This operation is done with 8 cascaded Full adders with the combination of AND, OR gates. Initially, the reset is set to 0 and the inputs A and B also set to 0. The value of A is stored and Latch A is pressed to complete the cycle inorder to store the clock in the Accumulator A. Same happens again but Value B is stored and Latch B is pressed to complete the cycle. So, that the clock is stored in the Accumulator B. So, after combining the results of Accumulator A and B, the ALU does the arithmetic operations (ADD/SUB).

There are two keyboards A and B. So, the operands from Accumulator A is simulated by keyboard A and the operands from Accumulator B is simulated by Keyboard B. To perform ADD operation, the carry signal propagates from low to upper stage. To perform SUB operation 1's complement of A is used if we have B-A condition. Here we are using subtractor. B is subtracted from A and the difference between them will appear in the carry-out. We are using three conditions for the subtractor operation. They are A less than B, A greater than B and A equal to B. When the input satisfies one of the above conditions then the LED blinks accordingly.



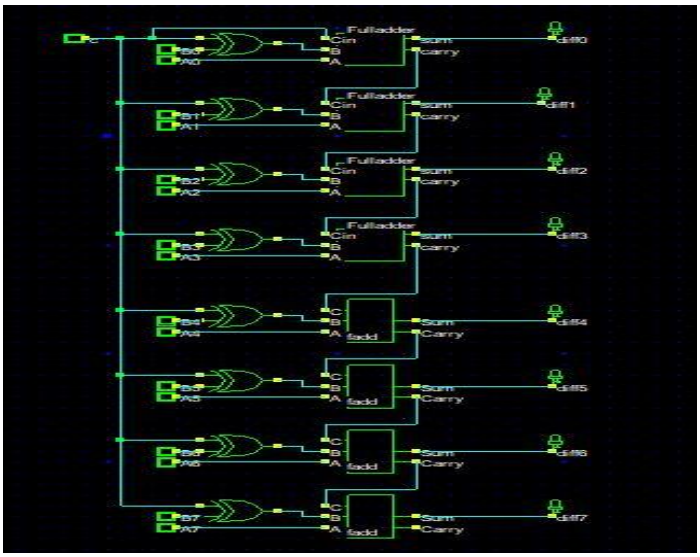
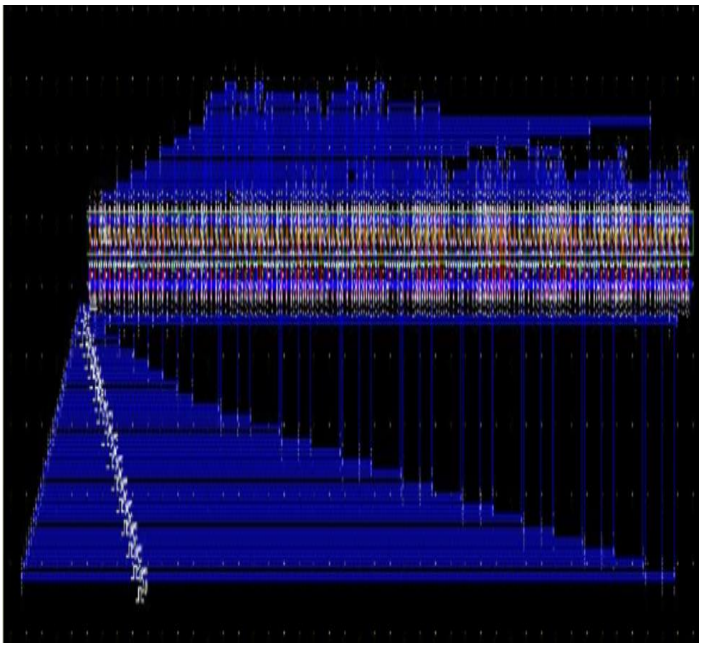


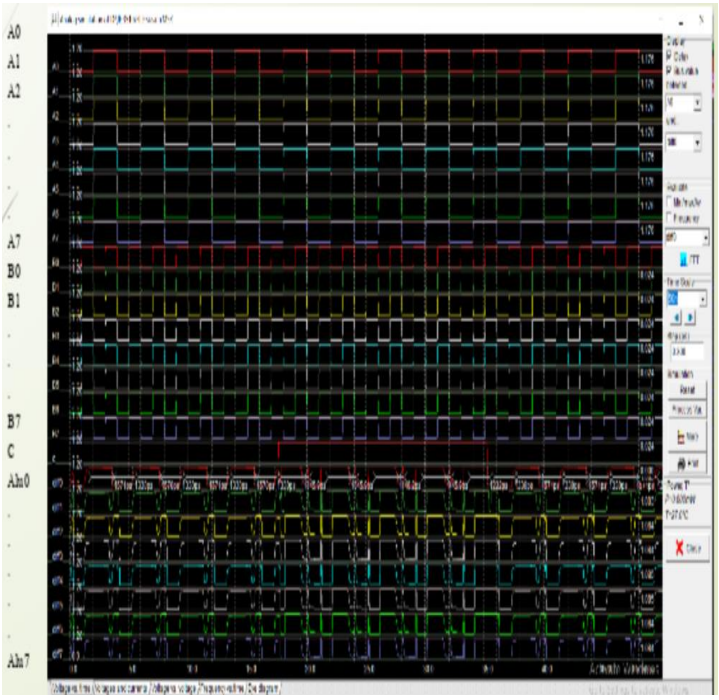
Fig11.1 ALU layout

ALU DESIGN USING MICROWIND



When addsub=1, the circuit finishes its subtraction operation. When add/sub=0, the circuit finishes its Addition operation

SIMULATION RESULT



XII. INPUT REGISTER

The input register consists of a 3-state buffer. The output will be same as data when the Enable is 1 and if Enable is 0 then the output will be 0. Fig 12.2 is the logical block of the 3-state buffer. We are adding capacitor in the output Fig 12.3 to smooth the output signal.

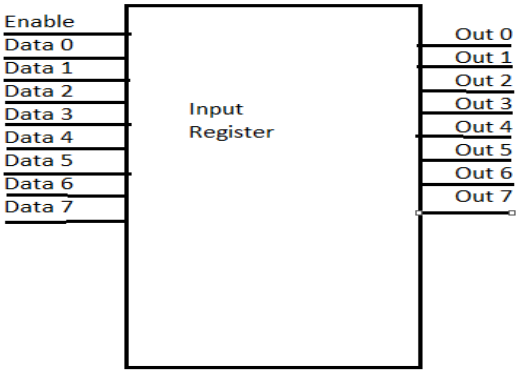


Fig 12.1 8-bit Input Register overall block

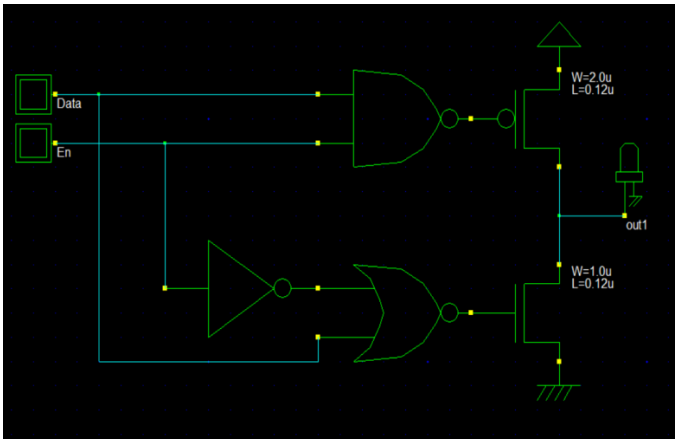


Fig 12.2 3-state buffer Layout

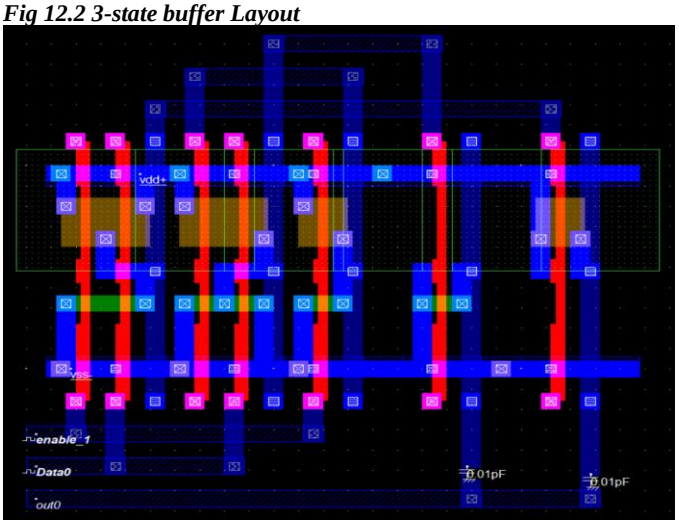


Fig 12.3 3-state buffer Microwind Layout

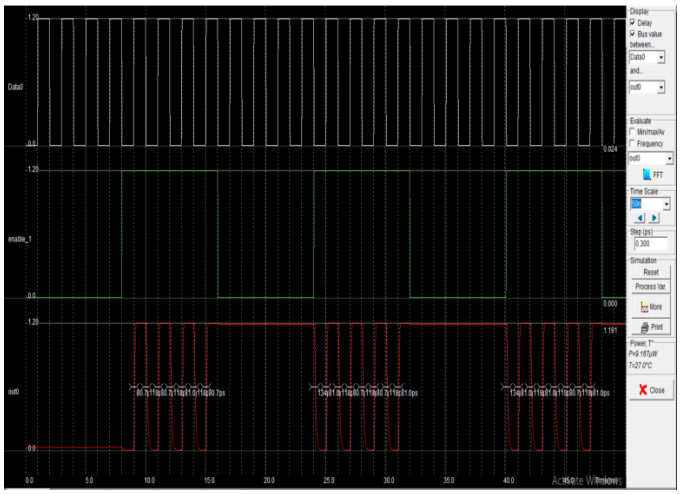


Fig 12.4. 3-state buffer Simulation

Here in 8-bit input register consist of eight 3-state buffers with common enable signal and the input is received from 8-bit ADC. The Enable signal is received from Microinstruction output Enable-input and the output of the input register is store in ALU for subtraction.

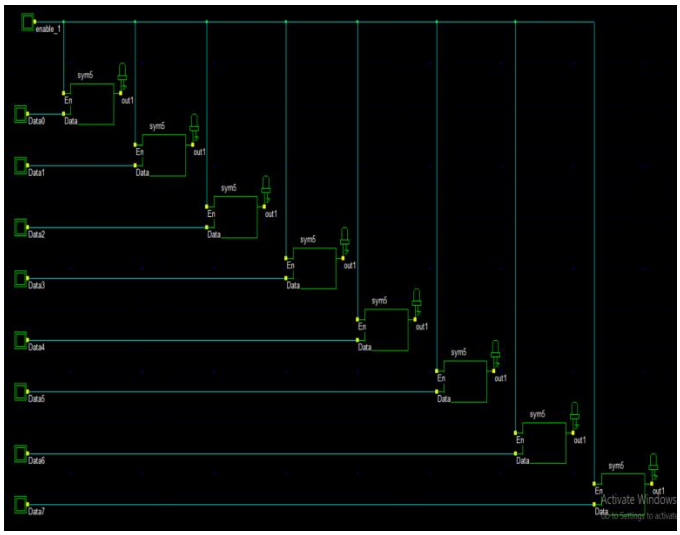


Fig 12.5 8-bit Input Register block

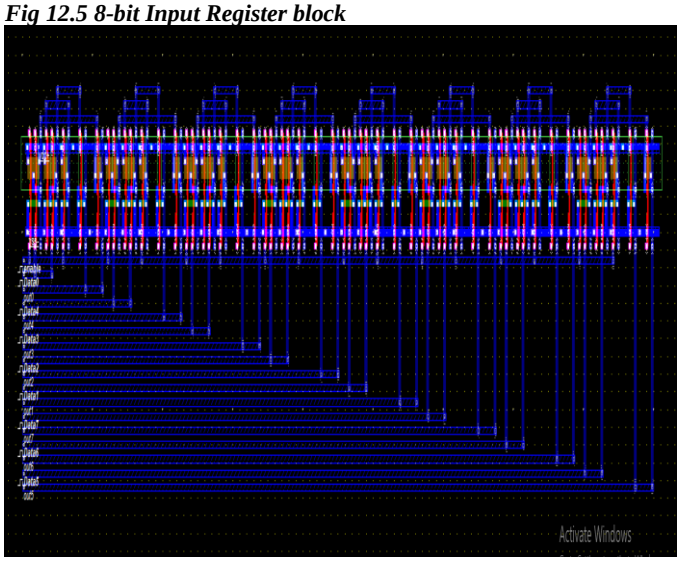


Fig 12.6 8-bit Input Register Microwind Layout

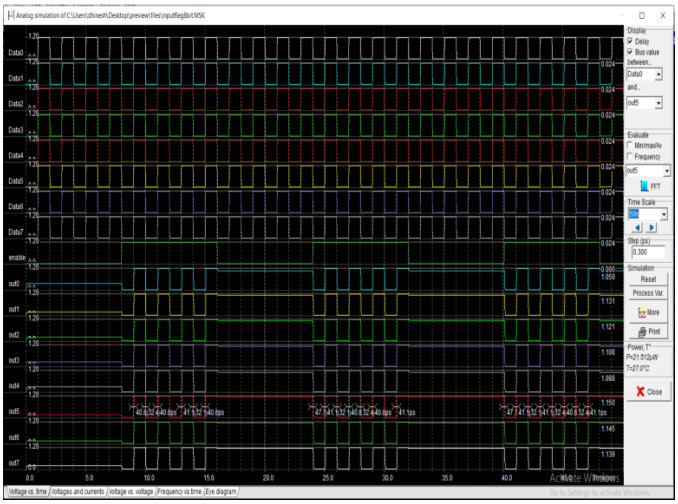


Fig 12.7 8-bit Input Register Simulation

XIII. OUTPUT REGISTER

Here Output register consist of 8 D-register as shown below Fig 13.2. We are storing the data in the D-register during the positive edge of the clock (i.e. rising edge). The purpose of storing the data in D-register during Positive edge and not in negative edge because to avoid Synchronization conflict, so we are adding a NAND gate to make the circuit sensitive to the rise edge of the Main clock in Fig 13.2.

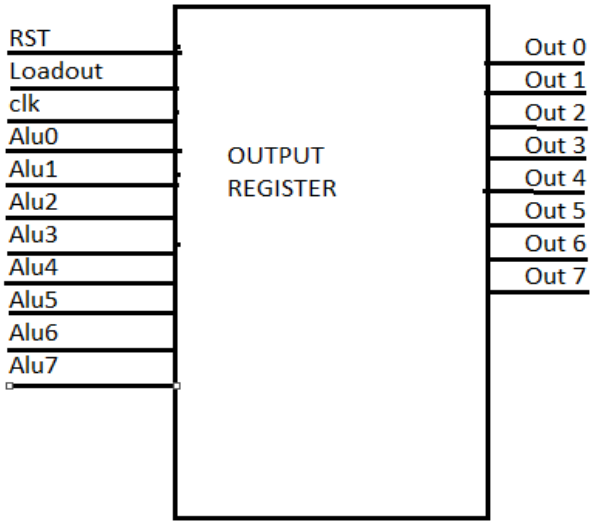


Fig 13.1 Output Register overall block

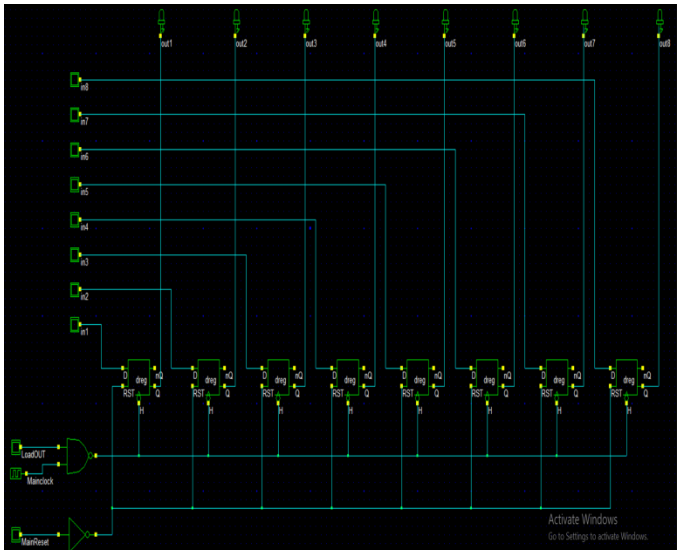


Fig 13.2 Output Register block diagram

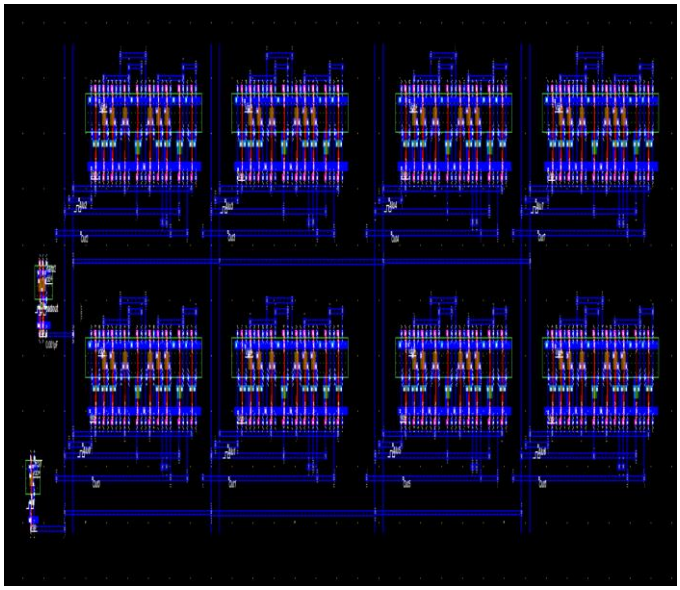


Fig 13.3 Output Register Microwind Layout

The ALU's Output is given as Input to the Output register. The 8-bit Alu output is stored in each D-register on Output register during the positive edge of the clock and output of the output register is send to DAC and then transmitted the converted data using Bluetooth

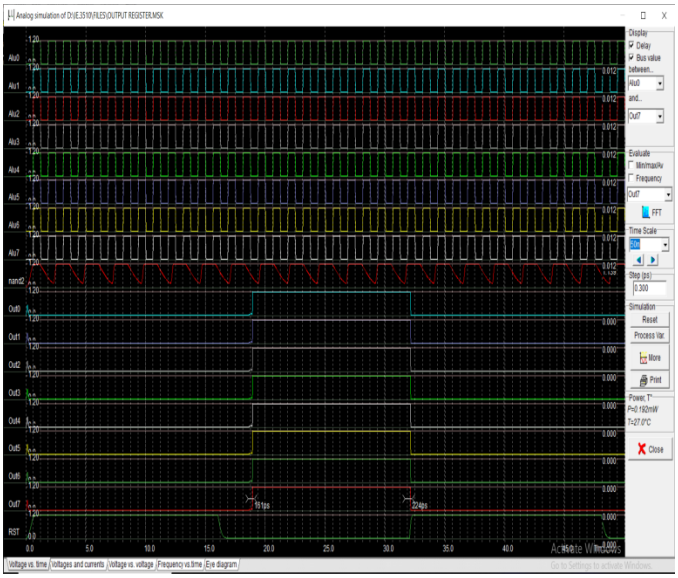
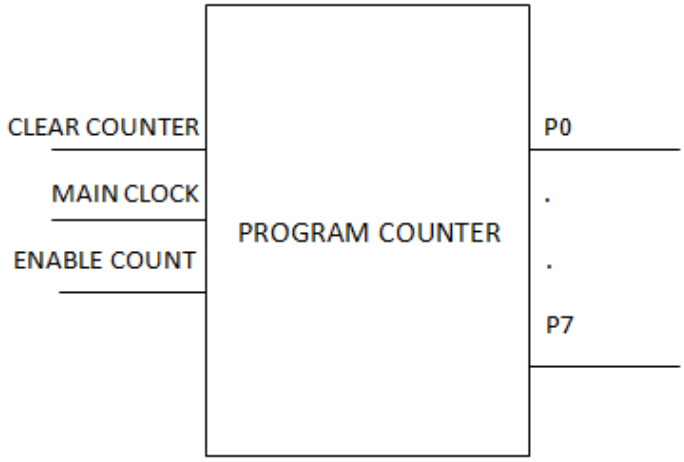


Fig 13.4 Output Register Simulation

XIV. PROGRAM COUNTER

The program counter is a circuit that works closely with the memory. While the decoder reads the instructions from the memory and gives the enabling pulse to the other circuits, for example the ALU, inputs and outputs and RF circuit, the counter will work as a primitive clock for the entire chip.



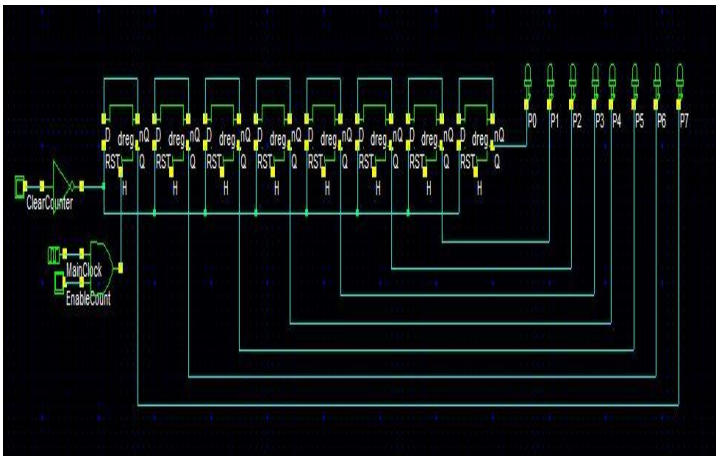


Fig 14.1 Schematic diagram of the program counter

The program counter plays a very important role in the microprocessor supplying the main program memory with the address of the active instruction. At the start, the program counter is set to 00000000 so that microprocessor starts by the first instruction. At the end of each instruction, the program counter is incremented in order to select the next instruction. It counts from 00000000 to 11111111.

The 8-bit counter performs the clock of the entire chip. The memory will give the instruction to the decoder on every positive edge of the counter. After receiving the instruction, the decoder will send pulse signals to enable every other circuit inside the chip.

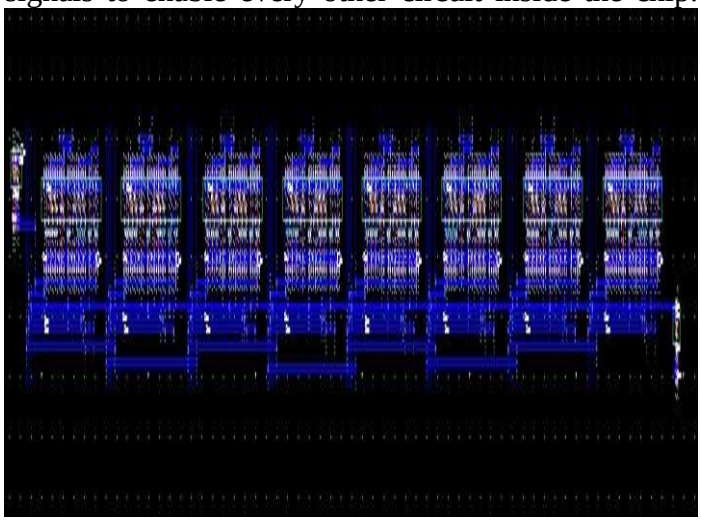


Fig 14.2 Layout of the program counter

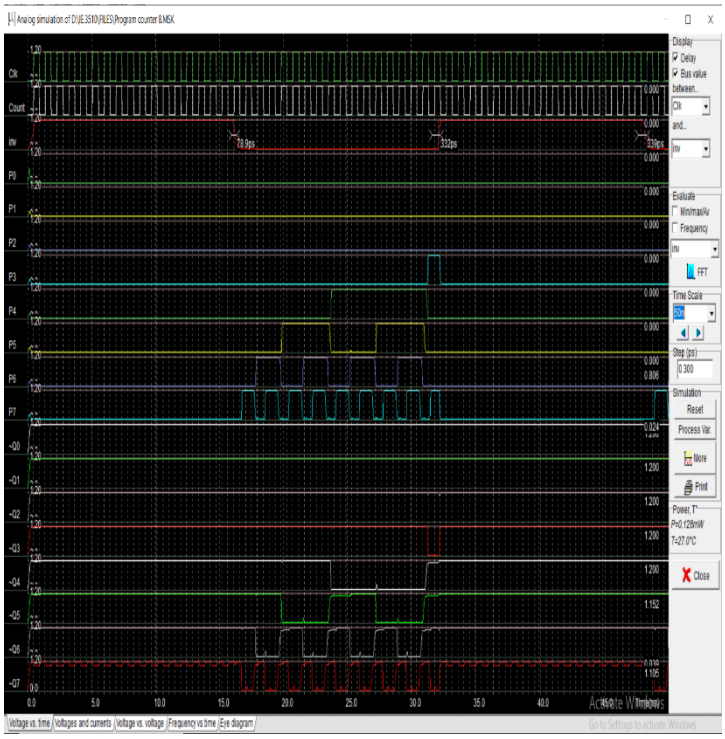


FIG 14.3 Simulation result

The design is based on Chain edge-sensitive D-registers. The circuit is very simple and works Asynchronously.

XV. INSTRUCTION REGISTER

The instruction register stores the instruction being executed. The eight-bit information is split into two parts: the most significant bits correspond to the instruction code, while the least significant bits are the data. The instruction code is stored in the four D-registers situated at the bottom of Fig. 14.1, in order to be available for the microinstruction decoder. The data is stored in four separate D-register cell and can be made available on the internal bus. The instruction register keeps a copy of the current instruction and releases the main memory, which can be accessed later for both read or write operation.

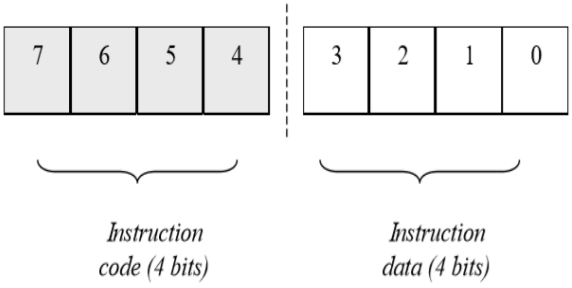


FIG 15.1 Instruction register stores contents of the memory and separates code part (lower registers) from data part (upper registers).

FIG 15.2 Layout of instruction register

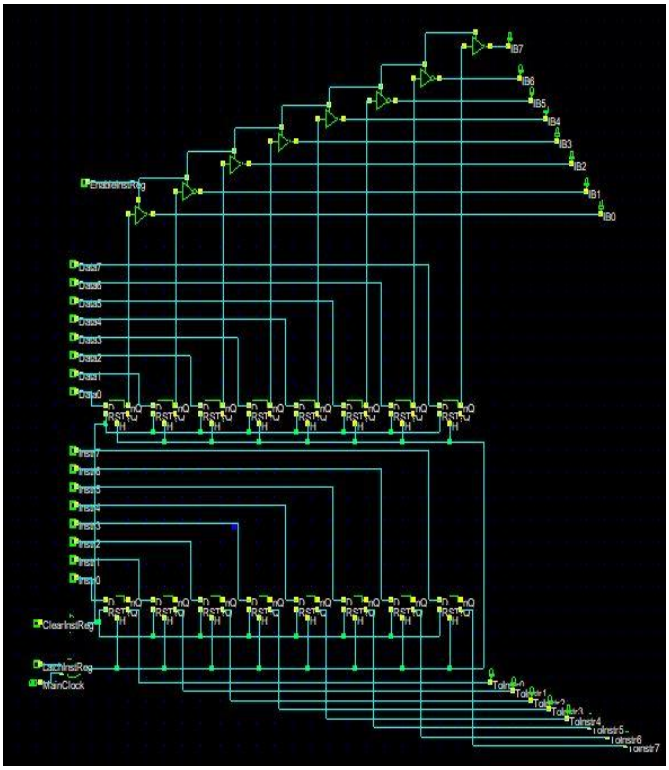


FIG 15.2 Layout of instruction register

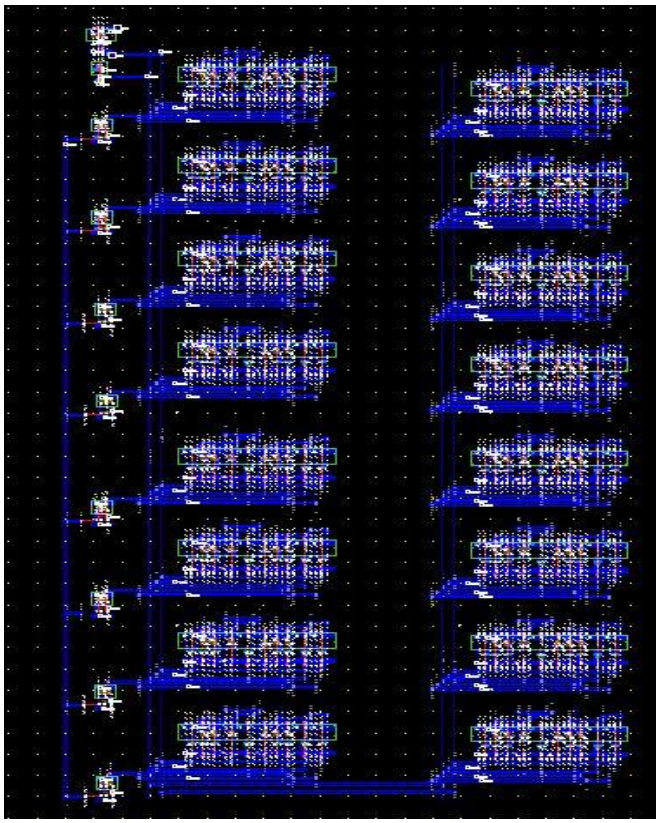


FIG 15.3 Schematic diagram of instruction register

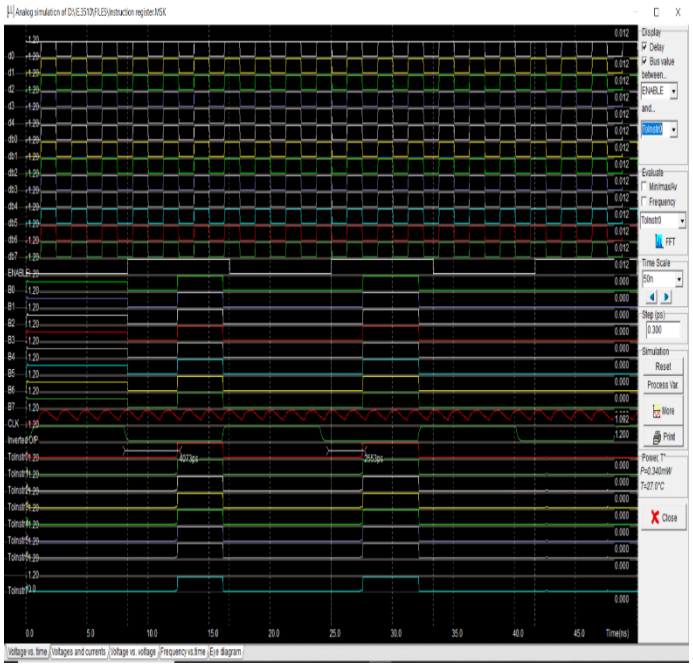


FIG 15.4 Simulation result of instruction register

XVI. MICROINSTRUCTION

The Microinstruction is the heart of the computer as it controls most of the Enable, Latch, ALU controls. The input for the microinstruction is received from the instruction register which is of 8 bit and from phase generator (ring counter). Here the inputs are connected to not gate to get both 0 and 1 based on the input at same time. Let's take a 4bit instruction for the below example (fig 15.2) is 0000 then it turns on the upper AND date which act as an instruction decoder. Output of the phase generator are phase (0-3) and Phase 0 and Phase 1 are not connected to the instruction decoder as both the phase 0 and phase 1 are not dependent to instruction register input.

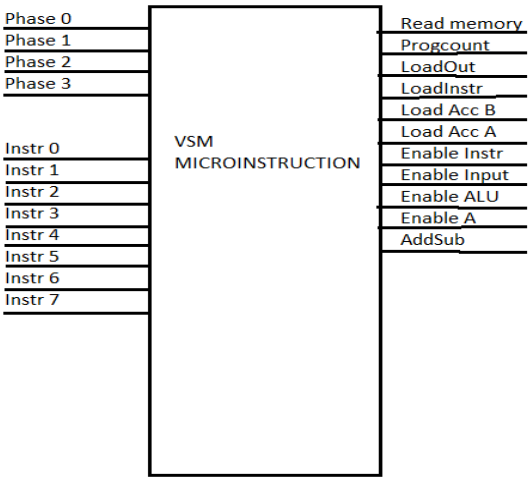


Fig 16.1 Microinstruction overall Block

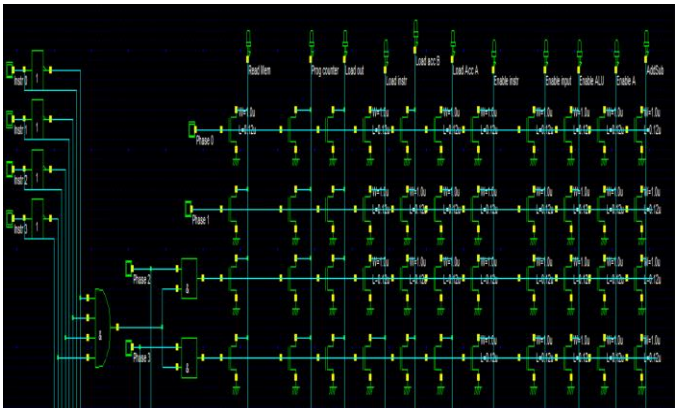


Fig 16.2 4 Bit Microinstruction layout

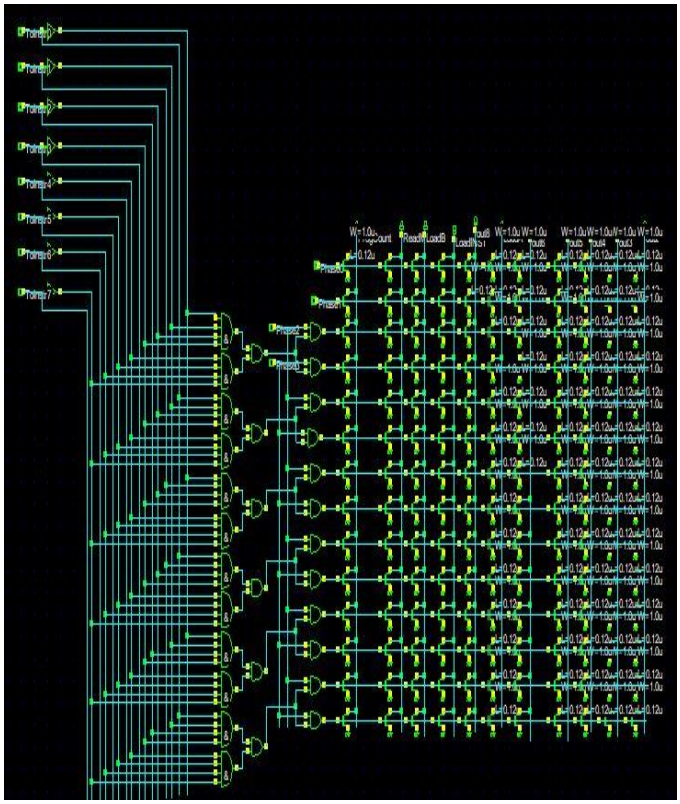


Fig 16.3. 8 bit input Microinstruction layout.

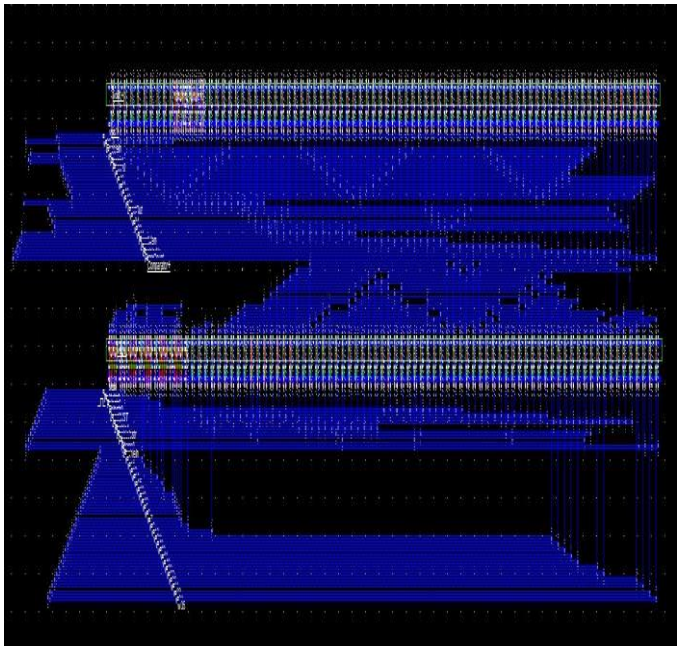


Fig 16.4. Microwind Layout

Here each output is used to give enable signal to the other blocks or Load data on Accumulator A & B and AddSub output is used to select if the ALU need to work as either Adder or Subtractor. Based on the instruction which we receive outputs operation are performed.

XVII. ADC (ANALOG TO DIGITAL CONVERTER)

A flash ADC (also known as a direct-conversion ADC) is a type of analog-to-digital converter that uses a linear voltage ladder with a comparator at each "rung" of the ladder to compare the input voltage to successive reference voltages. Often these reference ladders are constructed of many resistors; however, modern implementations show that capacitive voltage division is also possible. The output of these comparators is generally fed into a digital encoder, which converts the inputs into a binary value (the collected outputs from the comparators can be thought of as a unary value). Here we design ADC for converting the analog signal from the Temperature sensor into digital signals. The design and block diagram of the 8 bit Layout of ADC has shown below figure

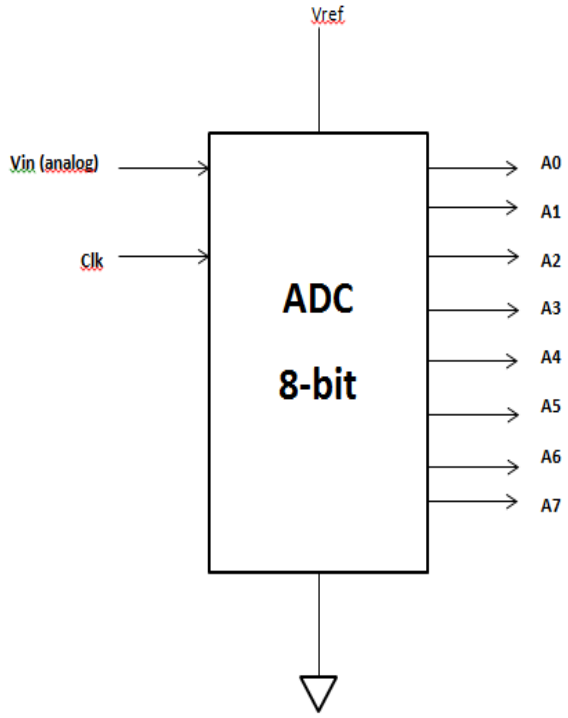


Figure 17.1, Block diagram

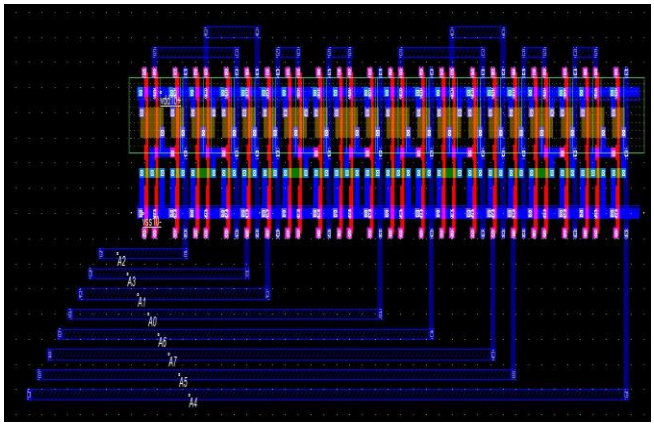


Figure 17.2, Layout of ADC

In our project, we designed 8 bit ADC for the conversion of analog to digital signals. The eight-bit analog-digital converter converts an analog value *V*in into a eight-bit digital value *A* coded on eight bits A7,A6,A5,A4,A3,A2,A1, A0. The flash converter uses 12 amplifiers, which produce results *C*0, *C*1 and *C*2,C3,C4,C5,C6,C7,C8,C9,C10, and C11 are almost logical signals as the comparators are connected in open loop. The schematic diagram of ADC is shown below,

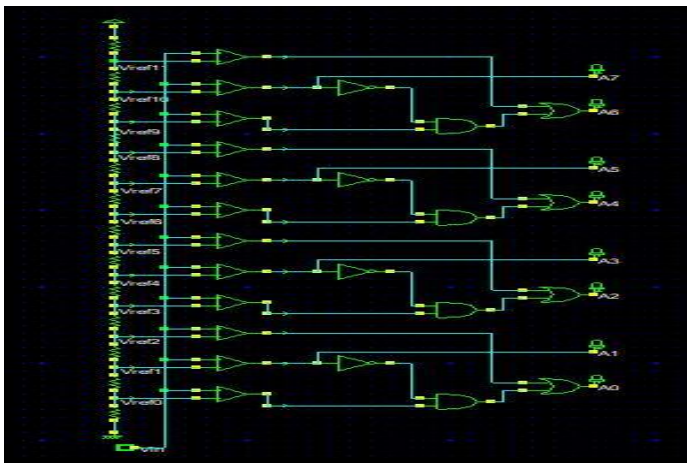


Figure 17.3 , schematic diagram of ADC

SIMULATION RESULT:

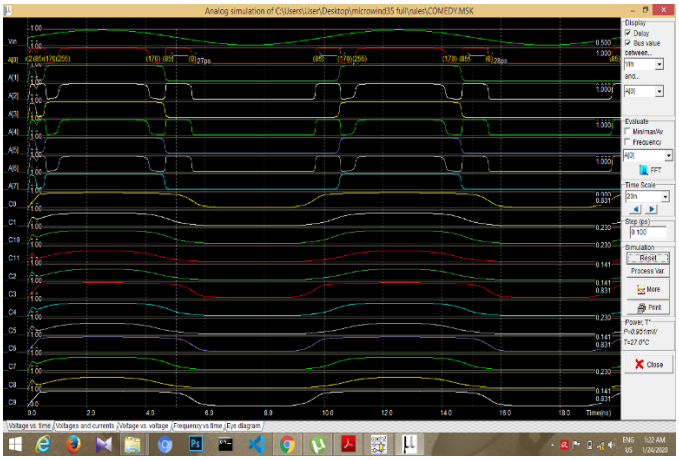


Figure 17.4 , Simulation Result of ADC

XVIII. DAC(Digital to Analog Converter)

A digital-to-analog converter (DAC or D-to-A) is a device for converting a digital (usually binary) code to an analog signal (current, voltage or charges). Digital-toAnalog Converters are the interface between the abstract digital world and the analog real life. Simple switches, a network of resistors, current sources or capacitors may implement this conversion. Here We design DAC for converting the receiving digital signal into analog signal. The project is an 8-bit digital-to-analog converter that utilizes a resister ladder network to divide current with equal current sources, and an operational amplifier to sum these currents and convert them into an output voltage. The use of an R-2R ladder architecture is very useful for binary-weighted currents. However, the R-2R based converter is easy to implement and the resistance ratio is independent of the number of bits the precision of the resistor is significant. Because the resistance of the R-2R architecture must be so closely matched (as close as 0.01% for the LSB on an 8-bit DAC) and the current ratio through the switches is still large the implementation of current sources is needed. With equal current flow through all the switches the architecture will be slower but more stable. The Layout design and Block diagram of the DAC are shown below figure ,

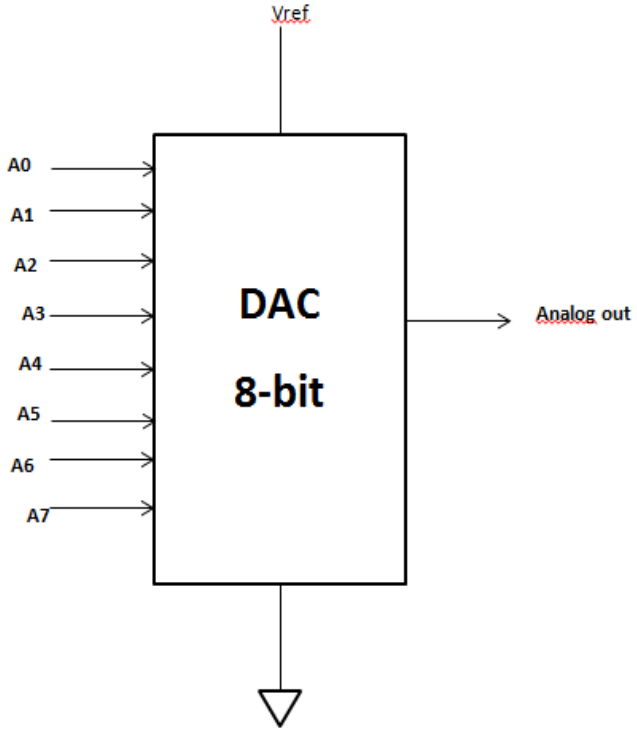


Figure 18.1 , Block diagram of DAC

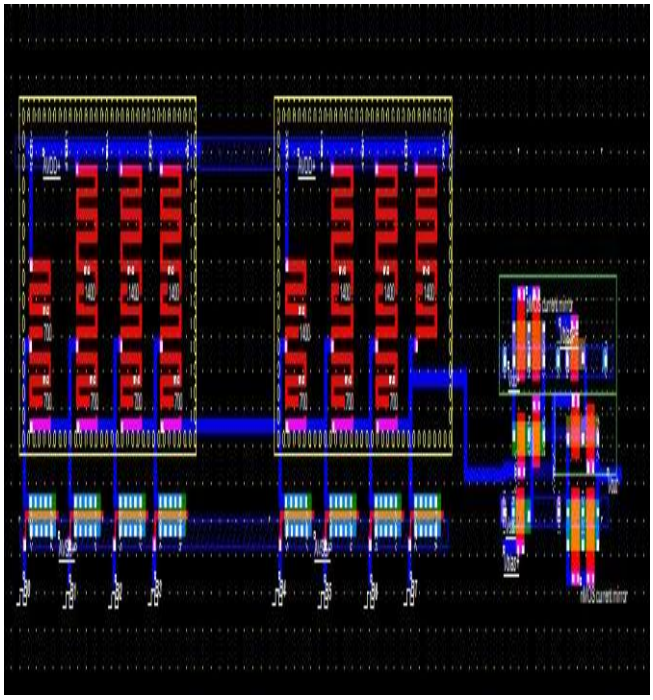


Figure 18.2 , Layout of DAC

The digital-analog converter uses the three-bit input B7,B6,B5,B4,B3,B2,B1 and B0 to control the transmission-gate network, which selects one of the voltage references (a portion of *V*dac). This voltage reference is transferred to the output *V*out. The schematic diagram of DAC is shown below,

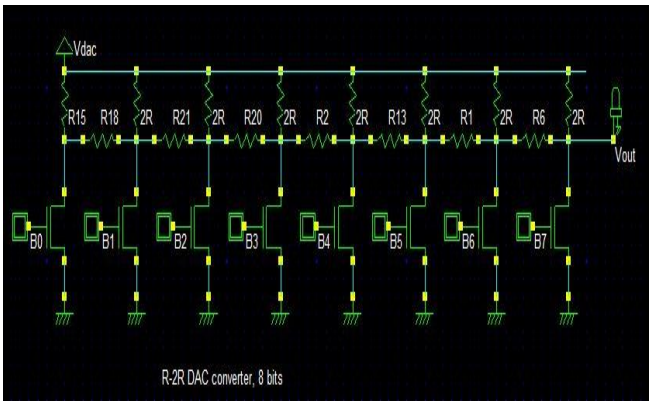


Figure 18.2, schematic diagram, of DAC

SIMULATION RESULT:

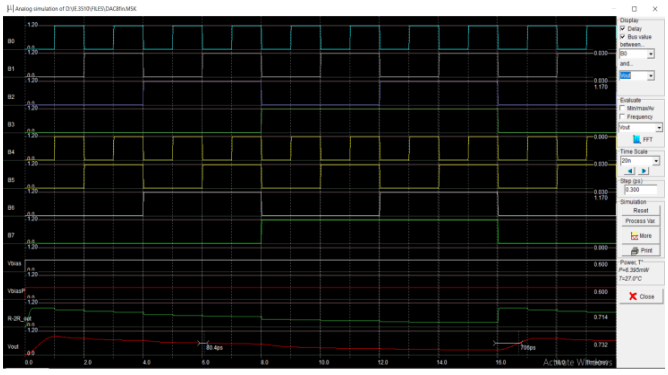
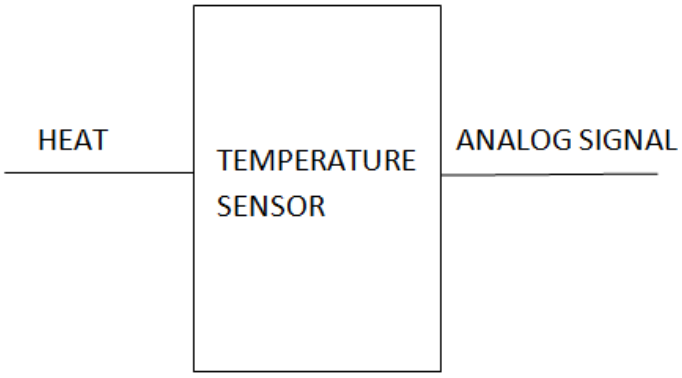


Figure 18.3 , Simulation Result of DAC

XIX. TEMPERATURE SENSOR

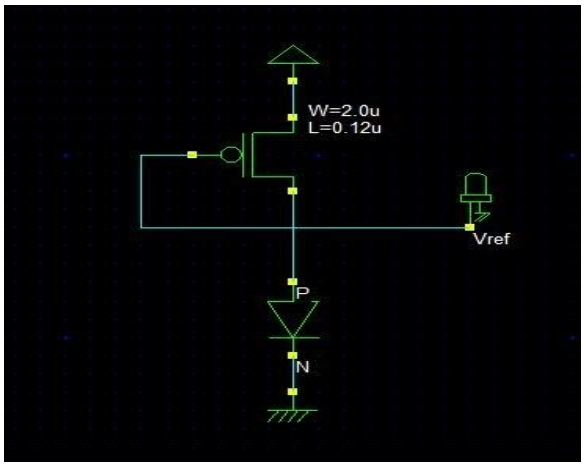
A small Integrated Temperature sensors are becoming an integral part of all high performance system on chips(SoC). It is one of the simplest temperature sensing element in the pn diode. All Temperature sensor applications fall into the temperature measurement or thermal measurement category. The diode voltage called Vak is the measure of temperature. During temperature sensing, the np diode is forward biased by a small constant current. P+/N well is the temperature sensor, when pMOS device is a load.



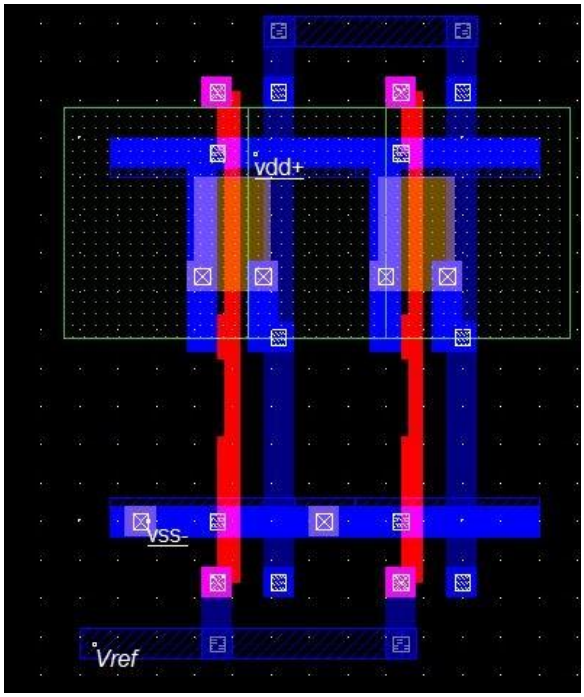
DESIGNING OF TEMPERATURE SENSOR

The n-well region is connected to the ground and it creates a p+ diffusion that makes a P+/N well diode. And the diode voltage serves as a measuring of the temperature. The expression of current that includes temperature dependent parameters,

$$Iak = IsatS(exp[q/kt(Vak)]-1$$



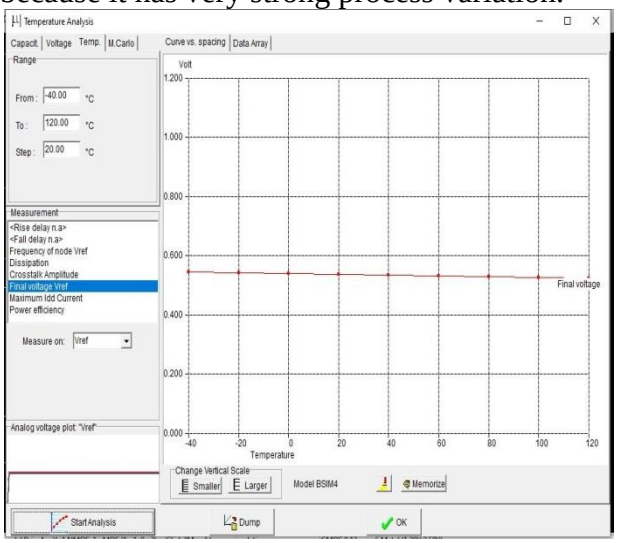
TEMPERATURE SENSOR USING MICROWIND



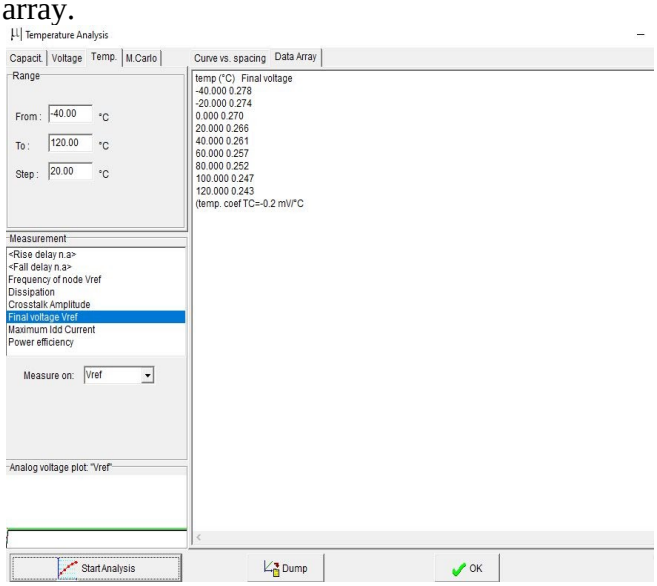
SIMULATION RESULT

The below diagram shows the temperature analysis with respect to voltage. Here it should be -40 degree

C to 120 degree C with the step of 20 degree C. Sometimes,it's hard to find the exact value of Temperature. So, Calibration process is necessary because it has very strong process variation.



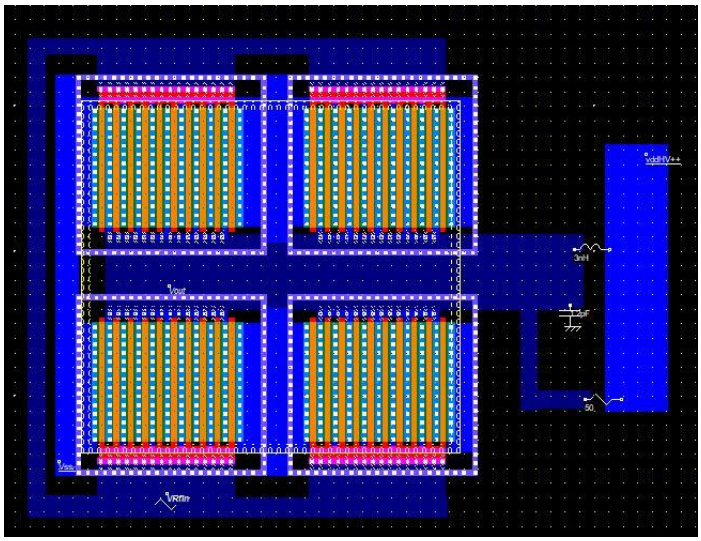
The below diagram shows the Temperature data array.



XX. BLUETOOTH

POWER AMPLIFIER

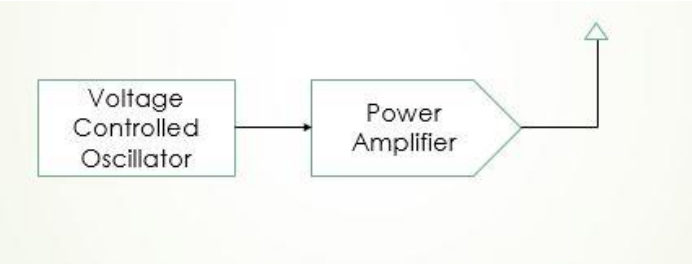
The power amplifier is part of the radio-frequency transmitter, and is used to amplify the signal being transmitted to an antenna so that it can be received at the desired distance. Most CMOS power amplifiers are based on a single MOS device, loaded with a "Radio-Frequency Choke" inductor *LRFC*. The inductor serves as a load for the MOS device (At a given frequency *f*, the inductor is equivalent to a resistance *L*.2p.*f*), with two significant advantages as compared to the resistor: the inductor do not consume DC power, and the combination of the inductor and the load capacitor *CL* creates a resonance. The power is delivered to the load *RL*, which is often fixed to 50 W. This load is for example the antenna monopole, which can be assimilated to a radiation resistance, as described in the previous section. The resonance effect is obtained between *LRFC* and *CL*. A Class A amplifier is polarized in such a way that the transistor is always conducting. The MOS device operates almost linearly.



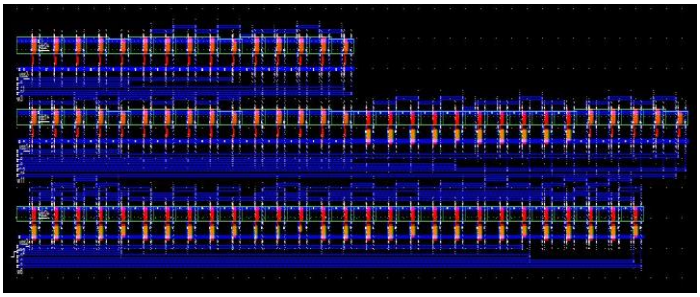
Voltage-Controlled Oscillator

The voltage-controlled oscillator (VCO) generates a clock with a controllable frequency. The VCO is commonly used for clock generation in phase-lock loop circuits, as described later in this chapter. The clock may vary typically by +/–50% of its central frequency. The current-starved inverter chain uses a voltage control *Vcontrol* to modify the current that flows in the *N1*, *P1* branch. The current through *N1* is mirrored by *N2*, *N3* and *N4*. The same current flows in *P1*. The current through *P1* is mirrored by *P2*, *P3*, and *P4*. Consequently, the change in *Vcontrol* induces a global change in the inverter currents, and acts directly on the delay. Usually more than three inverters are in the loop. A higher odd number of stages is commonly implemented, depending on the target oscillating frequency and consumption constraints.

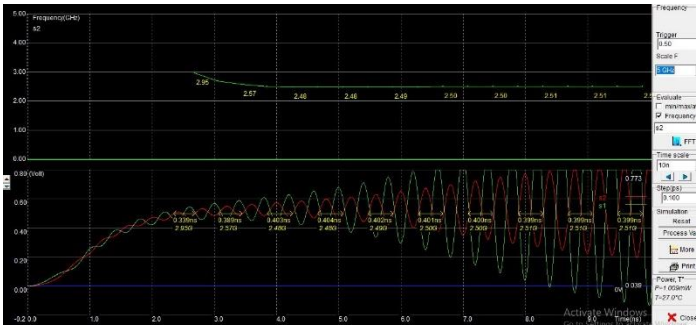
Block Diagram



Layout Design



Simulation Result



XXI. CONCLUSION:

The processor has numerus number of blocks based on the requirement but in all the blocks the main part is the clock. Based on the clock pulse the entire processor acts. In our project clock plays a vital role as we edge sensitive D-register in most of the blocks. ADC and DAC are used to make the transition easy from getting Analog data from temperature sensor and also to transmit the data by Bluetooth.

XXII. REFERENCE

[1] SICARD Etienne, BENDHIA Sonia. Embedded Memories. In: Advanced CMOS cell design. Tata McGraw-Hill Education, 2007, 364p.

[2] SICARD Etienne, BENDHIA Sonia. Basic Gates. In: Basic CMOS cell design. Tata McGraw-Hill Education, 2007, 432 p.

[3] E. Sicard, S. Delmas-Bendhia, “Arithmetics”, in CMOS Circuit Design, Simulator in hands. Ed. Mc Graw Hill, India.

[4]https://my.eng.utah.edu/~bowen/DAC_Proj/8-bit_r2rdac_current_sources.html

[5]https://www.academia.edu/8988649/8_bit_parallel_adder_and_subtractor?auto=download