

DIGITAL HEART RATE VARIABILITY MONITORING SYSTEM

Vishal Bhola vishal.bhola@utah.edu

Venkateshwara Reddy Sanivarapu venkatesh.sanivarapu@utah.edu

Siva Sandeep Garigipati sivasandeep.garigipati@utah.edu

Abstract

In this paper we describe the implementation of an application specific integrated chip (ASIC) for digital heart rate variability monitoring system. The proposed system extracts the R peaks from the input ECG signals using peak detection algorithm and stores the computed R-R intervals into an efficient SRAM as the internal memory. A histogram sorter sorts the R-R intervals which are then indexed into histograms representing the variation in patient's heart rate over a period of time. The system has been designed and synthesized for a 0.5 μm CMOS technology and placed in TCU-4 chip with area of 3x3 mm² including core and pad rings. Additional multiplexer and de-multiplexer units were incorporated in the design to allow for testing of the memory function. They were then fabricated by MOSIS and returned in packaged form. The parts received from MOSIS were tested using LV 512 tester at 2.5 MHz, where 4 out of the 5 parts received passed the testing.

Keywords: ECG, R-R intervals, peak detection, Heart Rate, IC testing, LV 512

I. INTRODUCTION

Heart rate variability (HRV) refers to the physiological phenomenon of variation in the intervals between successive heart beats, representing the underlying cardiovascular function and control mechanisms. HRV measurement and analysis finds significant applications in myriad medical applications like heart failure, diabetes, hypertension [1] and sleep apnea [2]. Furthermore, because HRV measurement is a non-invasive process, it can be used by patients suffering from chronic heart diseases to maintain a check on their condition and minimize risk. A power efficient and compact HRV system would expand the applications of HRV monitoring and has the potential to improve the overall health index of its users by complementing the medical treatment.

Figure 1 presents the block diagram for the proposed system. A peak detection algorithm identifies the R-peaks in the input ECG signal and corresponding R-R intervals are calculated in a simple peak to peak counter. The computed R-R intervals are stored efficiently in the interval memory as well as sorted by the histogram sorter to maintain a histogram of the patient's HRV over a period of time. The beat to beat intervals or the histogram data can be output through the control unit which interfaces with an external microcontroller and preferentially outputs the requested data, reading it from the internal memory.

All modules except SRAM were implemented as clocked synchronous machines with a clock rate of 2 KHz to have

more accuracy in localization of the detected peaks as compared to the benchmark design [3].

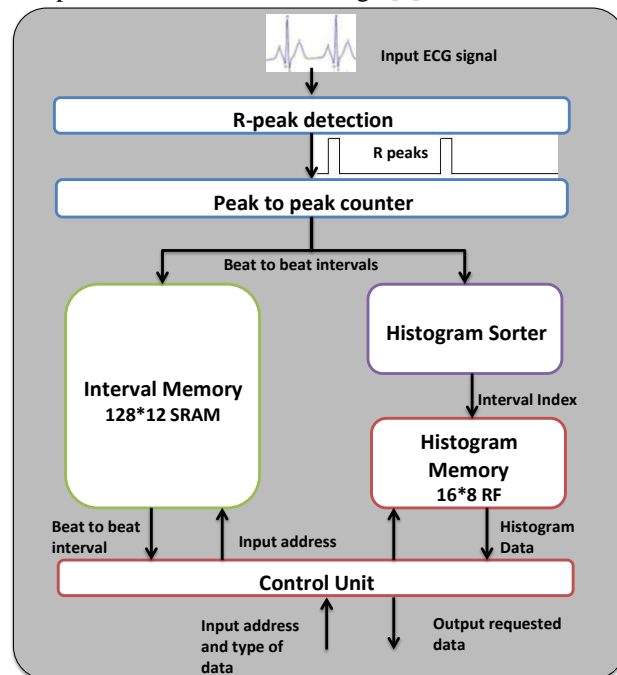


Figure 1: Block diagram of the proposed HRV system showing the interaction between different modules. R-R intervals are computed and stored into the interval memory and indexed into histogram.

The implemented design was able to identify the R-peaks correctly with a latency of 1ms corresponding to eight clock cycles that were used for the verification of the peaks detected. Since, this latency period is constant across the peaks, it does not introduce any error in the R-R interval measurements.

Figure 2 presents the placement of the proposed ASIC as an interface with the micro-controller and ECG sensor in a complete HRV system.

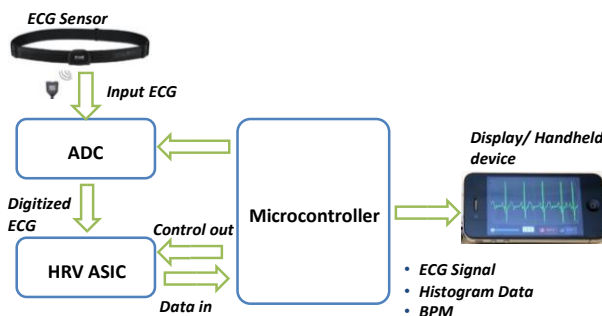


Figure 2: HRV ASIC interfaced with microcontroller and display and updating the threshold

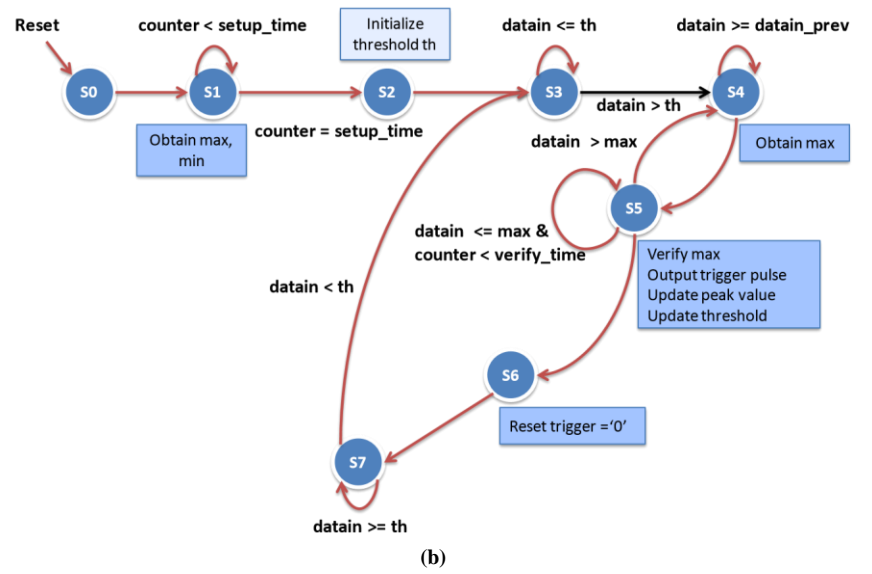
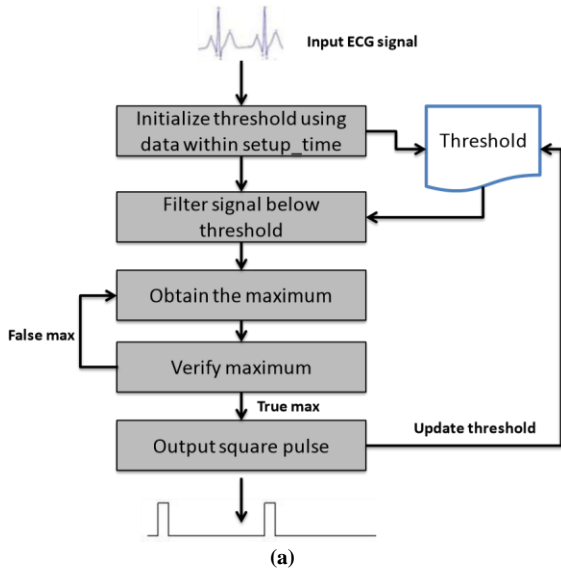


Figure 3 (a) Block diagram for the implementation of the R-peak detector, (b) state diagram depicting the various states used for obtaining the peak represented by $verify_time/ F_{clock}$. Figure 3 (a) and (b) present the block diagram and state diagram for the implementation of the peak detector module as a synchronous finite state machine.

II. SYSTEM DESIGN AND DESCRIPTION

The design of individual modules is described next.

i. R-peak detection

The peak detector is implemented using an adaptive threshold based method to detect the R-peaks in the input ECG signal. An appropriately set adaptive threshold can help eliminate the effects of unwanted variations in the input signal like baseline wander noise that is very common to ECG recordings. The initial threshold is set by extracting information from the input ECG signal for a specified time referred to as $setup_time$ here. The maximum and minimum values of the ECG signal during $setup_time$ are used to define the threshold as follows:

$$Th_{initial} = \frac{3 ECG_{Max}}{4} + \frac{ECG_{Min}}{4}; 0 \leq t < setup_time$$

The threshold value is stored in 16-bit register. A digital comparator is used to filter the input signal value lower than the set threshold. The R-peak is then obtained as the maximum value from the signal in the window where it exceeds the threshold. Following the detection of R-peak, the threshold is then updated as follows:

$$Th_{present} = Th_{prev} + RPeak_{pres} - RPeak_{prev}; t > setup_time$$

The peak is first verified for a fixed number of clock cycles ($verify_time$) to eliminate false peaks. The R-peak detection block outputs a narrow square pulse referred to as $trigger$ when it verifies the peak. This implies that all the R-peaks are detected with a latency of fixed number of clock cycles

The peak detector's performance was analyzed over varying $setup_time$ and $verify_time$ to come up with an optimal pair of values. From the analysis experiments over two different sets of data it was concluded that a $setup_time$ equal to 2000 clock cycles (at frequency 2 KHz) and $verify_time$ equal to 8 clock cycles gives good results (simulation results in the results section).

ii. Peak to peak counter

This block is a simple counter that counts between the $trigger$ pulses output from the R-peak detection block. Implemented as a synchronous design, the module maintains the counts in an internal register which is loaded into the external register at $trigger$ and reset thereafter. The $trigger$ is also made available to the interval memory, histogram sorter and histogram memory to allow for syncing of the read, write and index and operations that must be performed following the execution of R-peak detection. The output from counter module is a 12-bit interval length which is stored in the SRAM interval memory and indexed into appropriate bin in the histogram memory by the histogram sorter. 12-bit interval length can store up to 4095 clock cycles which equals a maximum 2 sec R-R interval.

iii. R-R interval memory- SRAM

Because, we have chosen area as a figure of merit for our design, we used SRAM as the preferred method for implementation of the interval memory to reduce the area that will be required on the chip. Due to the limited time and space on our module, in this project, the interval memory will use a 32X12 SRAM. The SRAM will only hold 32 intervals but can easily be scaled to make it much larger using this unit as a building block. The SRAM can be broken up into smaller

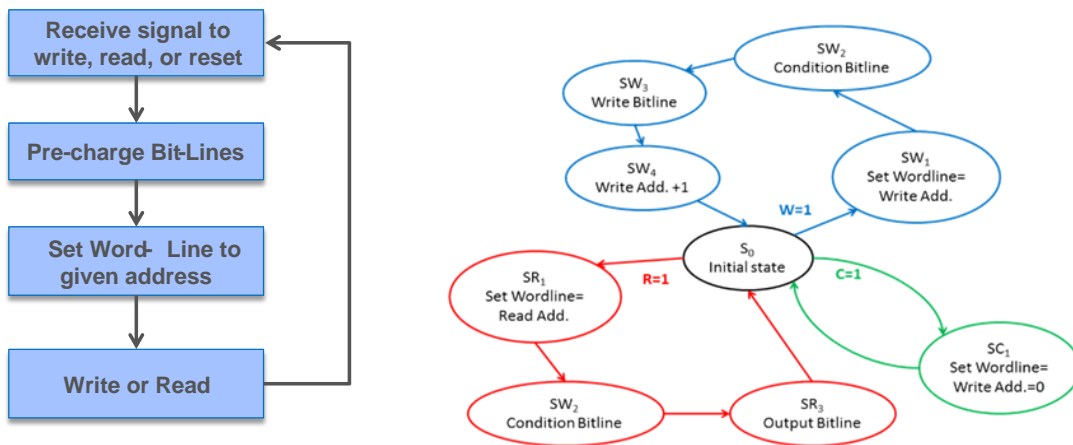


Figure 4: Block diagram and State diagram for a SRAM

functions: storing the bit in the SRAM cell, preparing to access the SRAM cell, read or write to the cell, and dealing with the address.

and prepare the SRAM for access is to pre-charge BL and BLB so that both are high.

SRAM Cell

The SRAM cell is a '6T SRAM cell' as seen in 4. This cell was chosen because of its compact size and ease of use. The cell consists of 2 inverters connected input to output so that once a value is set it will stay there as long as there is power. These inverters are sized to 1X so they can be written with small transistors. To access data on these inverters an NMOS transistor is connected to the bit and inverted bit. The other sides of the transistors are connected bit-line (BL) and inverted bit-line (BLB) respectively with the gates connected to the word-line (WL). The access transistors should be larger than the size of the inverters but still kept small as to not increase the capacitance on BL and BLB

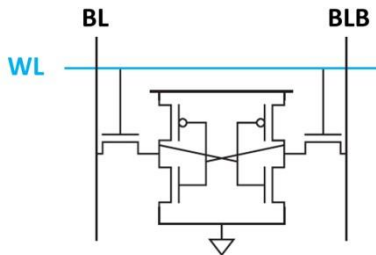


Figure 5: Transistor level schematic of the implemented 6T SRAM cell

The in many cases these cells are laid out not using standard cells but in our design the cells are made with the standard cell rules and sizes. This was done so the cells would interface well with the other cells in the library as seen in Figure . This means, in the future if the cell was going to be made smaller the size could be reduced even further.

Pre-Charging

As the number of words in each unit is increased so does the capacitance on BL and BLB. The capacitance being too large results in not only a delay limiting the speed of access but can cause the charge stored on BL to be written to the bit as soon as the cell is opened. The way to overcome this issue

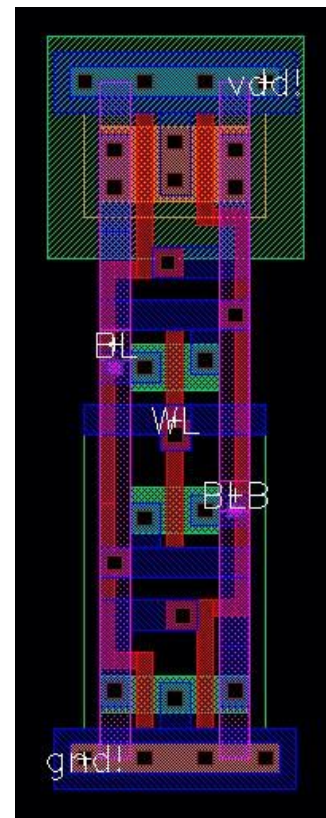


Figure 6: layout of 6T SRAM cell where the standard cell size is maintained.

Pre-charging consist of different circuits for reading and writing. In the case of the read circuit every WL must have a value of zero before BL and BLB are charged. Having a value on the WL could result in the values in those bits being overwritten to an unknown value. A decoder will always output a high value on one of the lines so the decoder cannot be attached directly to the WLs. To fix this a buffer was added which consists of a NAND and an INV. This can be used so the output of the decoder is one input and the second input will be an active-low signal, when a zero is placed on

the line a zero will output regardless of the first input and when the signal is high the value will follow the first input.

The value can then be cleared so the data can be written again. This counter then stops counting when Cout goes high

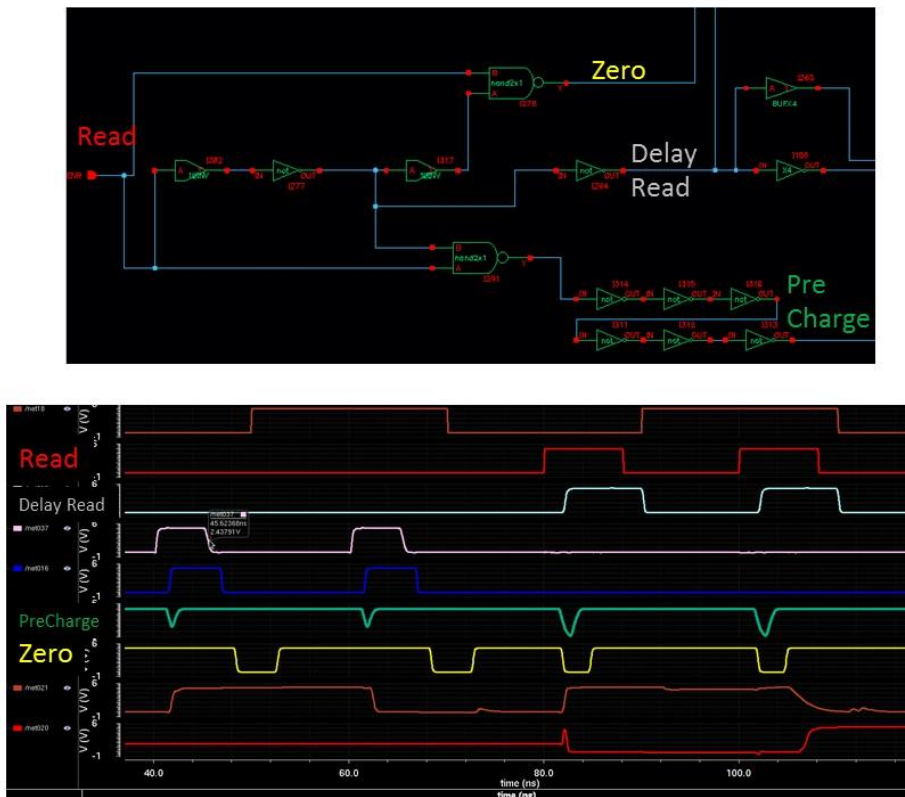


Figure 7: (a) Delay circuit and (b) spice simulation (spectre)

Once the WLs are all zero the BL and BLB can be pre-charged. This is done using a single large PMOS transistor connected to Vdd on each line. The value on BL and BLB are now both high so when the WL is activated the bit or inverted bit with a low value will begin to decrease but will take time to go below the threshold voltage and be driven low. In the initial spice simulations this delay has been seen as around 4ns for a 32 word array but for this application time is not a driving factor so a 4ns delay is no problem. In the case of writing, the word that is going to be written should be active so the values in the cells are overwritten before the next value tries to write. After the cells are pre-charged they can end up switching to one state or the other depending on which capacitor discharges more quickly, BL or BLB, therefore the value should be written soon after the pre-charge is complete.

Address

Before anything can be written or read, the wanted word line must be selected. To do this a 5 bit address is passed to the 5x32 decoder that will make the WL select to be used be activated. There are two addresses that can be selected, either read or write. The selection will be done with a MUX. The read address is passed in from the micro controller but the write address must be kept track of. To do this a counter was made with an input of enable write as the clock signal.

so to write more data the micro controller must send a clear signal.

Reading and Writing

Once the pre-charging is completed the values can be read or written. The SRAM will get a signal to read or write at which point the bit SRAM needs to prepare for access. With the clock frequency being as slow as it is rather than making the delays by waiting for the clock, inverters were added in series to delay the signals and create pulses as seen in Figure 7(a).

Once pre-charging is completed the delayed read or write signals will go to the enable of a Tri-state inverter. The read Tri-state inverter is made do minimum size so their bits can more quickly drive the output. Even with this, the delay can be seen in Figure 7(b) as 4ns. The write Tri-state inverter must drive the load of the BL and BLB so this inverter is of much larger size, in this design it is 4.5X.

With all of these components working together data can be stored and recalled when it is needed. Figure 8 shows the complete schematic of the SRAM with some of the key circuits shown.

Array of SRAM cells D) single SRAM cell.

iv. Histogram sorter

The input of the histogram sorter is the beat-beat RR interval that comes from the beat-beat RR interval counter. This input

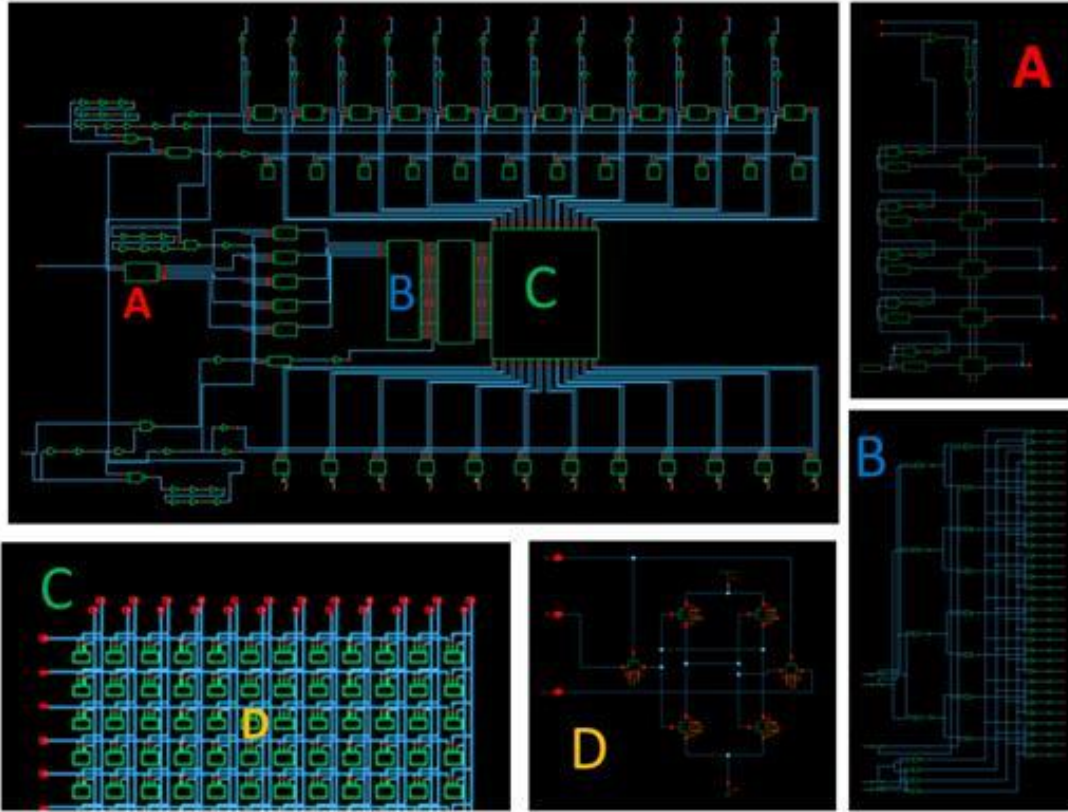


Figure 8: schematic of SRAM. A) Counter for keeping track of word to be written to B) Decoder for addressing the word lines C) is represented as 12 bit binary number. The histogram sorter categorizes these intervals into sixteen different time ranges that vary from 0 to 2 seconds.

Description of how the RR intervals are categorized into sixteen different time ranges:

The first range represents the RR interval from '0' to '0.4' seconds. The second range represents the RR interval from '0.4' to '0.6' seconds while the third range represents the RR interval from '0.6' to '0.7' seconds. The fourth to thirteenth time ranges represents the RR intervals that range from '0.7' to '0.9' seconds with a 20 milliseconds increment. Again the fourteenth and fifteenth time ranges represents the RR intervals from '0.9' to '1' second and '1' to '1.2' seconds respectively. The sixteenth time range represents RR intervals greater than '1.2' seconds. Finer resolution has been used R-R intervals between 0.7-0.9 s because of the higher variability observed in this range. Since we use 2 KHz sampling rate, all the above specified time ranges can be converted to 12bit binary numbers by simply multiplying the corresponding time range with 2000. For example, in the first time range, 0.4 seconds can be represented as 12 bit binary number as $0.4 \times 2000 = 12d'800$. This will be compared with the 12 bit binary input that comes from the beat-beat interval counter. In the histogram sorter, the basic block that is being used is the comparator.

TABLE 1: Sixteen different time ranges for comparing the input R-R interval and selecting the appropriate bin

Time Range	Histogram Bin Selected
0.0 – 0.4	1
0.40 – 0.6	2
0.60 – 0.7	3
0.70 – 0.72	4
0.72 – 0.74	5
0.74 – 0.76	6
0.76 – 0.78	7
0.78 – 0.8	8
0.80 – 0.82	9
0.82 – 0.84	10
0.84 – 0.86	11
0.86 – 0.88	12
0.88 – 0.90	13
0.90 – 1.00	14
1.00 – 1.20	15
1.20 – 2.00	16

For each comparison stage, we use two comparators: for comparing the 12 bit binary input with the lower boundary and the upper boundary of that time range. Since there are sixteen different time ranges, there should be $16 \times 2 = 32$ comparators that should be used. But in the last stage, there is no upper boundary for the time range. On a whole we have $32 - 1 = 31$ comparators used in the histogram sorter.

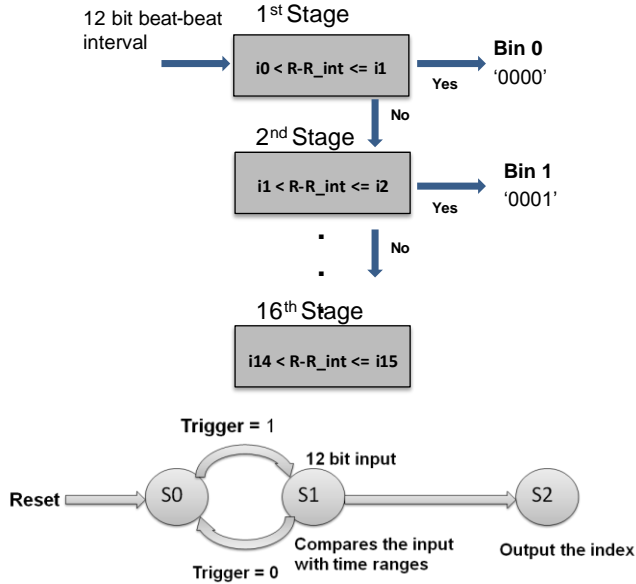


Figure 9: Block diagram and state diagram of Histogram Sorter

v. Histogram Memory

We have used the histogram memory in order to provide a graphical representation of the RR time intervals. The histogram memory consists of 16bins, each with an 8-bit register. The input is the 4-bit address obtained from the histogram sorter. The value of each register in the histogram memory is the number of beats for the interval represented by the register.

The working of the histogram memory is explained by depicting the state diagram of the histogram memory.

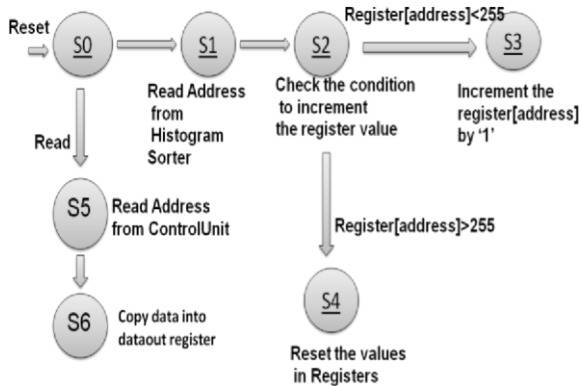


Figure 10: Histogram Memory State Diagram

As shown in Figure 9, histogram memory block performs two functions: 1) Increments the corresponding register value

after checking the condition and 2) Loads the corresponding register value into the output register when the 'read' signal is high. When the histogram memory block receives the address from histogram sorter block it checks the value in the corresponding register. If the value is less than the register file limit then the register value is incremented by one, otherwise, all the data in the histogram memory is moved to an external memory and then the entire block is reset. The histogram memory obtains the read signal from the control unit. When the read signal is low the histogram memory performs the function of incrementing the register values based on the input address, and when the read value is high then it performs an additional function of reading the corresponding data from the address obtained from the control unit in to the output register.

vi. Control unit

The control unit thus acts as an interface between the external micro-controller and the HRV ASIC. The control unit obtains the input Address, read request and an enable signal from the microcontroller, and based on the Enable signal, data from one of the memories is chosen. If the Enable signal is high, then data is read from Interval memory, otherwise data is read from the histogram memory. The output of the control unit block is 12-bit data, which is sent to the microcontroller. When the control unit receives high reset signal then it resets the every block in the system.

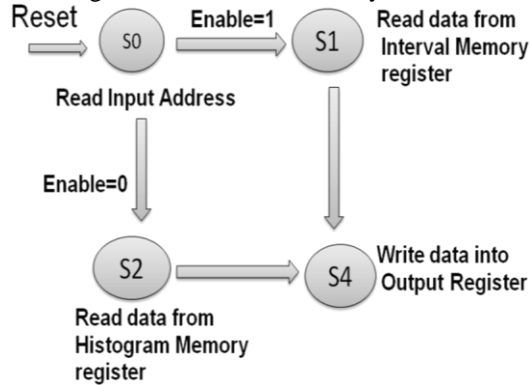


Figure 11: Control unit state diagram

III. SYSTEM IMPLEMENTATION

A standard library using 0.5u CMOS technology reference library, was designed with the following cells: INVX1, INVX2, INVX4, INVX8, BUFX4, BUFX8, NAND2X1, NAND2X2, DFFSR, FAX1, COMP1, MUX2X1, XOR2X1, TIELO and TIEHI. Synopsys Design Compiler was then used to synthesize the gate level design from the behavioral verilog codes for all the modules except SRAM which was designed in Virtuoso schematic editor and manually doing the layout using the standard cells. For the peak detection module, the synthesis was forced to use the single bit comparator (COMP1) and results in 10% improvement in area. Because the application is not highly performance incentive, we chose area as a figure of merit in our design and hence were the reason for custom synthesis of the peak

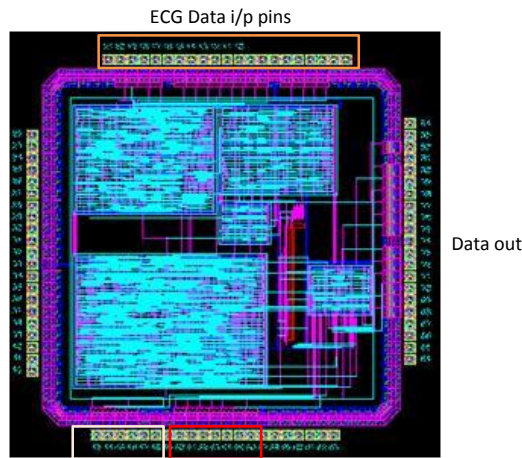


Figure 12: Final chip after placement and routing of all cells and pad rings

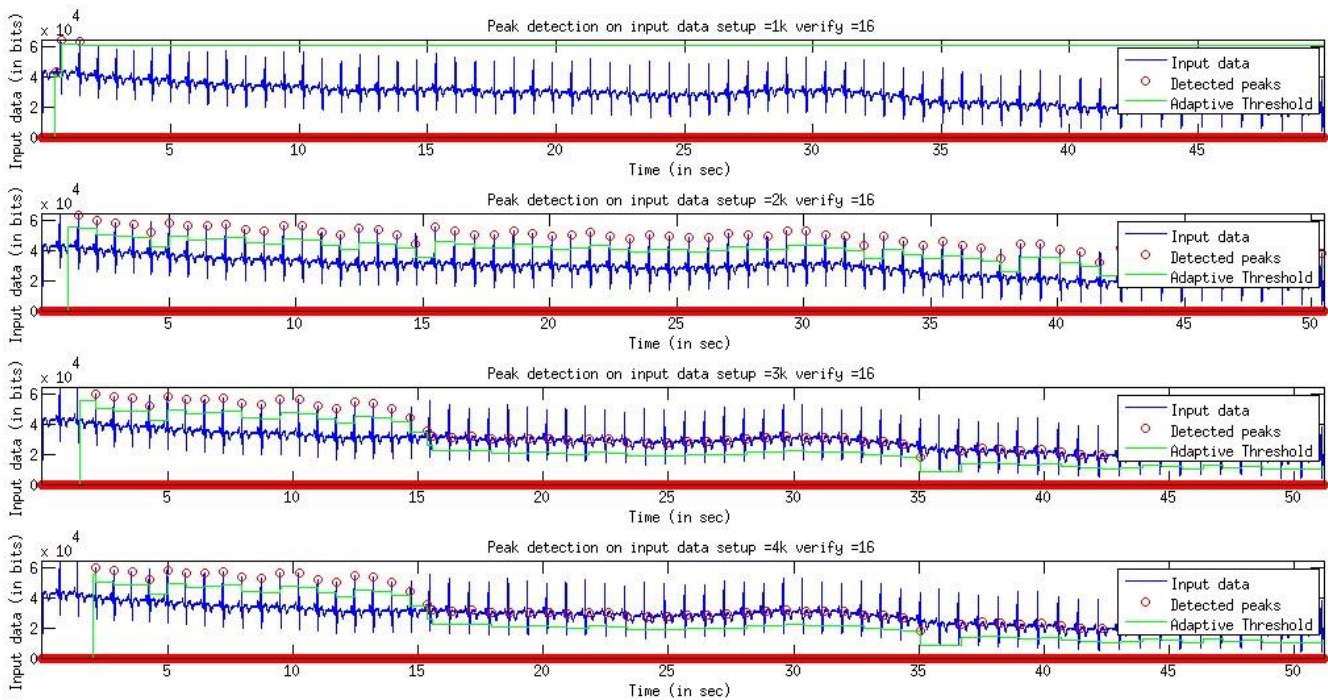


Figure 13: R-peak detection results for varying setup time, test dataset: s0015lre

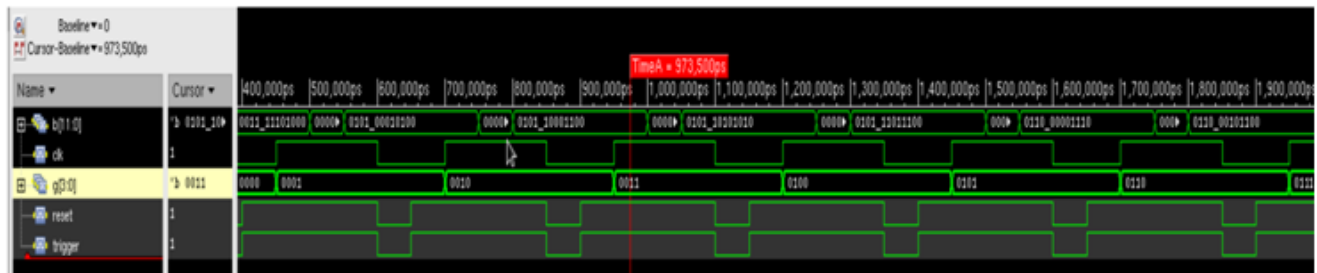


Figure 14: Simulation results for histogram sorter

detector and histogram sorter modules which are highly comparison intensive. The individual modules were placed and routed using Cadence Chip Encounter. The modules integrated together fit in a core of size 2.295 * 2.295 mm² area

and placed in TCU-4 frame for connecting the pads. The final placed and routed integrated design with core and pad rings is shown in Figure 12.

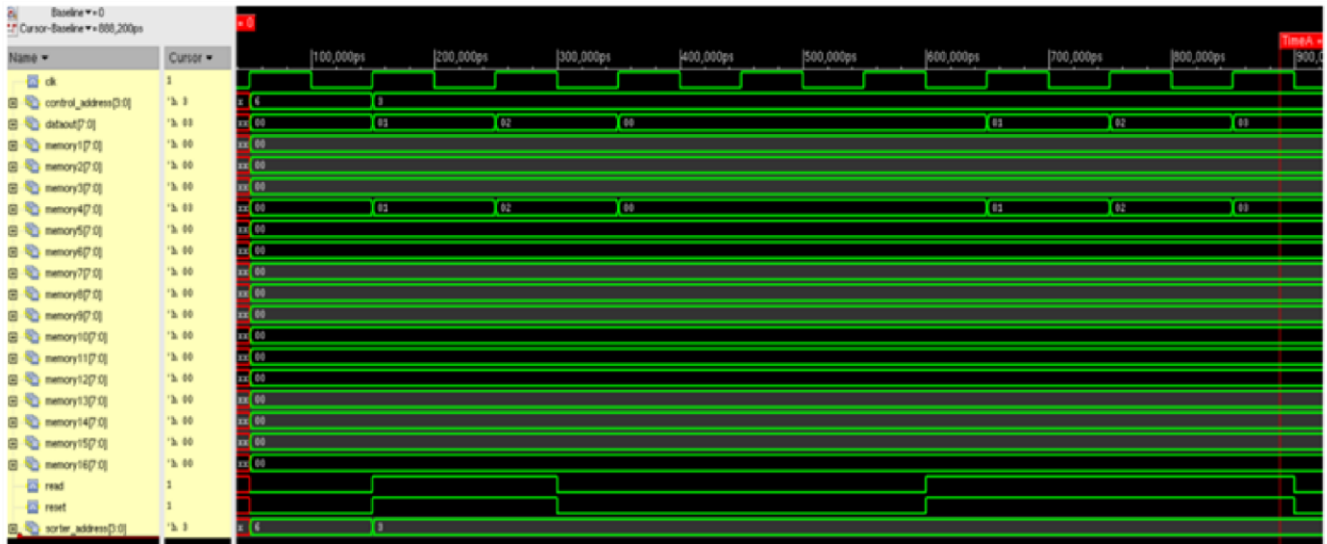


Figure 15: Histogram memory simulation

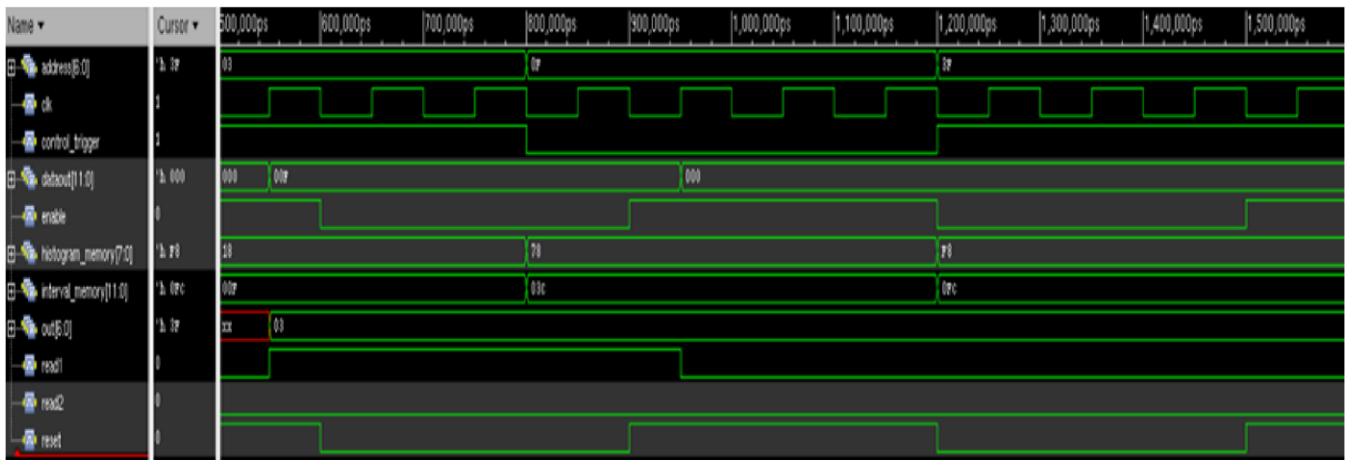


Figure 6: Control unit simulation

IV. RESULTS AND DISCUSSIONS

In this section we present the simulation results for all the modules. Two test datasets s0017re and s0015re were downloaded from Physionet ptbdb database. Lead v6 was used for generating the results for both the test datasets. The input ECG was interpolated to a sampling frequency of 2Hz, scaled and converted to 16-bit unsigned representation and imported into NC Verilog for testing. The results are shown in Figure 13 for dataset s0015re for varying *setup_time*. Figure 14, shows the simulation results for histogram sorter in NC Verilog. The ‘reset’ and ‘trigger’ go high and there is a positive edge of the clock. The 12 bit input falls in the fourth time range and hence the output is displayed as ‘0011’.

Figure 15 depicts the working of histogram memory block. The bins corresponding to the input index from histogram sorter are incremented at the trigger pulse.

The simulation results of the control unit in Figure 16 depict the proper functioning of the block. In the shown simulation,

the control unit obtains the input signals, address and enable, from the microcontroller and based on these two signals the output of the control unit is changed. All transitions occur at the positive edge of the clock pulse. When the reset signal is high there is no output, and when it’s low the output register has the data from the corresponding memory register based on the enable signal.

V. FABRICATION

A multiplexer/ de-multiplexer was implemented and included in the design to allow a switch between simulated memory and the implemented SRAM. This was done to account for possible issues in SRAM post design fabrication. The updated schematic and layout after this modification are shown in Figure 7 and Figure 8 respectively. The GDS stream was generated post synthesis and submitted to MOSIS for fabrication in 0.5 um technology.

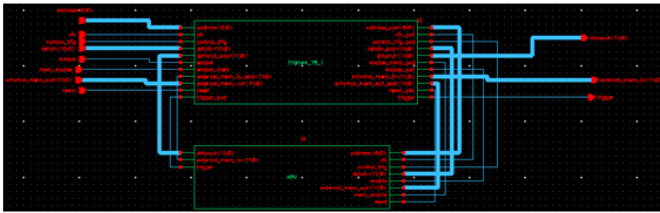


Figure 7: Schematic after incorporating the multiplexer and demultiplexer

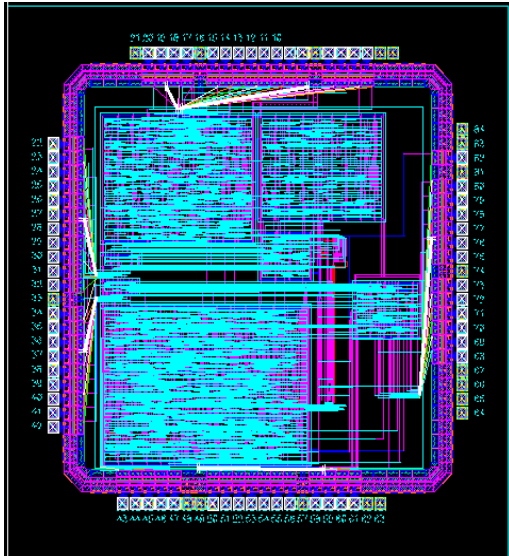


Figure 8: Routed layout after incorporating the multiplexer and the demultiplexer

Next, a mapped bonding diagram (Figure 9) was created by connecting the chip pins of the design to the pad pins with a straight black line. This basically defined the pin connections in the packaged chip. The fabricated chip as received from MOSIS is shown in Figure 10.

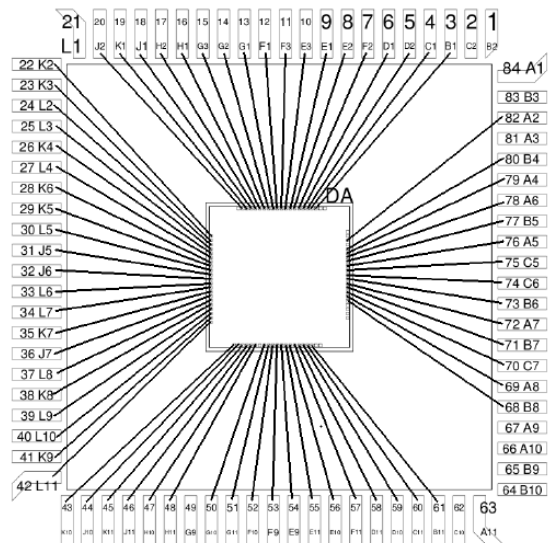


Figure 9: Bonding diagram showing inter-connects for chip packaging

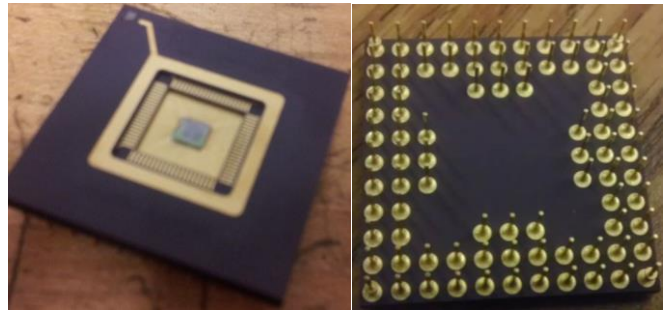


Figure 10: Fabricated IC received from MOSIS

VI. CHIP TESTING

i. Test Environment

The chip was fabricated in the AMI 0.5 μm CMOS technology and its functionality was verified using Digital IC Tester, Tektronix LV 500 machine. Tektronix LV-500 machine was built in 1989 on 'Tektronix DAS 9200 Logic Analyzer Platform' for testing ASICs. This LV500 tester has test speeds up to 50 MHz and can test up to 64,000 unique test vectors. The network connection for uploading the tests is through 'Thinlan ethernet'. LV500 tester has mainly two flavors which are LV512 and LV514. The LV 512 tester has 128 test channels which are divided into 8 sectors and all these 128 channels are usable. This LV 512 tester is mostly used for tutorial purposes. After passing the tutorial, the chip was placed on LV 514 tester (shown below) which has 192 channels and out of these channels only 160 channels are usable. These 192 channels are divided into 12 sectors.

To start testing the fabricated chip, we have to have the following things

1. DUT (Device Under Test) which connects the chip to LV 514 tester
2. Generate 'Test Vectors' with all combinations of input vectors and its corresponding output results
3. Make sure the testing equipment, LV 514 is configured properly

The input vectors and the expected output vectors were generated using NC Verilog Simulation. The ECG waveform sampled at 2 KHz was generated from MATLAB in a format compatible with NC Verilog. There are two main output variables to be tested here, of which DATAOUT is the first one and Control Trigger is the second. Due to a design flaw described in the testing issues section, DATAOUT cannot be tested and hence we tested only the Trigger output.

ii. Test Setup

This section describes the MSA file that is used for preparing the setup for the LV 512 tester. The MSA file includes all the input vectors and the expected output vectors, configuration to setup a test on the LV512 tester.

The signals that need same Voltage characteristics are grouped in the same sector and the signals that need the same timing are grouped in the same quadrant. All the input and

output signals are divided into groups - Address, Reset, Control trigger, Enable, Mem_enable, Datain, Clock, Trigger and Dataout.

The MSA file defines for each group the following information – dut sector, channel, phase, radix and specify timing once for whole group. The input signals are set to ‘Force’ on ‘dnrz_l’ and output signals are set to ‘Compare’ on ‘edge_t’. Six different phases were used based on the mapping of the respective pins. Table II shows the signal groups and the phases used for each group.

TABLE II: Input and Output Signal Groups and their respective phases

Signal Group	Phase
Address	2B
Control_trigger	1B
Datain	1D, 2A
Enable	1B
Mem_enable	2B
Dataout	1A
Trigger	1A
Reset	1C
Clk	1C

iii. Testing Issues

1. While creating the test vectors in NC Verilog from the design, a design bug was observed, which dealt with the resetting of the states in the multiplexer. Due to this design bug, the functionality of the blocks following the R-peak detector could not be tested. The only testing that could be performed was the testing of ‘Trigger’ output which represents the location of detected R-peaks in the input ECG signal.
2. The IC never failed while generating the Schmoos from the LV 512 tester. This was a tester issue because the Schmoos testing range included the values for the schmoos variables where it was observed to fail in the individual runs.
3. Out of the 5 parts received from MOSIS, one part simply didn’t function. This can be possibly due to the handling of the chip in and out from the packaging. We used this chip then to confirm if the die was accidentally rotated inside. But, the die was as designed.

iv. Testing Results

The chips were successfully tested for their functionality using the LV 512 tester equipment. Out of the 5 parts received from MOSIS, 4 ICs passed the test with the design bug as described earlier in the testing issues section. Figure 11 and Figure 12 show the schmoos plots with the Phase Delay and the Force High Voltage values for which the ICs worked correctly. Note that these schmoos plots were generated manually by observing the range of schmoos variables where the ICs did not function as expected.

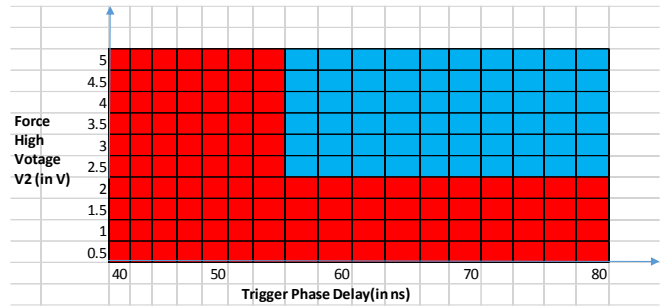


Figure 11: Schmoos plot for variables Trigger Phase delay and Force High Voltage. Blue-pass, Red- fail

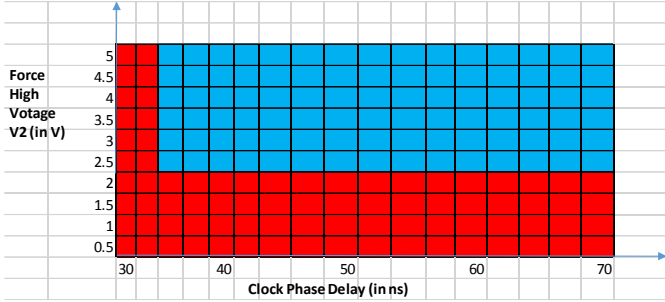


Figure 12: Schmoos plot for variables Clock Phase delay and Force High Voltage. Blue-pass, Red- fail

In summary, the ICs failed wherever the delay from inputs to the clock was less than 14.5 ns and wherever the delay from clock/ reset to the output comparison was less than 16.5 ns. The IC functioned well at 2.5 MHz clock speed. However, this high speed is not required for ECG R-peak detection algorithm because it is normally sampled at 1 KHz and in some cases at 2 KHz as is the case in the proposed implementation. The testing results are summarized in Table III.

TABLE III: Testing Results Summary

Design Number	88683
Fabrication ID	V31F-DA
Package Type	PGA84M
Number of parts received	5
Number of parts tested	5
Number of parts functional	4*
Part speed	2.5 MHz

VII. CONCLUSIONS

We have designed and implemented an HRV ASIC that can be integrated with a microcontroller and displays for the monitoring and analysis of the HRV of the users. The individual modules have been tested for functionality and simulation results for two test datasets from Physionet show that the peaks are detected with high accuracy. The counter then counts the interval length. Histogram sorter is able to sort the intervals input to it based on the ranges defined in the design. Control unit maintains the interface with the external microcontroller and has been simulated for proper functionality by showing the preferential output of the interval

memory and histogram memory based on the select signal received from the microcontroller.

The parts received from MOSIS were successfully tested using the LV 512 tester. It should be noted that the ICs passed with a design bug that allowed the partial testing of the ICs. The ICs functioned well even for speeds up to 2.5 MHz which is much higher than the normal frequency range of $\leq 2\text{KHz}$ in almost all ECG applications.

- d. While doing final place and route using VCAR make sure the instances carry same name also in the schematic and layout
- e. Be patient and hang in there!! Things will work eventually.

VIII. REFERENCES

1. Task Force of the European Society of Cardiology and the North American Society of Pacing and Electrophysiology, "Heart rate variability standards of measurement, physiological interpretation and clinical use," *Circulation*, vol. 93, pp. 1043–1065, 1996.
2. M. Hilton, R. Bates, K. Godfrey, M. Chappell, and R. Cayton, "Evaluation of frequency and time-frequency spectral analysis of heart rate variability as a diagnostic marker of the sleep apnoea syndrome," *Med.Biol. Eng. Comput.*, vol. 37, no. 1, pp. 760–769, 1999.
3. Massagram W, Hafner N, Chen M, Macchiarulo L, Lubecke VM, Boric-Lubecke O. "Digital Heart-Rate Variability Parameter Monitoring and Assessment ASIC". *IEEE T Biomed Circ S* 2010; 4:19-26.
4. Physiobank. [Online]. Available: <http://www.physionet.org>
5. LV514, LV 512 tutorials and manuals
6. The MOSIS Service <http://www.mosis.org/>

ACKNOWLEDGMENTS

The team would like to thank Prof. Ken Stevens for his constant help and guidance throughout the project. We would also like to thank Dipanjan Bhadra and other graduate students who spent a lot of time with us trying to make things work for the library characterization.

WORDS OF WISDOM

These are some pointers we would like to pass on to the next batch so they don't face all the problems we faced.

1. **Tools:** The best and the worst part. As Ken puts it, "You can't live without them and living with them is not easy either". Please focus on the labs very well in order to face less trouble during the project.
2. **DON'T:**
 - a. Go towards complex cells. Keep things as simple as possible.
 - b. Use pass transistors. They look good on paper but don't characterize.
 - c. "ASSUME" anything in the tools. Treat them like a 10 year old child and explicitly specify everything.
3. **DO:**
 - a. Take backup at every step
 - b. It's not your fault always if the simulations don't work. Try deleting the simulation run folder and you will see wonders happening.
 - c. While making cells for library, label all pins in caps.