An FPGA-Based Cloud System for Massive ECG Data Analysis

Xu Wang, *Student Member, IEEE,* Yongxin Zhu, *Senior Member, IEEE,* Yajun Ha, *Senior Member, IEEE,* Meikang Qiu, *Senior Member, IEEE,* and Tian Huang, *Student Member, IEEE*

***Abstract*—In this brief, we propose a stand-alone SOPC (System on a Programmable Chip) based cloud system to accelerate massive ECG data analysis. The proposed system tightly couples network IO handling hardware to data processing pipelines in a single FPGA, offloading both networking operations and ECG data analysis. In this system, we first propose a massive-sessions optimized TCP/IP hardware stack using a macro-pipeline architecture to accelerate network packet processing. Second, we propose a streaming architecture to accelerate ECG signal processing, including QRS detection, feature extraction and classification. We verify our design on XC6VLX550T FPGA using real ECG data. Compared to commercial servers, our system shows up to 38X improvement in performance and 142X improvement in energy efficiency.**

***Index Terms*—Cloud computing, ECG data analysis, FPGA.**

I. Introduction

Cardiovascular disease is still one of the most deadly disease in the world. Therefore, real-time electrocardiogram (ECG) monitoring and diagnosis are of great significance in early detection, prevention and treatment of heart diseases. There are two research directions in building ECG analysis systems. One is the portable solution based on wearable sensor technology and low-power specialized hardware [1] [2]. However, these portable systems are primarily aimed for real-time ECG monitoring, which is hard to deliver professional diagnosis that requires a sophisticated real-time ECG data processing procedure. The other is the cloud based ECG tele-monitoring and remote diagnosis that collects personal ECG data from wearable ECG devices and makes ECG analysis in the cloud-side health-care platforms. The cloud based solution has two major benefits. First, moving ECG processing from the terminal side to the cloud side makes wearable ECG devices more simple, low-power and cheap, which is more cost-effective for users. Second, with large amounts of ECG data gathered from different users, it can optimize the ECG analysis models by big data analysis method and eventually provides more accurate diagnosis to users.

Xu Wang, Yongxin Zhu and Tian Huang are with the School of Microelectronics, Shanghai Jiao Tong University, Shanghai, 800 DongChuan Road, China. (E-mail: wang2002xu@gmail.com; zhuyongxin@sjtu.edu.cn; ian_malcolm@sjtu.edu.cn.) They are supported by NSFC 61373032 and 863 Program 2015AA050204.

Yajun Ha is with the Department of Electrical and Computer Engineering, National University of Singapore, 21 Lower Kent Ridge Road, Singapore. (E-mail:elehy@nus.edu.sg)

Meikang Qiu is with the Department of Computer Science, Pace University, 1 Pace Plaza, Manhattan, New York City, NY 10038, USA. (Email: mqiu@pace.edu) Qiu is supported by NSF 1457506.

Recent study [3] built remote ECG data analysis service on commercial cloud infrastructure with a large number of connected commodity computers to support automated analysis of large patient populations. However, commercial servers are not efficient for this application both in terms of performance and energy efficiency. Since, remote ECG data analysis, like many other services on the cloud, is a web-based application, which incurs a large number of concurrent requests. Each of the requests involves an individual task which often desires short latencies in response from energy-hungry servers. Thus, for existing commercial servers, the performance is typically limited by overheads of the network packet processing and the connection management in the network interface controller and operating system kernel. Furthermore, the process of ECG analysis is complex and time-consuming, which imposes a heavy computation burden for commercial servers. As a result, network I/O handling and the high complexity of ECG data analysis are the two bottlenecks to have an efficient cloud computing platform for ECG analysis.

To address the issues, we present an FPGA-based cloud system for massive ECG data analysis. Although FPGA is widely used in massive data processing systems [4], little research has been performed to build a stand-alone FPGA based cloud system. Differing from previous work on ECG monitoring hardware [2] [5] that mainly focused on front-end preprocessing of ECG data such as QRS detection, our system is an all-in-one cloud system that tightly couples network IO handling and full ECG diagnosis pipeline in a single FPGA. First, we propose a massive-sessions optimized TCP/IP hardware stack to offload networking operations. We adopt a macro-pipeline architecture with hardware based connection management, where all TCP sessions share a coarse-grained pipeline and centralized memory. Compared to the current third party TCP offload engines (TOE) that are primarily optimized for inter-server data transfer among a few TCP connections in data centers, our TCP/IP hardware stack is mainly optimized for high concurrent connections, supporting up to 100K TCP sessions under 10Gb Ethernet. Second, we propose a streaming architecture to accelerate ECG signal processing and diagnosis, including not only QRS detection, but also feature extraction and classification. Our streaming architecture is optimized for massive concurrent ECG processing and tightly coupled to our TCP/IP hardware stack, which can achieve much higher throughput. We verified our design on Xilinx XC6VLX550T FPGA. Compared to commercial servers, our system shows up to 38X improvement in performance and 142X improvement in energy efficiency.

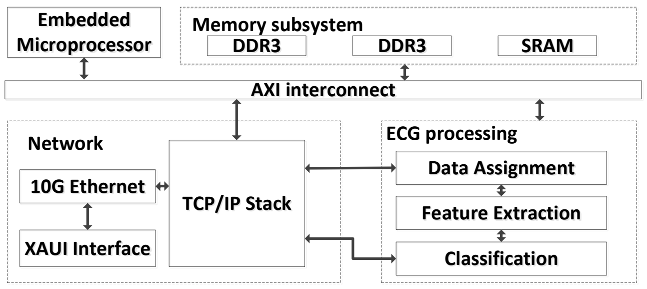

Fig. 1: SOPC based architecture that tightly couples network IO handling and ECG data processing.

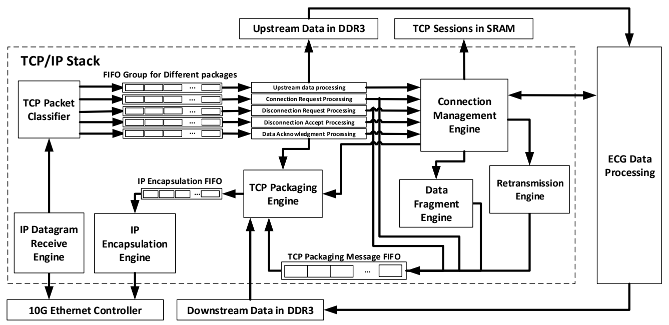

Fig. 2: Micro-architecture of the TCP/IP stack.

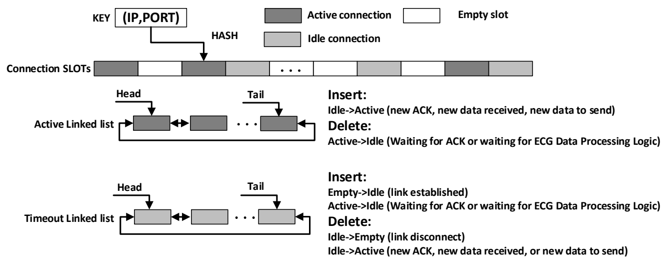
KEY
(IP,PORT)
HASH
Active connection
Idle connection
Empty slot
Connection SLOTs
Head
Tail
Active Linked list
Insert:
Idle->Active (new ACK, new data received, new data to send)
Delete:
Active->Idle (Waiting for ACK or waiting for ECG Data Processing Logic)
Head
Tail
Timeout Linked list
Insert:
Empty->Idle (link established)
Active->Idle (Waiting for ACK or waiting for ECG Data Processing Logic)
Delete:
Idle->Empty (link disconnect)
Idle->Active (new ACK, new data received, or new data to send)

Fig. 3: Memory allocation and scheduling of TCP connections.

II. System Architecture for Massive ECG Data Analysis

To support remote ECG data analysis for massive users concurrently, we propose a SOPC based system architecture (Fig.1) that tightly couples network IO handling and data processing in a single FPGA. It includes a network IO handling subsystem, an ECG data processing subsystem, as well as an embedded microprocessor and a memory subsystem. In the network I/O subsystem, a massive-sessions optimized TCP/IP hardware stack is implemented to support fast network package processing and rapid connection management. Moreover, a highly pipelined ECG data processing accelerator is directly connected to the hardware TCP/IP stack, which respond to massive user requests without any software interaction.

A. *TCP/IP Hardware Stack*

Mainstream TOEs are primarily optimized for massive data volume transmission rather than high concurrent connections to boost their support for inter-server data transfer among a few TCP connections in data centers. They typically adopt one-TCP/IP-session-per-pipeline architecture, where each TCP session contains full TCP/IP pipeline as well as private TX and RX buffer using on-chip SRAMs. However, cloud based ECG analysis is quite a different application scenario in that the TOE needs to maintain more than ten thousands of concurrent connections where each connection only contributes a little portion of total throughput. Thus these third party TOEs no longer satisfy this requirement due to limited on-chip resource to support more TCP sessions. To address the problem, we propose a massive-sessions optimized TOE. We use a macro-pipeline architecture, where all TCP sessions share a coarse-grained pipeline with carefully organized TX, RX buffer. Also, we develop a hardware based connection management to support rapid scheduling of up to 100K TCP sessions.

*1) **Macro-pipeline architecture**:* In order to achieve high system throughput, all processing modules in the macro-pipeline are operated in an asynchronous way, each of which handles a portion of a processing stage or directly responds to a specific type of TCP packets (Fig.2). These processing modules are event triggered, and they continuously read the events or messages from the input FIFO, process the events, and send messages to the next stage without any synchronization. Specifically, for incoming data processing, data from network are first offloaded from 10G Ethernet and unpacked by IP datagram receive engine to obtain TCP packets. Then these TCP packets are divided into different categories based on the control bit in their TCP headers and delivered to different processing engines respectively. Each processing engine deals with a particular type of TCP packet such as SYN packet, FIN packet or ACK packet and directly generates feedback packets to response. The connection management engine deals with the connection and disconnection of TCP links and schedules of the data transmission order among all established links. For outgoing data processing, the data fragment engine segments data that is to be transmitted into small data packages. Then these data packages are encapsulated with TCP header and IP header successively by TCP packaging engine and IP encapsulation engine. When a timeout occurs, the retransmission engine retransmits the lost TCP packets based on the information from the most-recent data ACK number.

*2) **Connection management**:* Due to the limited on-chip memory resources, connection information can only be stored in external memory, while the shared memory may become the bottleneck of the system when it incurs random accesses from different processing modules concurrently. We mitigate the impact of the problem in two ways: to reduce access latency and to reduce the number of memory accesses. We first choose SRAM instead of DRAM to store TCP sessions for its much lower access latency. Then, as described in Fig.3, the memory space of external SRAM is segmented into a number of consecutive TCP connection slots, where we use a hash mechanism with source IP address and port number as the key to fast lookup of a dedicated TCP connection. We avoid using a round-robin scheduling scheme, which is widely used in TOEs that support small number of TCP sessions, to schedule the execution of data transmission among established TCP

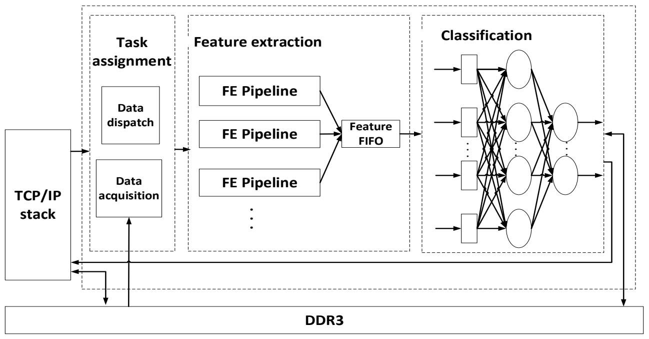

Fig. 4: Architecture of the ECG data processing module.

connections. Because in web-based cloud computing scenario with high concurrent connections, it is time-consuming to check every connection slot to find an active connection. Instead, we maintain two linked list to fast locate active connections and timeout connections. The active linked list is used to store TCP connections that have an event to be processed. The connection management engine continuously obtains the connection in the head, executes the event, then puts the connection to the tail of the timeout linked list. When an event (new ACK received, new data received or new data to send) for certain connection occurs, the connection is inserted into the tail of the active linked list. On the other hand, the timeout linked list is used for monitoring the transmission timeout of idle connections. Idle connections are inserted into the tail of the timeout linked list with a timeout point that adds a fix timeout value to current time. Since the timeout linked list is naturally sorted in timeout point, the retransmission engine only needs to check through the head of the list to obtain the most timeout connections, avoiding recurrently traversing the whole established TCP connection slots that may impose access burdens to the shared memory.

B. ECG Data Processing Module

The ECG data processing module in our work differs from the previous portable ECG monitoring hardware designs [2] [5] in that: (1), we achieve ECG diagnosis more than ECG monitoring, so that our design includes full ECG diagnosis procedures, including not merely QRS complex detection, but also feature extraction, and classification; (2), we consider ECG data analysis for massive concurrent users rather than only one user, so that our design objective is to maximize system throughput within limited on-chip resources.

The architecture of our ECG data processing module is composed of three processing stages: task assignment, feature extraction, and classification (Fig.4). After the TCP/IP stack offloads users' ECG data to the external DDR3 memory, the task assignment module dispatches these ECG data to different feature extraction (FE) pipelines. In the FE pipeline, we first detect the QRS complex of the ECG data. Then we extract several features from the QRS complex as the inputs of the classification stage. We further implement an artificial neural network (ANN) to classify the extracted features. Finally, the diagnostic results are sent back to users by the TCP/IP engine.

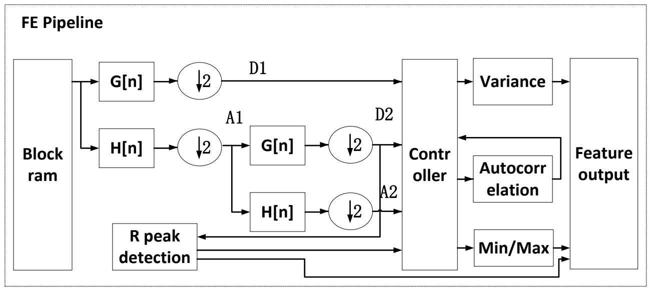

Fig. 5: Architecture of the feature extraction pipeline.

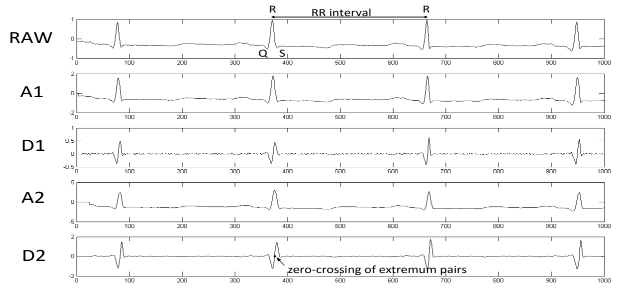

Fig. 6: Wavelet transform of the ECG signals.

*1) **Feature Extraction Pipelines**:* In our work, we first use wavelet transform (WT) method [6] to detect QRS complex, which is the most important and informative segment in ECG to help determine heart conditions. WT decomposes a target signal into an approximate signal and a set of high-frequency wavelets. We implement a two stage WT pipeline, each of which consists of a high-pass filter $G[\cdot]$, a low-pass filter $H[\cdot]$ and two down-samplers by 2 ($\downarrow 2$) (Fig.5). The down-sampled outputs of the first high and low-pass filters provide the detail, $D1$ and the approximation, $A1$, respectively. The approximation $A1$ is further decomposed in the second level to produce the second level approximation A2 and detail D2. As shown in Fig.6, the detection of QRS complex is focused on the detection of R peaks in the raw ECG data. In D2, every special wave where a negative minimum point is followed by a positive maximum point corresponds to each R peak. Thus, we capture the position of R peaks by identifying the zero-crossing of extremum pairs in D2. Then the QRS complex is extracted from the raw ECG data by segmenting a fixed number of samples centered at R peaks.

To support massive ECG data analysis within limited on-chip resources, we do not use all sample points of QRS complex, but rather extract features from the QRS complex as the inputs of the classification, which reduces the size of classification hardware. We use 11 statistics over the set of the wavelet coefficients to represent the time-frequency distribution of the ECG signals, including variance of the original QRS complex, variance of the wavelet coefficients in each subband (D1,D2,A2), variance of autocorrelation functions of the wavelet coefficients in each subband (D1,D2,A2), ratio of the minimum to the maximum of the wavelet coefficients in each subband (D1,D2,A2), and instantaneous RR intervals. For more details please refer to [7].

Since our design objective is to maximize the system throughput rather than reduce the processing latency, we adopt a more resource-saving FE pipeline architecture (Fig.5), and achieve higher throughput by instantiating multiple pipelines (10 pipelines to match 10Gbps processing rate). First, we tightly combine the QRS detection and the feature extraction procedures in our PE pipeline. We share one wavelet transformation module between these two procedures to save resources. When ECG data is ready in the local block ram, the WT module executes a pipelined wavelet transformation. The D2 output is sent to R peak detection module, where all R peak positions as well as the 64-point QRS segments centered at R-peaks are extracted from the record. After the QRS detection, a second wavelet transformation to the QRS segments is executed using the same WT hardware, which generates sub-band wavelet coefficient inputs to the feature extraction procedure. Second, we share the execution engines among FE computations of all sub-band data. We use a controller logic to manage the execution orders of Min/Max, autocorrelation, and variance computations. Finally, each FE pipeline outputs 11 statistics features as well as user's connection information message to one feature FIFO for further classification.

*2) **ANN Based Classification**:* The classification module contains a pre-trained ANN with 11 input neurons, 20 hidden neurons and 6 output neurons. Since we reduce the size of input data to the ANN via the feature extraction stage, the computation complexity of ANN is greatly reduced. Thus, a more light-weight neuron design is adopted to save the on-chip resources. In each neuron, we put only one multiply-accumulate unit and a finite state machine to control the access of data from the output of the former neurons. Thus, the neuron executes one multiply-accumulate operation to one of its input data with its pre-trained weight in each cycle. Also, we uses a shared look-up table (64K depth and 16bit width) stored in the on-chip block ram to implement the sigmoid function, which improves its processing speed.

III. Experimental Results and Discussion

A. Experiment Setup

We evaluate our design on one computing node of mimicry computer [8] based on Xilinx XC6VLX550T FPGA, which contains two 8GB DDR3 SDRAM, one 72Mb SRAM, and 10G Ethernet. The embedded microprocessor is implemented by Microblaze, a light-weight soft core provided by Xilinx. The system clock frequency is 156.25Mhz which is equal to the frequency of 10Gbps PHY interface.

In our study of cloud based ECG classification, 23 ECG records were selected from the widely used MTI-BIH arrhythmia database [9]. All records were digitized at 360 samples per second with 11-bit resolution for slightly over 30 minutes. These records include six ECG beat types. For each beat type, we choose 50% beats for training and another 50% beats for testing. The parameters related to the neuron network were trained offline using software and then fixed to the implementation in FPGA. Ten FE pipelines were integrated in feature extraction module to maximize the throughput and make full use of reconfigurable resources. Since the ECG input data is 11-bit resolution, and there is a square term in variance computation, so that we choose a unified 32-bit fixed point for all arithmetic logical units. It is worth noting that we implement the sigmoid function in 16bit 64K depth look-up table. However, according to [10], 8 bits fixed point precision in all ANN forward retrieving computation causes less than 10^{-4} error of the floating point version. Thus, our precision configuration can reduce the accuracy error greatly. Our experiment shows that the accuracy of diagnosis in our implementing are the same as that of the software version.

Table I shows the device utilization summary of our design generated by Xilinx ISE 14.1. It uses only 19.0% of total slice registers, 34.7% of total slice LUTs, 31.1.1% of total BRAMs and 53.1% of total DSP48E1s.

TABLE I: Device utilization summary

	Slice Registers	Slice LUTs	BRAM	DSP48E1
Available	687360	343680	2528	864
Utilization	19.0%	34.7%	31.1%	53.1%

B. Performance Evaluation

TABLE II: Latency of three ECG data processing steps.

Task assignment	Feature extraction	Classification
4096	9728	61

Table II illustrates the latency of three ECG data processing steps in our design. The latency of the FE pipeline to process 4K points ECG data containing 12 heart beats is 9782 cycles. Although the latency of reading data from the DDR3 memory to the block is 4096 cycles, such data acquisition latency was hidden in the execution pipeline using double buffering. Since the system clock is 156.25Mhz, the throughput of single FE pipeline is $8KB \div (9728cycles \times 1/156.25Mhz) = 128.5MB/s$. Considering the max theoretical upstream throughput of 10Gb Ethernet, ten FE pipelines are sufficient to meet the requirement of the system-level throughput: $(10Gb/s \div (128.5 \times 8)Mb/s = 9.72 < 10)$. On the other hand, the latency of the neural network module to classify one heart beat is 61 cycles. As a result, its throughput can be calculated as $8KB \div (61cycles \times 1/156.25Mhz \times 12beats) = 1.71GB/s > 10Gb/s$, which also meet the requirement of the system-level throughput.

To evaluate the performance, we simulate the user requests by client PCs, which generate multiple connections to our prototype board and upload ECG data from the testing set. Each connection uploaded 4K sequential sample points to the server. We implemented a software version of the same ECG analysis application on IBM x3635 M3 server with Intel x5650 CPU (2.66 Ghz), 8GB DDR3 memory and Intel X520-SR2 10GE network adapter. The software version was programmed in standard C using Linux socket for network communication.

*1) **Maximum Transactions Per Second**:* We used maximum transactions per second (Tps) as the performance metric to evaluate the system-level throughput. A completed transaction included the uploading of the 4K points ECG data, ECG

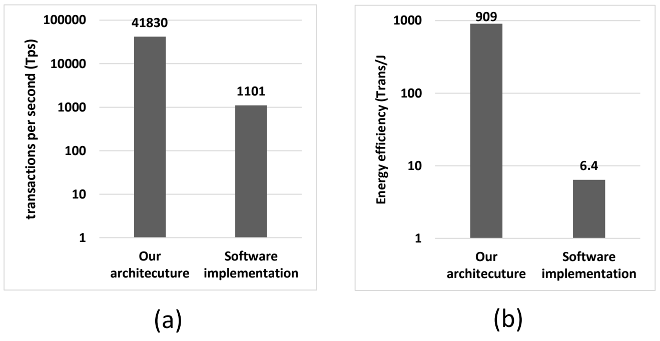

Fig. 7: Comparison of performance (a) and energy efficiency (b) between our standalone SOPC implementation and software implementation.

data classification and the sending back of the result. Our FPGA based solution was able to accomplish 41830 transactions per second, which is 38 times that of the software version (Fig.7(a)). Compared to the software version, where both I/O-intensive workload (network handling) and computation-intensive workload (ECG data processing) contend for the same CPU time, our standalone SOPC based architecture deals with two parts of workload by separated hardware modules in a pipeline form, thus achieves a better performance.

*2) **Energy Efficiency**:* We use the term transactions per joule ($Trans/J$) to define the energy efficiency of two systems. We measured the total energy consumption $P(J)$ and the total completed transactions T during one-hour test, then we calculated the energy efficiency as $T/P(Trans/J)$. As shown in Fig.7(b), our energy efficiency was $909Trans/J$, which is 142 times over the software version.

*3) **Benchmarking with the State-of-the-arts**:* A comparison with prior works in processing throughput (ECG samples processed per second (Sams/s)) is given in Table III. Since most of the ECG processing systems [2] [5] [11] are designed for portable and single user scenario, they typically employ embedded processors or light-weight hardwire to save power and chip size. So that these systems have a moderate ECG processing capacity (e.g. Hashim [11] achieves only 29.06×10^3 Sams/s). Ieong [12] otherwise designed a throughput optimized QRS detection hardware in FPGA, which achieves 57.53×10^6 Sams/s. However, it only accelerated QRS detection while our work presents a whole cloud based ECG analysis system including network handling, connection management, QRS detection, feature extraction and classification. Zhou [13] proposed an FPGA-assisted cloud framework for massive ECG processing. It uses a commercial server to handle network connections and attaches an FPGA via PCIE to accelerate ECG processing. However, the data transmission latency between the PC host and the FPGA degrades the whole performance (17.75×10^6 Sams/s). Also, it does not contain feature extraction and classification logic. Compared with it, our system is a stand-alone SOPC system. By offloading both networking operations and ECG data analysis in a close-coupled architecture, our system can achieve 63.89×10^6 Sams/s.

TABLE III: Benchmarking with the state-of-the-arts (ECG samples processed per second (Sams/s))

Hashim [11]	Ieong [12]	Zhou [13]	Ours
29.06×10^3	57.53×10^6	17.75×10^6	63.89×10^6

IV. Conclusion

In this brief, we have presented the design and implementation of a standalone SOPC based system that tightly couples the network IO handling and data processing in a single FPGA to accelerate massive ECG data analysis. We have designed and implemented a novel TCP/IP hardware stack that supports 10Gb Ethernet and 100K concurrent TCP connections to meet the requirement of high concurrent requests. Due to the hardware pipeline implementation of network protocols and full-featured ECG signal processing, our implementation can achieve higher performance, lower energy consumption with stand-alone cloud service functionalities of ECG data analysis. We evaluated our design using a prototype FPGA implementation, showing 38X speed-up in performance and 142X improvement in energy efficiency over existing servers.

References

[1] F. Massé, M. V. Bussel, A. Serteyn, J. Arends, and J. Penders, "Miniaturized wireless ECG monitor for real-time detection of epileptic seizures," *ACM Trans. Embed. Comput. Syst.*, vol. 12, no. 4, p. 102, 2013.

[2] Y. Zou, J. Han, S. Xuan, S. Huang, X. Weng, D. Fang, and X. Zeng, "An energy-efficient design for ECG recording and R-peak detection based on wavelet transform," *IEEE Trans. Circuits Syst. Express Briefs (TCAS-II)*, vol. 62, no. 2, pp. 119–123, 2015.

[3] S. Pandey, W. Voorsluys, S. Niu, A. Khandoker, and R. Buyya, "An autonomic cloud environment for hosting ECG data analysis services," *Future Gener. Comp. Syst.*, vol. 28, no. 1, pp. 147–154, 2012.

[4] M. U. K. Khan, J. M. Borrmann, L. Bauer, M. Shafique, and J. Henkel, "An h. 264 quad-fullhd low-latency intra video encoder," in *Proc. Conf. Design, Automation and Test in Europe*. EDA Consortium, 2013, pp. 115–120.

[5] C. I. Ieong, P. I. Mak, C. P. Lam, C. Dong, M. I. Vai, P. U. Mak, S. H. Pun, F. Wan, and R. Martins, "A $0.83-\mu$w QRS detection processor using quadratic spline wavelet transform for wireless ECG acquisition in $0.35-\mu$m cmos," *IEEE Trans. Biomed. Circuits Syst.*, vol. 6, no. 6, pp. 586–595, Dec 2012.

[6] S. Kadambe, R. Murray, and G. F. Boudreaux-Bartels, "Wavelet transform-based QRS complex detector," *IEEE Trans. Biomed. Eng*, vol. 46, no. 7, pp. 838–848, 1999.

[7] S. N. Yu and Y. H. Chen, "Electrocardiogram beat classification based on wavelet transformation and probabilistic neural network," *Pattern Recogn. Lett.*, vol. 28, no. 10, pp. 1142–1150, 2007.

[8] J. Wu, "Meaning and vision of minic computing and minic security defense," *Telecommunications Science (in Chinese)*, vol. 7, pp. 2–7, 2014.

[9] G. B. Moody and R. G. Mark, "The impact of the mit-bih arrhythmia database," *Eng. Med. Biol. Mag.*, vol. 20, no. 3, pp. 45–50, 2001.

[10] J. L. Holi and J. N. Hwang, "Finite precision error analysis of neural network hardware implementations," *IEEE Trans. Comput.*, vol. 42, no. 3, pp. 281–290, 1993.

[11] A. Hashim, C. Y. Ooi, R. Bakhteri, and Y. W. Hau, "Comparative study of electrocardiogram qrs complex detection algorithm on field programmable gate array platform," in *IEEE Conf. Biomedical Engineering and Sciences*. IEEE, 2014, pp. 241–246.

[12] C. I. Ieong, M. I. Vai, and P. U. Mak, "Ecg qrs complex detection with programmable hardware," in *Proc. 30th Annu. Int. Conf. Engineering in Medicine and Biology Society*. IEEE, 2008, pp. 2920–2923.

[13] S. Zhou, Y. Zhu, C. Wang, X. Gu, J. Yin, J. Jiang, and G. Rong, "An fpga-assisted cloud framework for massive ecg signal processing," in *Proc.12th Int. Conf. Dependable, Autonomic and Secure Computing*. IEEE, 2014, pp. 208–213.