

Exploring and Prototyping Designs for Biomedical Applications

Researchers at the University of Hawaii at Manoa have implemented ECG analysis algorithms with Xilinx System Generator.

by Ashish Shukla
Graduate Student
University of Hawaii at Manoa
ashishshuklabs@gmail.com

Luca Macchiarulo
Assistant Professor, Department of Electrical Engineering
University of Hawaii at Manoa
lucam@hawaii.edu

Many physicians use electrocardiogram (ECG) machines to monitor the electrical activity of the heart and the heart condition in general. But today, there is a lengthy delay in the time it takes to transfer data from ECG monitoring machines to trained physicians.

Here at the University of Hawaii at Manoa, we are researching ways to transfer – in real time – preprocessed ECG data from the patient's heart to physicians. As a first step toward this goal, we implemented two variants of a well-known software detection algorithm for ECG in hardware and explored the design choices using Xilinx® FPGA tools and hardware.

Many biological instrumentation-based designs require designers to combine filtering stages and customized logic such as finite state machines in the same system. But now it is easier for researchers to design filters in the Xilinx System Generator environment by simply connecting the various blocks from the Xilinx blockset (included with the System Generator blockset library) instead of creating these modules from scratch using a hardware description language like Verilog or VHDL.

Using System Generator to create most of the blocks allows you to concentrate on the critical parts of the design and delegate the details of the implementation of those standard modules to the tool. You can then import your custom logic modules into the System Generator environment; the tool will integrate those custom logic blocks with the rest of the design.

Because noise contamination is an inherent problem in ECG monitoring and we cannot completely remove it, we had to come up with a way to suppress noise contamination.

Automated ECG System Analysis

A heart's natural pacemaker, composed of special self-exciting cells, generates and propagates a polarization and depolarization electrical signal that regulates the proper contraction of the heart muscle. This electrical activity is recorded as an electrocardiogram (ECG) by a machine called an electrocardiograph, which provides physicians with a wealth of information on heartbeats and the heart's condition in general [1].

The QRS complex – a wave structure that corresponds to the depolarization of ventricles and has a spiked shape in the ECG – is often the most telling waveform found in an ECG signal. The morphology, duration, and amplitude of the QRS complex in an ECG signal provides significant information to physicians diagnosing various arrhythmias and other cardiac ailments.

Health professionals can best judge the state of a patient's heart by monitoring the heart's activity (and the QRS complex) during stress or physical activity. Therefore, health professionals often connect ECG monitors to their patients for many hours (generally 24 hours at a time).

Currently, most health professionals use Holter System monitors for this purpose. But Holter System monitors have an inherent battery power and processing power constraint that limits their functionality to data acquisition systems.

A Holter System's heart monitors record the patient's ECG data to flash memory or to a tape attached to the system. A technician then removes the drive from the Holter System and sends it to the lab, where technicians analyze the data. Once the lab technicians complete their analysis, they send the ECG report to the physician. Of course, this means that there is a delay between hooking up the monitor to the patient and getting the ECG data to the patient's physician, thus delaying treatment, possibly with tragic consequences.

Here at the University of Hawaii at Manoa, we are researching how to create ECG analysis systems that will reduce the amount of time it takes to transfer data from the patient's heart to the physician by carrying out the analysis in real time. We hope to achieve this by integrating the ECG analysis system into portable ECG monitors, which physicians can use directly and thus eliminate the technician translation/analysis step.

To aid us in this effort, we have implemented real-time ECG analysis on a Xilinx Spartan™ XC3S500 device in the Spartan-3E starter kit, which allows the system to perform this analysis much faster than previous software-based methods. FPGAs offer many advantages for this and other applications because they support rapid prototyping, are less expensive than ASICs at low volume, and are quickly reprogrammable.

Furthermore, their fast and efficient testing option, combined with System Generator software, makes them a great choice for algorithm exploration as well as hardware implementation and prototyping.

Implementation of the Algorithm in Hardware

In our research, we are implementing a well-known algorithm by Tompkins and Hamilton for QRS detection in hardware ([2], [3], [4], [5]). To help us with this effort, we took advantage of the fast prototyping features included in the Xilinx toolset.

In our system, the processing of the ECG signal begins with a number of filtering stages through which a signal must pass before the system accurately detects a QRS complex. We divided the design into two main stages. The first stage, or pre-processing stage, comprises four linear filters and one non-linear filter. The second stage, or peak detection stage, identifies the peak signal in the QRS complex and applies decision rules to qualify a feature as a QRS complex.

Filtering Stage

The ECG recording is very sensitive to even the smallest body movement or noise, such as electrical muscle (myographic) signals and the system's own electrical power-line interference. Because noise contamination is an inherent problem in ECG monitoring and we cannot completely remove it, we had to come up with a way to suppress noise contamination. We achieved this by employing filtering during the preprocessing stage (as in [4], [5]).

To do this, we first pass an ECG signal through an infinite impulse response (IIR) low-pass filter, which suppresses the high frequency noise in the signal. Then the signal passes through an IIR high-pass filter, which attenuates the P and T waves in the signal and suppresses the DC offset present in the signal.

The low-pass and high-pass filter together form a bandpass filter. It then feeds the output of the high-pass filter to a finite impulse response (FIR) derivative filter, which further emphasizes the QRS complex. This typically presents a more pronounced slope variation compared to the other signal features, making it somewhat easier for observers to identify [1].

The FIR derivative filter step also helps further reduce the noise content of the signal. After filtering, we employ a squaring stage that carries out non-linear amplification of the QRS complex and makes all data points positive.

Finally, we use the output of the squaring stage as the input to a moving window integrator. Each sample output from this filter is an average of the previous 32 values. We implemented all these filtering stages directly in the Simulink environment using the System Generator tool. Then we tried out two different approaches for the next stage, which is a moving window integrator.

The first approach involved the connection of a number of register, delay, and adder blocks to implement the filter as a



Figure 1 — Input ECG and filtering stages output, from top to bottom: original signal, low-pass output, high-pass output, derivative output, squaring stage output, moving window integrator output. The arrow indicates the position of the QRS complex.

direct Form I structure. With this approach, however, we ended up using 31 adders and an equally large number of delay or register blocks.

We investigated further and later came up with a more resource-efficient structure that uses the block RAM resources in the Spartan-3 FPGA. This final structure uses a

small RAM of 32 40-bit words and just two adders and is also very efficient in timing. Figure 1 shows a typical ECG input, with results of the various filtering operations.

Peak Detection Stage

In the peak detection stage, we are trying to find the peak in the output of the moving

window integrator. The peak detection process depends on the calculation of the threshold value, and we can locate a peak among the sample values that are greater than the threshold value. But the algorithm does not report a peak until a sample appears in the falling slope of the moving window integrator signal with a value less than half of the peak value.

We use this method because it reduces the number of false peak detections caused by noise. Once the algorithm locates the maximum point of the QRS signal in the moving window integrator output, we search for a peak in an appropriately delayed copy of the output of the high-pass filter and use it as a fiducial point for the position of the QRS complex.

Essentially, we are finding the peak of the QRS complex in the output of the bandpass signal instead of the original signal, because the original signal is highly contaminated by noise. We achieved this using a memory module, which records the previous 60 samples from the high-pass filter and sends a fixed interval of samples

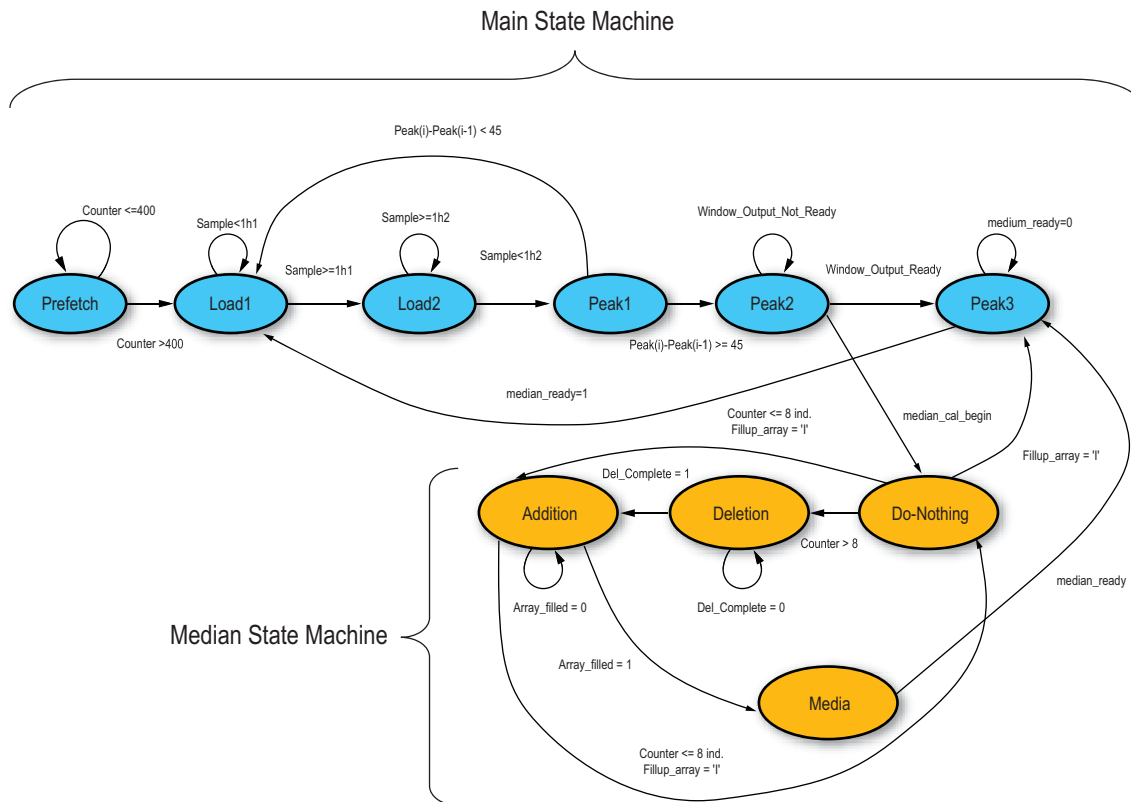


Figure 2 – State machine for the complete design

containing the QRS complex as an input to the window module. The window module finds the maximum QRS signal.

We based the threshold for the next peak detection process in the moving window integrator output on the median of the eight previously detected peaks.

To calculate the running median calculation, the system maintains an updated, sorted list of such peaks. The computation is supervised by a median calculation FSM (called the median state machine) communicating with a separate peak detection FSM (called the main state machine) (see Figure 2). The main state machine finds the peak in the moving window output (as discussed); then the median state machine sets up the threshold value for the next detection.

We implemented the entire peak detection stage in VHDL. It contains the memory, window, main, and median state machines and the median RAM modules. Later, we imported these VHDL modules into the System Generator environment as black boxes and simulated the design (Figure 3).

Programming and Testing the Hardware Implementation

We programmed the FPGA through the JTAG port. To test the design, we used System Generator's JTAG co-simulation feature, which allowed us to pass data from the Simulink environment through the USB cable to the board (the Xilinx Spartan-3E starter kit) containing the downloaded design.

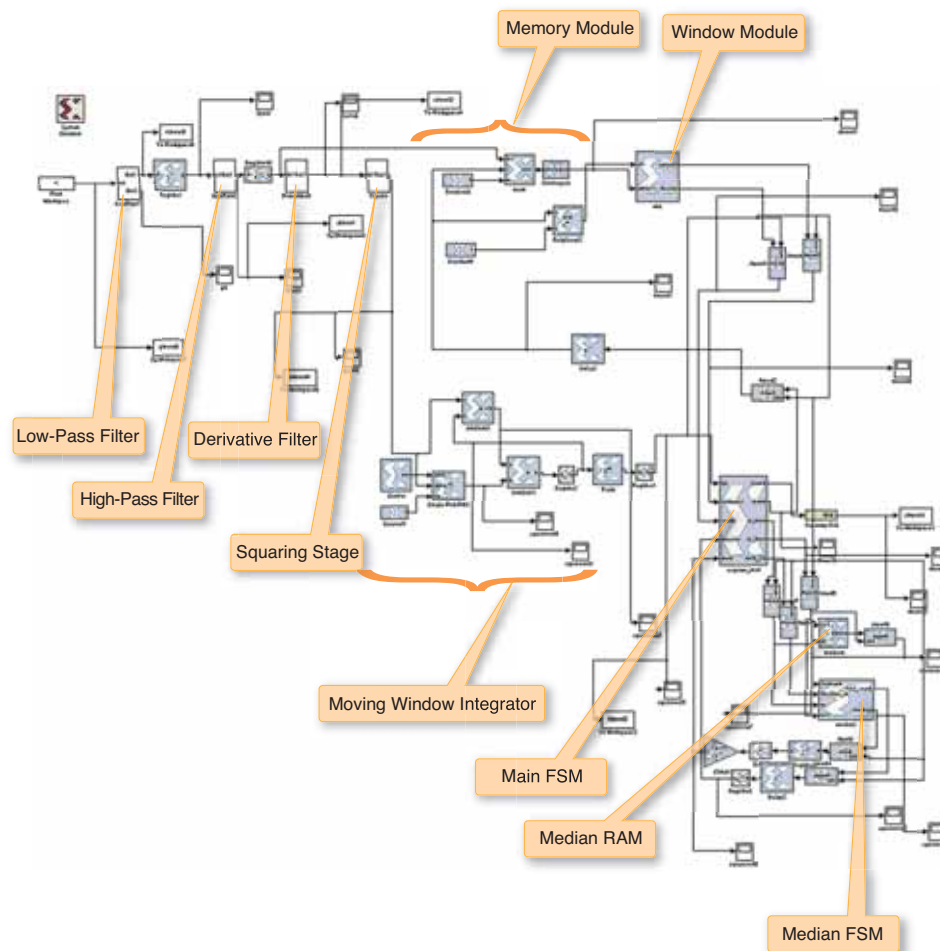


Figure 3 – Complete design in the System Generator environment

The design implemented in the FPGA processes the data and sends the output back to the MATLAB environment so that we can compare it to the simulation output. This feature was very convenient and helped us establish the accuracy of the design implementation in hardware. It also allowed us to run tests on complete sets of benchmark ECG data (from www.physionet.org) in a fraction of the time we previously needed to test the software version of the algorithm.

Overall, we found System Generator to be very useful for implementing designs for biomedical instrumentation applications. The tool provides easy filter design options and simplifies design implementation by accepting imported custom modules as black boxes. It also provides a fast and efficient way to test a design on hardware, with easy interfacing options between the board and the user's workstation. 🌈

Citations

- [1] B-U. Kohler, C. Hennig, and R. Orglmeister, "The principles of software QRS detection," *IEEE Engineering in Medicine and Biology Magazine*, vol. 21, no. 1, pp. 42-57, January 2002.
- [2] A. Shukla and L. Macchiarulo, "FPGA based ECG Analysis System," *Proceedings of the Sixth IASTED International Conference on Biomedical Engineering*, Austria, pp. 68-72, February 2008.
- [3] A. Shukla, "Hardware Implementation of real time ECG analysis algorithms," M.S. thesis, University of Hawaii at Manoa, Honolulu, Hawaii, 2008.
- [4] J. Pan and W. Tompkins, "A real-time QRS detection algorithm," *IEEE Transactions on Biomedical Engineering*, vol. BME-32, no.3, pp. 230-236, March 1983.
- [5] P. Hamilton and W. Tompkins, "Quantitative Investigation of QRS Detection Rules Using MIT/BIH Arrhythmia Database," *IEEE Transactions on Biomedical Engineering*, vol. BME-33, no.12, pp. 1157-1165, December 1986.