Third Prize

FPGA-Based Clinical Diagnostic System using Pipelined Architectures in the Nios II Soft-Core Processor

Institution: **Jadavpur University, Calcutta**

Participants: **Shubhajit Roy Chowdhury**

Instructor: **Professor Hiranmay Saha**

Design Introduction

Clinical decision making is a complicated process. It is based on medical knowledge derived from medical books and literature and on data obtained from various clinical trials and diagnostic tests; however, it is also dependent on experience, judgment, and reasoning, which are functions of the human brain. In many situations, human decision making is not available, and in these situations, instruments play a major role in helping reduce human suffering.

In third-world countries, very few doctors are available in rural areas. For example, in India, 75% of qualified consulting doctors are in urban areas and 23% are in semi-urban areas, which leaves only 2% in rural areas where, unfortunately, nearly 78% of Indians reside. This imbalance has created a patient-doctor ratio of more than 10,000 patients for each doctor in rural India. Therefore, equipment that can predict imminent health hazards and can red-alert patients to contact a doctor for necessary care is urgently needed. Each doctor must handle a large number of patients; therefore, it would be useful for a doctor to be able to track patient data, especially because data and document preservation, such as investigation reports, is poor in rural areas. It would be useful to have a system that can be used effectively for a variety of chronic disease conditions such as renal dystrophy and diabetes mellitus. These types of diagnostic decision making can be performed with fuzzy logic. Initiated by Zadeh in 1965, fuzzy logic and fuzzy set theory are being used more and more in medical expert system applications.

The current research focuses on an FPGA-based smart processing system that can predict the patient's physiological state given the patient's past physiological data. The scheme can provide an alarm to the

relevant personnel, who would contact a physician at a remote site before the patient reaches a critical state. The physician would take the necessary actions to provide medical support to the patient. The smart processing system consists of blocks for fuzzification, inferencing, and defuzzification of patient data. It can handle patients' peripheral health screening, and help caregivers focus on the few critical patients who really need a physician's clinical assistance.

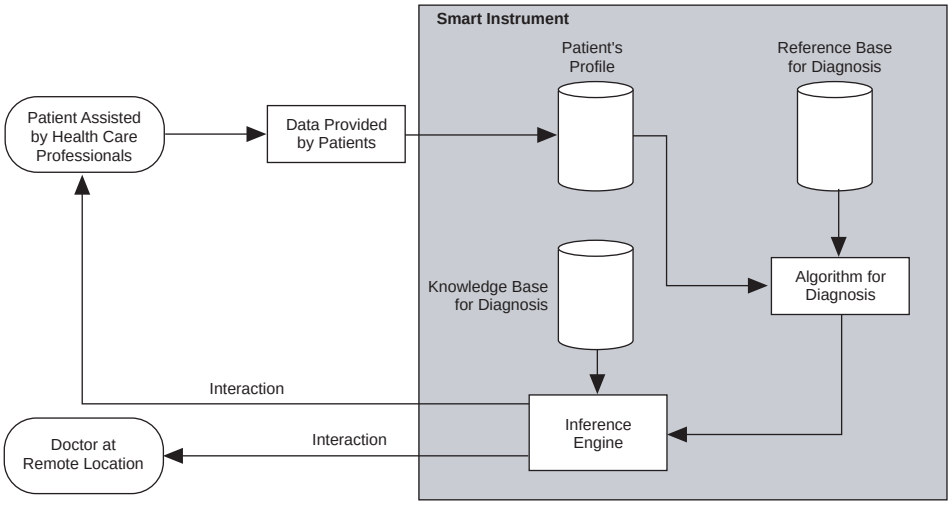
To reduce the combinational logic blocks required to implement the system, we implemented the division process required for normalizing membership functions using intelligent multiplication techniques. To make the computing system fast, we used pipelined data processing architectures. An FPGA implementation is useful in developing countries because of the low investment required compared to ASIC prototyping costs. Additionally, FPGAs are reprogrammable, which allows design improvements. This feature is important because it supports new structures, e.g., upgrading the current smart diagnostic system, supporting other diseases, mapping to other fields of human expertise, etc.

With the Nios® II soft-core processor, we can overcome design issues such as limited peripheral resources, difficult I/O configuration, complex hardware design, and software programming. This design also meets time sequence and function requirements, optimally uses the processor's resources, and greatly improves the overall system efficiency. Because the system has external memory and I/O, memory access is frequent. With the Nios II processor's user-defined peripherals, user-defined logic, and direct memory access (DMA), the design can easily access memory and move data when using SDRAM, SRAM, and flash memory. In our design, the patient profile is stored in flash memory. By combining the requirements of both software and hardware in a coordinated development process, Altera's SOPC solution is the best choice: it can fully showcase the advantages of an FPGA's logic control and data processing capabilities. This design approach allows for flexible system configuration, provides simple, convenient development, supports various processing modes, and offers powerful data processing capacity at low cost.

Function Description

The designed system has a pipelined smart processing unit that can predict the patient's future pathophysiological state based on past pathophysiological data. Figure 1 shows the functional architecture of a diagnostic system that includes a smart agent that we plan to implement.

Figure 1. Smart Agent Based Diagnostic System Functional Architecture



In Figure 1, the smart agent is represented by a fuzzy system. At least three entities are required in this concept of diagnostics:

- The healthcare personnel provide data by measuring the patients' health parameters.
- The physician interacts with the smart instrument and confirms or denies the diagnosis made by the smart instrument.
- The smart instrument performs the diagnosis at regular times and predicts future states of the patient using fuzzy logic.

Based on previously fed data, the smart instrument can give an early signal of deterioration in the patient's health status and indicate an imminent emergency situation. Initially the data provided by the patients under the assistance of health care professionals is stored in a patients' profile. The data from the patients' profile is subjected to a diagnostic process using a knowledge base for diagnosis. The diagnosis process is based on fuzzification of patient data. The inference engine makes a prediction about the future physiological state of the patient based on the fuzzified data. Based on the prediction, the smart system gives an indication about the possible next physiological state of the patient.

The smart processor we developed can fuzzify and defuzzify patient data. The patient data cannot always be trusted because it relies on the quality and accuracy of measuring units and the technician's skill. Moreover, based on a single bit of data, it would be highly difficult to make an accurate decision about the future pathophysiological state of the patient, particularly in a chronic case. Therefore, we fuzzified the patient data to transform periodic measures into likelihoods that the pathophysiological parameter of the patient is high, low, or moderate compared to a reference value set.

As an example study, this project analyzes patient renal data and predicts the patient's future physiological state. The system calculates the patient's body mass index (BMI) using the patient's height (in feet) and weight (in kilograms). Because doctors are more interested in knowing whether the pathophysiological risk parameters of a patient is high, moderate, or low as well as the patient's physiological parameter trends, it is more useful to represent the patient's pathophysiological risk parameters as linguistic variables instead of ordinary variables. Then, we can use fuzzy logic to build a model that predicts the fuzzy set (low, moderate, or high) in which the patient's particular risk parameter (e.g., B.M.I, glucose, urea, creatinine, and blood pressure) lies to be referenced at the next reading of that patient data. For this purpose, we used triangular and trapezoidal fuzzy operators. A typical triangular function takes the form:

$$A(x; a, m, b) = \max\{\min[(x - a)/(m - a), (b - x)/(b - m)], 0\}$$

Similarly, a typical trapezoidal function takes the form:

$$A(x; a, m, n, b) = \max\{\min[(x - a)/(m - a), 1, (b - x)/(b - n)], 0\}$$

We determined the membership function in accordance with the ranges and tolerance limits set up by the World Health Organization. Figure 2 shows the plot of the membership functions defined above.

Figure 2. Plots of the Membership Functions

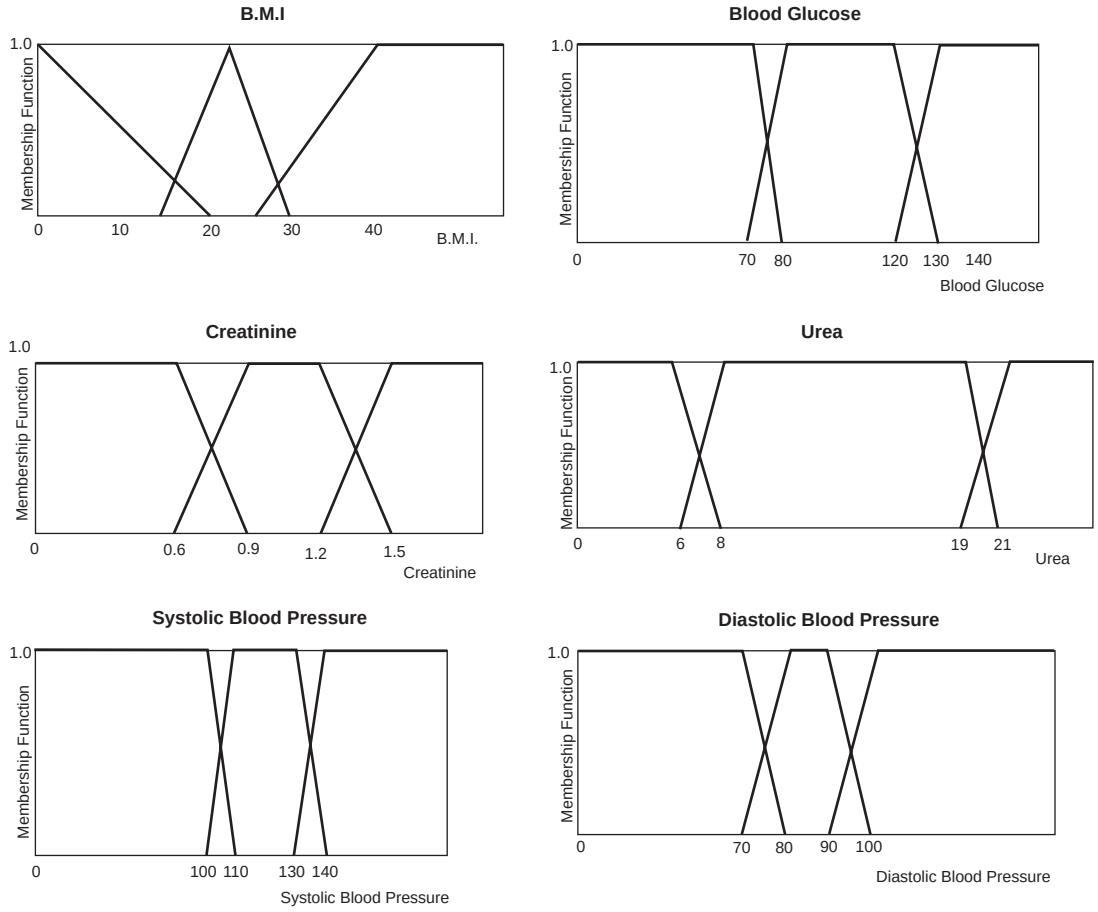


Figure 2 depicts the cognitive frames used for fuzzy modeling patients' BMI, blood glucose, blood urea, blood creatinine, systolic, and diastolic blood pressure data. It is obvious that all low, moderate, and high risk parameter ranges (modeled here as fuzzy sets) of the patients fall in the same universe of risk parameter values.

Inferencing involves deciding whether the patient is in a normal condition, is heading towards a moderately critical condition, or is in a severely critical condition. Inferencing is done by taking the diagnostic algorithm's next possible state output at different points in time. Typical rules for inferencing take the following forms:

- *R1*—If (BMI is high) and (glucose is high) and (urea is high) and (creatinine is high) and (systolic blood pressure is high) and (diastolic blood pressure is high) then the (patient's renal condition is severe).
- *R2*—If (BMI is high) or (glucose is high) or (urea is high) or (creatinine is high) or (systolic blood pressure is high) or (diastolic blood pressure is high) then the (patient's renal condition is moderately critical).
- *R3*—If (glucose is high at time Ti) and (glucose is low at time Tj) and (Ti ≠ Tj) then the (patient should go for glycosylated hemoglobin).

- *R4*—If (BMI is moderate) or (glucose is moderate) or (urea is moderate) or (creatinine is moderate) or (systolic blood pressure is moderate) or (diastolic blood pressure is moderate) then the (patient's renal condition is normal).

and so on.

Defuzzification involves taking a crisp action based on the inference drawn, and can be implemented by illuminating an LED or by outputting data. In our scheme, LEDs indicate that the patient's state is approaching criticality.

The system has two rule bases. The knowledge base contains rules for inferencing based on the patient data set currently received and the already stored patient data. The reference base contains the reference values for fuzzifying the patient data. Based on these values, the system computes the membership function values of low, moderate, and high for the patients'pathophysiological parameters at different points in time. The diagnostic algorithm uses these values to compute the possibility that the next pathophysiological data will be low, moderate, or high.

The diagnosis algorithm computes the time-weighted mean of the membership functions of the patient's pathophysiological data. The possibility that the next pathophysiological data will be low, moderate, or high is computed as:

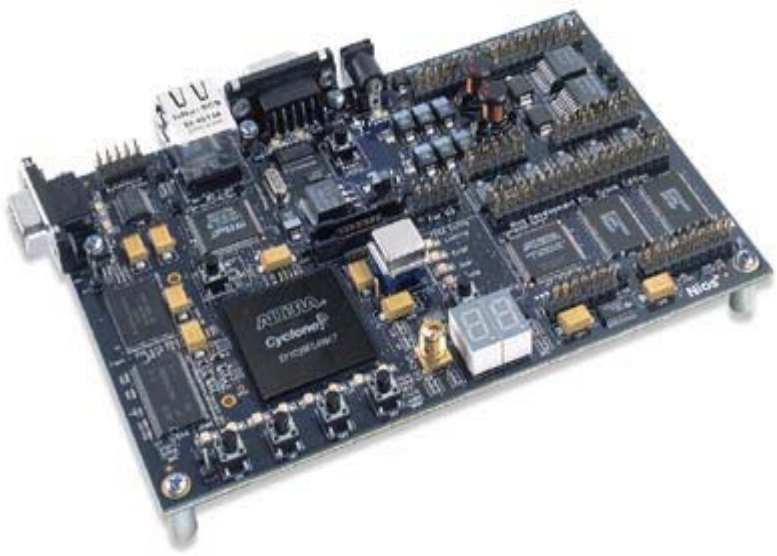
$$P_R(x) = \frac{\sum_{i=1}^{n} i\mu(x)}{\sum_{i=1}^{n} i}$$

where the summation is done from i = 1 to n, and the value of n is the sequence number of the time instant at which the current pathological data of the patient is taken and $R \in \{\text{low, moderate, high}\}$. $\mu(x)$ is $\mu l(x)$ $\mu m(x)$, or $\mu h(x)$ acordingly as the membership function refers to a low, moderate, or high fuzzy set, respectively. To predict the fuzzy set in which the next state input of a certain pathophysiological parameter will lie, the value of $P(x)$ is considered for which $P(x) \geq P_R(x)$.

We implemented the smart data processing system on an Altera® Cyclone® EP1C6Q240C8 FPGA. We could also have implemented the system using software; however, this solution would require a powerful computer to run the software at reasonable speed and accuracy. A powerful computer is too costly and would require a steady supply of electricity in rural sectors. The cost would be an impediment to adoption of the smart diagnostic system in the rural health care centers in third-world countries. Additionally, a software-only solution could take longer to process if we constructed more complex systems covering many different infected parts of the human body, However, a few milliseconds delay cannot be considered important for medical diagnosis. The main reason for a hardware-based implementation is the need for an inexpensive, portable diagnostic system. The main disadvantages of an ASIC-based solution is the high development cost and the low reconfigurability. An FPGA solution ensures that new changes in the proposed diagnostic algorithm can be mapped onto the hardware without having to make costly changes.

Figure 3 shows the UP3 board on which the FPGA is mounted.

Figure 3. UP3 Board



We generated an SRAM Object File (**.sof**), which is a bitstream pattern, using the system's VHDL model and downloaded it to the FPGA via the JTAG interface and ByteBlaster II cable. The configuration data is stored in the FPGA's SRAM.

The input is sent to the FPGA using push-button switches. We need 12 push-button switches but the UP3 board does not have that many. Therefore, we developed our own printed circuit board (PCB) containing the required push-button switches and connected it to the UP3 board. The FPGA receives a 0 input when the user presses a switch. The binary data entered using the switches are converted into real numbers for computation using conversion weights stored in a look-up table (LUT) implemented on the SDRAM.

Using these parameter values, the system computes the corresponding membership function values μ_l, μ_m, and μ_h. These values refer to whether the pathological parameter value is in the low, moderate, or high fuzzy set, respectively. The values are stored in the CMOS flash memory using the Nios II soft-core processor. Based on these values, the system computes the possibility that the values of the different physiological parameters are low, moderate, or high. The maximum of these three possibilities at any instant suggests the patient's next possible physiological state.

The output therapeutic decision is displayed on UP3 board's 7-segment display. The 7-segment display indicates whether the pathological parameters will be low, moderate, or high values. Because, there are four 7-segment displays in the final system and the board only has port available, we used a 4-bit output called SCAN (0 to 3). The SCAN's bit lines are connected to the cathodes ofthe LED 7 segment display, which selects the 7-segment LED in time-shared mode. The display codes are stored in a LUT.

The user can reset the system at any time using a push-button switch. Two LEDs connected in common anode mode indicate whether the patient's condition is moderately critical (MC) or severely critical (SC). The system has a battery back-up that provides a continuous power supply to overcome the FPGA volatility.

The FPGA system implementation is very attractive because FPGAs are reconfigurable and becoming more economical and faster as time goes on. We tested the FPGA implementation with a patient to compare the decision result of the physician vs the smart agent.

To test the system, we analyzed data from a 5-foot tall, 42-year-old patient. Tables 1 through 6 show the results. In the tables, AS refers to the actual physiological state of the patient. In the actual experiment, the patient's weight (Wt), glucose, urea, creatinine, systolic, and diastolic blood pressure data taken at 10-day intervals are input to the system at time Ti (where i = 0,1,…, 9). Initially, the height of the patient is also given. Based on the height and weight data, the system computes the patient's BMI at different times. Using these parameters, the system, computes the corresponding membership function values μ_l, μ_m, and μ_h.

In the AS clumn, M indicates a moderate risk, H* indicates high risk that still falls within the tolerance limits of moderate value, and H indicates strictly high risk.

Table 1. BMI Data Results

Time	Weight	BMI	μ_l	μ_m	μ_h	P_l	P_m	P_h	AS	PNS
T1	64.1	27.97	0.00	0.29	0.14	0.00	0.29	0.14	M	M
T2	66.2	28.31	0.00	0.24	0.16	0.00	0.26	0.15	H*	M
T3	66.8	28.57	0.00	0.20	0.18	0.00	0.23	0.17	H*	M
T4	67.5	28.87	0.00	0.16	0.21	0.00	0.20	0.18	H*	M
T5	66.9	28.61	0.00	0.19	0.19	0.00	0.18	0.17	H*	M
T6	67.8	29.00	0.00	0.14	0.21	0.00	0.14	0.18	H*	H
T7	68.2	29.17	0.00	0.12	0.23	0.00	0.11	0.20	H*	H
T8	69.5	29.73	0.00	0.04	0.27	0.00	0.09	0.22	H*	H
T9	70.5	30.15	0.00	0.00	0.29	0.00	0.07	0.24	H	H
T10	70.6	30.62	0.00	0.00	0.33	0.00	0.06	0.25	H	H

Table 2. Glucose Data Results

Time	Glucose	μ_l	μ_m	μ_h	P_l	P_m	P_h	AS	PNS
T1	120	0.00	1.00	0.00	0.00	1.00	0.00	M	M
T2	125	0.00	0.50	0.50	0.00	0.67	0.33	H*	M
T3	128	0.00	0.20	0.80	0.00	0.70	0.30	H*	M
T4	127	0.00	0.30	0.70	0.00	0.54	0.46	H*	M
T5	128	0.00	0.20	0.80	0.00	0.33	0.57	H*	H
T6	128	0.00	0.20	0.80	0.00	0.29	0.71	H*	H
T7	128	0.00	0.20	0.80	0.00	0.26	0.74	H*	H
T8	129	0.00	0.10	0.90	0.00	0.23	0.77	H*	H
T9	129	0.00	0.10	0.90	0.00	0.20	0.80	H*	H
T10	131	0.00	0.00	1.00	0.00	0.16	0.84	H	H

Table 3. Urea Data Results

Time	Urea	μ_l	μ_m	μ_h	P_l	P_m	P_h	AS	PNS
T1	17.0	0.00	1.00	0.00	0.00	1.00	0.00	M	M
T2	17.5	0.00	1.00	0.00	0.00	1.00	0.00	M	M
T3	18.7	0.00	1.00	0.00	0.00	1.00	0.00	M	M
T4	19.1	0.00	0.95	0.05	0.00	0.98	0.02	H*	M
T5	20.7	0.00	0.15	0.85	0.00	0.70	0.30	H*	M
T6	20.6	0.00	0.20	0.80	0.00	0.56	0.44	H*	M
T7	20.8	0.00	0.10	0.90	0.00	0.44	0.56	H*	H
T8	20.9	0.00	0.05	0.95	0.00	0.36	0.64	H*	H
T9	20.9	0.00	0.05	0.95	0.00	0.29	0.71	H*	H
T10	21.0	0.00	0.00	1.00	0.00	0.24	0.76	H	H

Table 4. Creatinine Data Results

Time	Creatinine	μ_l	μ_m	μ_h	P_l	P_m	P_h	AS	PNS
T1	1.0	0.00	1.00	0.00	0.00	1.00	0.00	M	M
T2	1.1	0.00	1.00	0.00	0.00	1.00	0.00	M	M
T3	1.2	0.00	1.00	0.00	0.00	1.00	0.00	M	M
T4	1.3	0.00	0.67	0.33	0.00	0.87	0.13	H*	M
T5	1.4	0.00	0.33	0.67	0.00	0.69	0.31	H*	M
T6	1.4	0.00	0.33	0.67	0.00	0.59	0.41	H*	M
T7	1.4	0.00	0.33	0.67	0.00	0.52	0.41	H*	M
T8	1.4	0.00	0.33	0.67	0.00	0.48	0.52	H*	H
T9	1.8	0.00	0.00	1.00	0.00	0.38	0.62	H	H
T10	2.4	0.00	0.00	1.00	0.00	0.31	0.69	H	H

Table 5. Systolic Blood Pressure (SBP) Data Results

Time	SBP	μ_l	μ_m	μ_h	P_l	P_m	P_h	AS	PNS
T1	128	0.00	1.00	0.00	0.00	1.00	0.00	M	M
T2	131	0.00	0.90	0.10	0.00	0.93	0.07	H*	M
T3	132	0.00	0.80	0.10	0.00	0.87	0.13	H*	M
T4	136	0.00	0.40	0.20	0.00	0.68	0.13	H*	M
T5	137	0.00	0.30	0.70	0.00	0.55	0.45	H*	M
T6	138	0.00	0.20	0.80	0.00	0.45	0.55	H*	H
T7	139	0.00	0.10	0.90	0.00	0.36	0.64	H*	H
T8	140	0.00	0.00	1.00	0.00	0.28	0.72	H	H
T9	140	0.00	0.00	1.00	0.00	0.23	0.77	H	H
T10	143	0.00	0.00	1.00	0.00	0.18	0.82	H	H

Table 6. Diastolic Blood Pressure (DBP) Data Results

Time	DBP	μ_l	μ_m	μ_h	P_l	P_m	P_h	AS	PNS
T1	87	0.00	1.00	0.00	0.00	1.00	0.00	M	M
T2	88	0.00	1.00	0.00	0.00	1.00	0.00	M	M
T3	90	0.00	1.00	0.00	0.00	1.00	0.00	M	M
T4	94	0.00	0.60	0.40	0.00	0.84	0.16	H*	M
T5	96	0.00	0.40	0.60	0.00	0.69	0.31	H*	M
T6	98	0.00	0.20	0.80	0.00	0.55	0.45	H*	M
T7	98	0.00	0.20	0.80	0.00	0.46	0.54	H*	H
T8	97	0.00	0.30	0.70	0.00	0.42	0.58	H*	H
T9	100	0.00	0.00	1.00	0.00	0.34	0.66	H	H
T10	101	0.00	0.00	1.00	0.00	0.28	0.72	H	H

Based on these membership function values, the possibilities the values of the different pathophysiological risk parameters will be low, moderate or high has been computed by the system (see Table 7). The AS and PNS subscripts are the first letter of the corresponding risk parameter. For example, AS_B refers to the patient's state based on BMI data.

Table 7. Smart Agent Decision Results

Time	AS_B	PNS_B	AS_G	PNS_G	AS_U	PNS_U	AS_C	PNS_C	AS_D	PNS_D	AS_S	PNS_S	SC
T1	M	M	M	M	M	M	M	M	M	M	M	M	0
T2	H*	M	M	M	M	M	M	M	M	M	H*	M	0
T3	H*	M	H*	M	M	M	M	M	H*	M	H*	M	0
T4	H*	M	H*	H	H*	M	H*	M	H*	M	H*	M	0
T5	H*	H	H*	H	H*	M	H*	M	H*	M	H*	M	0
T6	H*	H	H*	H	H*	M	H*	M	H*	M	H*	H	0
T7	H*	H	H*	H	H*	H	H*	H	H*	H	H*	H	1
T8	H*	H	H*	H	H*	H	H*	H	H*	H	H	H	1
T9	H	H	H*	H	H*	H	H	H	H	H	H	H	1
T10	H	H	H	H	H	H	H	H	H	H	H	H	1

Although the system gives a crisp decision regarding the patient's future pathophysiological state, the point is that the system predicts a critical state at time T7, which is well before a clinically overt criticality occurs at time T9. The system can thus be deployed in a variety of telediagnostic environments, in which health care professionals provide support services without a doctor present.

Performance Parameters

Our system has the following performance parameters:

- *Power supply*—DC voltage 5 V and the operating current is 175 mA.
- *Operating temperature*—5^{o} to 45^{o} C and relative humidity or 8% to 95%.
- *Data input*—Data is input via push-button switches. The input data is converted into real numbers for the ease of computation.
- *Data output*—The output data is displayed on the 7-segment display. A patient's critical state is signaled by a glowing LED.
- *Storage*—The patient data is stored in a TC58FVB160AFT CMOS flash memory device.

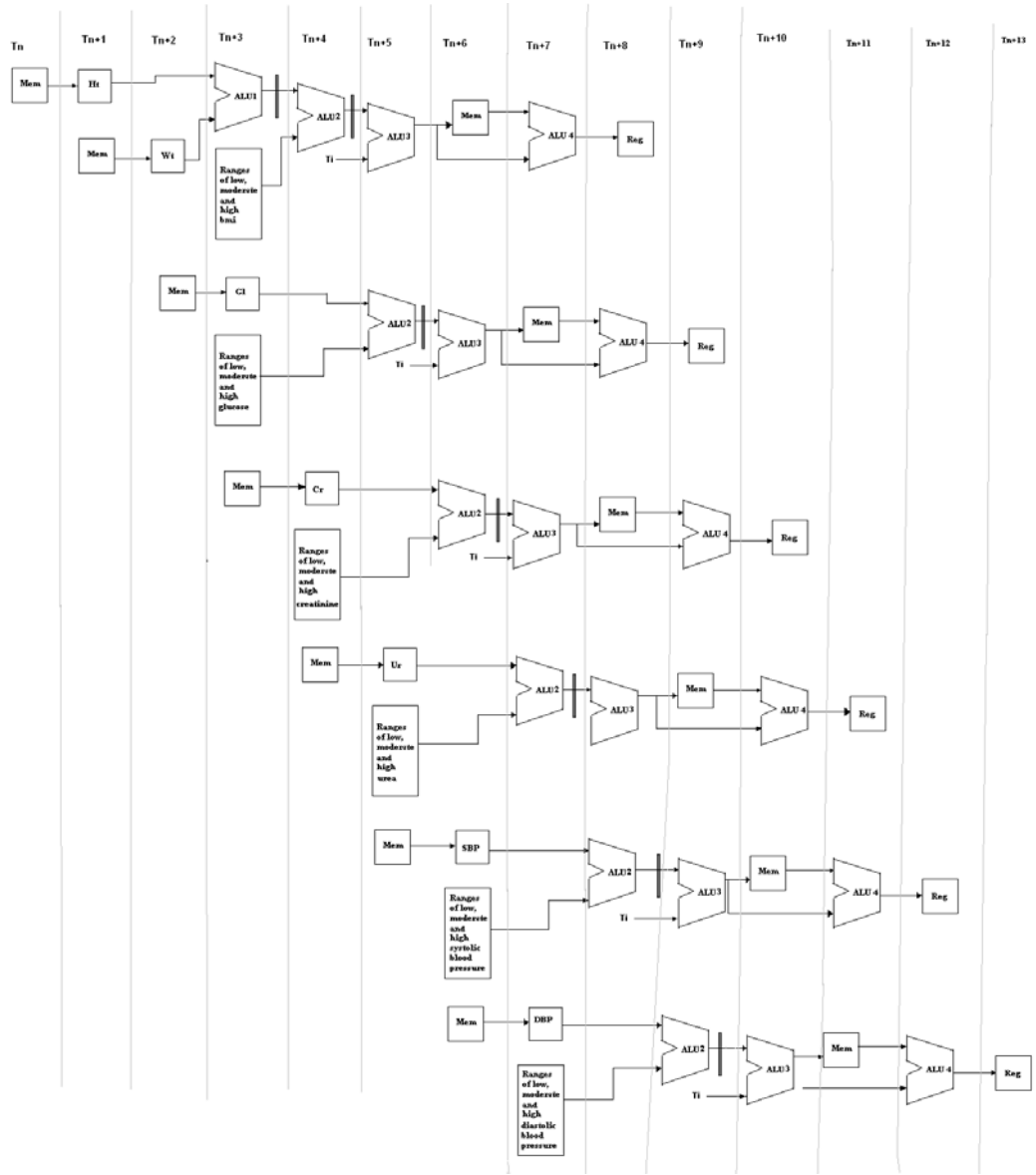
We used the Nios II processor to manage the FPGA's internal resources, define the time sequence requirements for data processing, and handle display and control of the system. Additionally, we needed to access multiple peripherals frequently from the main system, and the Nios II processor helped us to improve the system's overall operational efficiency. The Nios II functions are descibed below:

- It is the main processor, and controls the whole system logic.
- It handles the transfer of patient data between FPGA and CMOS flash memory.
- It handles all instructions to the FPGA's internal logic through user-defined programmable I/O (PIO) peripherals.

Design Architecture

For fast computation, we implemented a finite state machine with pipelined data processing on the FPGA. Figure 4 shows the pipelined architecture.

Figure 4. Pipelined Architecture Implemented in FPGA



The system has four arithmetic logic units (ALUs):

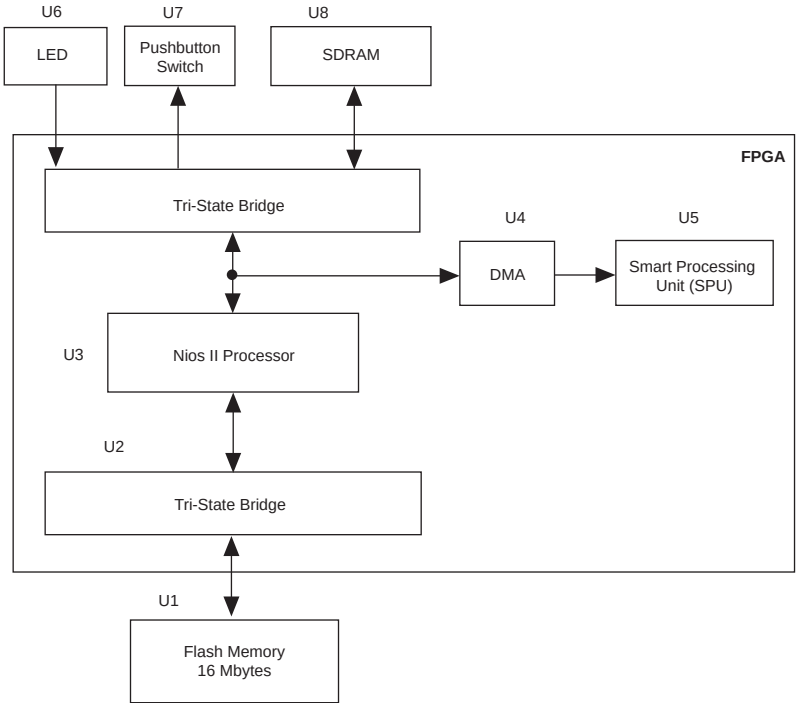
- ALU1 computes the BMI using height and weight data.
- ALU2 computes the membership function values using the instantaneous values of the pathophysiological parameters.

- ALU3 computes the probability that low, moderate, or high pathophysiological parameters will occur.
- ALU4 decides whether the next pathophysiological parameter reading is low, moderate, or high.

With a pipelined architecture, the system can compute the decision for all pathophysiological parameters in 14 clock cycles vs. the 44 clock cycles required in an unpipelined architecture. This difference increases performance about 3.14 times.

Figure 5 shows the system architecture including the Nios II processor.

Figure 5. Medical Diagnostic System Architecture Co-Design



The design's main modules are:

- *U1*—2-Mbyte flash memory containing the patient data.
- *U2*—Tri-state bridge.
- *U3*—Nios II processor.
- *U4*—DMA controller configured to feed the smart processing unit (U5) with patient data.
- *U5*—Smart processing unit based on the pipelined architecture discussed previously.
- *U6*—Four 7-segment displays and two LEDs for displaying the output results.
- *U7*—11 push-button switches for entering data.
- *U8*—SDRAM LUT that stores the display codes and conversion weights.

The hardware/software co-design involves the following steps:

1. Develop the system algorithm.
2. Implement the hardware peripherals using hardware description languages.
3. Verify the hardware peripherals' functionality using the ModelSim software.
4. Synthesize the peripherals using the Leonardo Spectrum synthesis tool.
5. Perform layout and timing analysis of the hardware peripherals using the Quartus II software.
6. Implement the algorithm in the Nios II processor.
7. Use the Nios II Integrated Development Environment (IDE) to connect the Nios II processor with the hardware peripherals.
8. Build and load the Quartus® II project onto the FPGA.

Figure 6 shows the data processing software flow chart.

Figure 6. Software Algorithm Flow Chart

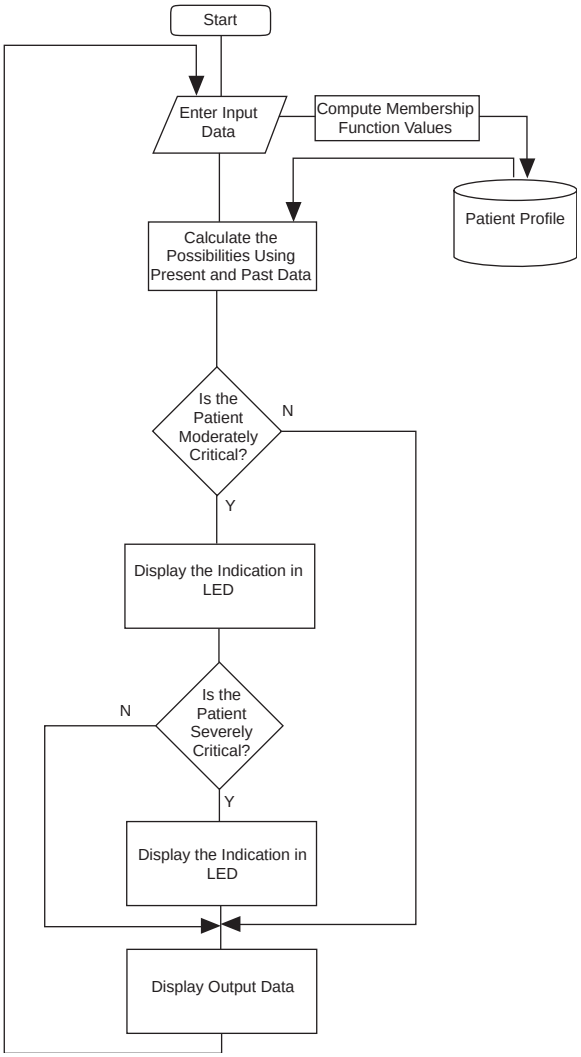
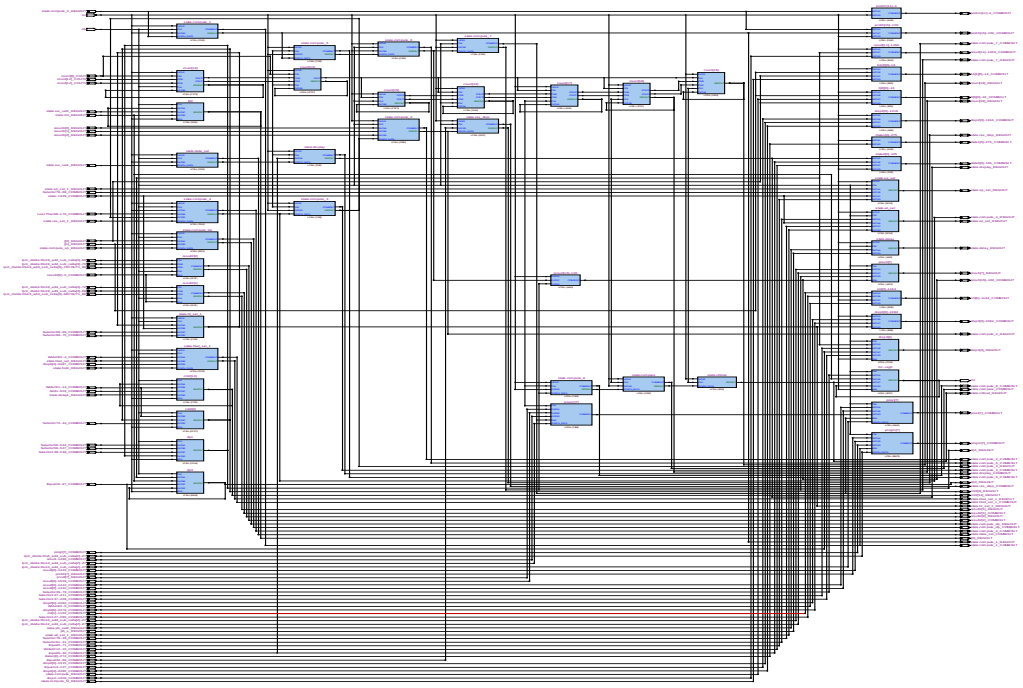


Figure 7 shows the smart agent's schematic.

Figure 7. FPGA-Based Smart Agent Schematic



Design Description

Using Altera's UP3 development board, we were able to design most of the system functions, and test and simulate all functions. Additionally, we designed a few circuit boards for design and test. We performed our system design and testing using the following steps:

1. Used SOPC Builder to access peripheral storage on the UP3 development board.

2. Referred to UP3 board examples and testing documents, and learned about the Nios II architecture, C language software programming in the IDE, and online programming and debugging for the device.

3. Implemented user-defined peripherals and logic on UP3 development board.

4. Implemented debouncing of data entered via the push-button switches.

Hardware Implementation

Our hardware design task was to combine the UP3 board testing methods and our circuits, and then design the necessary hardware modules to develop a medical diagnostic system that implemented the required functions on the Nios II processor. Using the PROTEL 99SE tool, we partitioned the design using the UP3 development board and our own circuit modules. We designed all functional system modules based on the FPGA, which represents a system-on-a-programmable-chip (SOPC) design concepts in hardware. Because the Nios II processor is already available, we simply needed to configure peripherals such as flash devices. Other system peripherals include a power management unit, LED interface, and push-button switches.

To make it easy to understand the system hardware design process, we split the design description into the following sections:

- *Schematic diagram design*—Because we completed most of the functional testing and simulation on the UP3 development board and self-designed circuits, the schematic diagram mainly refers to designs that combine these elements into the schematic most representative of the circuit system. With Altera's SOPC solution, we integrated all processors and functional control units into the FPGA, further simplifying the design structure.

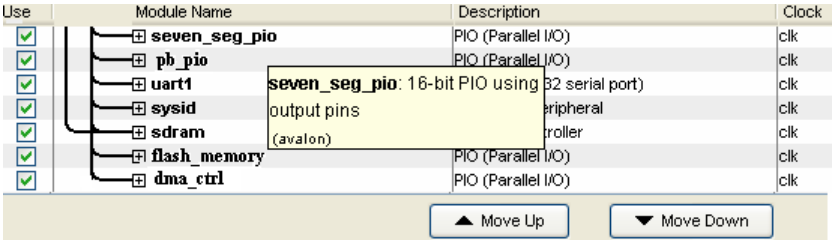
- *Schematic diagram functional verification*—Although most of the functional testing was completed on the test board, we needed to functionally verify the hardware integration. The schematic diagram functional verification primarily demonstrates the proof of concept. This process confirmed our complete circuit design.

- *Component purchase*—We purchased the necessary components, such as the 7-segment displays and push-button switches, while designing the schematic diagram and verifying it. All component packages were clearly marked on the schematic, which made PCB development easier.

- *Implementing the LUTs*— Our design requires two LUTs. One LUT stores the conversion weights that convert the binary data entered using the push-button switches to integers and real numbers for data computation purposes. The other LUT stores the display codes for displaying data in the 7-segment displays.

We added peripherals using SOPC Builder (see Figure 8).

Figure 8. Adding Peripherals Using SOPC Builder



Software Implementation

Our software design task was to migrate the VHDL and C language programs of the previously described functions (on the UP3 development board and self-designed circuits) to the Cyclone EP1C6Q240C8 FPGA with a Nios II soft-core processor. We based the software module design on highly integrated hardware modules so that we could complete core modification and perform upgrades. We used the Altera Quartus II software, SOPC Builder, and Nios II IDE to build the Nios II processor and to develop the system control program, highlighting the SOPC solution's highly integrated and programmable concepts. The design made it possible to implement, add, and remove multiple peripherals easily, access user-defined peripherals, and design user-defined instructions. Generally, the Nios II processor controls the system. However, we implemented most of the software modules based on cooperation between the Nios II processor and the FPGA logic.

We used the VHDL and C languages for the software design. We wrote the logic control and data processing program in VHDL, and used C language routines for the control program of the main and sub Nios II processors. Based on the functional tasks, the system software is divided into system initialization, data acquisition, data display, and patient state prediction. The implementation method and steps are as follows:

- *System initialization*—The system is initialized by the Nios II processor via the tri-state bus, which includes user buttons, 7-segment displays, and the flash memory.

- *Data acquisition*—The patient data is acquired through push-button switches. The system receives a binary 0 input when each switch is pressed. The binary data entered through the push-button switch array is converted into real numbers for computation using conversion weights stored in a LUT.

- *Data display*— The output therapeutic decision is displayed on 7-segment displays. The displays indicate the possibilities of low, moderate,and high values of the different pathological parameters at the patient's next physiological state. Because there are four 7-segment displays for output in the final system and there is only one port available for display, we used a 4-bit output called SCAN (0 to 3). The display codes corresponding to the 7-segment display are stored in ROM.

- *Patient state prediction*—The patient state is predicted by the smart agent implemented on the FPGA and interfaced with the Nios II processor. The smart agent's operation logic is discussed in "Function Description" on page 374.

Design Features

Using Altera's SOPC solution, we learned a new way to solve system design problems. By creating an SOPC design, we learned its advantages and disadvantages. Because SOPC designs use multiple IP modules to optimize the hardware design, we could simplify the system revision and debug process. This approach also allowed us to design software and hardware modules simultaneously. In this design contest, we acquired hands-on experience and were able to use some excellent hardware development tools. The Quartus II software and SOPC Builder made it easy for us to modify and change hardware, depending on the application. We also think that this design approach is economical—you do not need to buy additional hardware, you just change the Nios II configuration. We can easily build new functions by adding related hardware based on the changed Nios II processor.

The main features of our design are:

- *Portability*—Because the requirements of the medical diagnostic system software may change from time to time, we can design the system such that it can port to different hardware configurations. The Nios II processor provides this flexibility.

- *Power consumption*—The system is applicable for rural health care centers where power sources may not be available; therefore, the system should consume very low power. The whole system consumes as little as 60.00 mW.

- *Ease of operation*—The system is designed and developed to be operable easily so that it can be handled by health care professionals. Moreover, it can give an indication about the future pathophysiological state of a patient in the absence of the physician.

- *Integration*—The Nios II processor makes it possible to integrate the FPGA with the peripherals.

Conclusion

During Altera's 2007 Nios II processor competition, our design group divided the design tasks into system integration, hardware development, and software design.

- *System intregration*—The convenience of the Nios II IDE and the SOPC Builder tools gave us the flexibility to implement the design quickly on a prototype machine, which accelerated the development process. In this competition, we learned the process of consumer-electronics product development. The SOPC design approach reduces the cost of manpower and material resources during development. Therefore, we believe that this design approach will become popular in the future. Although we did not add many components, this competition made us appreciate the potential system integration capabilities. Additionally, we hope that Altera can provide a variety of demonstration board programs that will help interested students quickly grasp the FPGA design development process.

- *Hardware development*—In this competition, we used a top-down design approach and planned the complete hardware design in the beginning. Therefore, we needed to establish data stream rules at the start of the planning process. These rules eliminated problems during the design stage, allowing us to complete the project on time. Teamwork was an integral part of this contest. Although the Quartus II tool was easy and flexible to use, there were design issues that required experience; for example, using different frequencies while accessing the SDRAM. This competition gave us an important opportunity to learn about teamwork and problem-solving, understand system development, and resolve challenging design questions.

- *Software design*—We developed the necessary software interface, focusing on implementing the smart agent on the FPGA and interfacing peripherals with the Nios II processor. We used the SOPC Builder C++ tool to create the software design for the Window's interface. We also used it as a verification tool and performed Nios II communication debugging for the phase test. We hope to learn more about SOPC design in the future!