Design of a Hardware Accelerator of Bit_Q_Apriori Algorithm for Feature Analysis of Electrocardiogram (ECG) Signals.

Masudul Quraishi, Sheena Mohan, Neha Aggarwal
Department of Computer Engineering
Arizona State University
Tempe, AZ
mquraish@asu.edu, smohan21@asu.edu, neha.aggarwal@asu.edu

Analysis of Electrocardiogram signals is one of the most useful techniques in diagnosis of heart issues. For proper diagnosis, it is important to extract the most frequent patterns in ECG features of a patient. Bit_Q_Apriori algorithm is a powerful technique to get the frequent items from a database. The motivation to design a hardware accelerator for this algorithm is to improve the existing solutions for ECG analysis. Existing solutions are not operated in real time and they are not scalable as well. With the emerge of Internet of Things (IoT), our designed accelerator will be very useful in a cloud based real time ECG diagnosis application as well as wearable ECG monitoring devices.

.

Keywords—Bit_Q_Apriori, Algorithm, FPGA, AND, bitset, ECG, frequent itemset, interleaving

I. INTRODUCTION

Association rule mining [1] is a popular method for analysis of existing relations among large datasets. One of the steps in Association rule mining is to find the most frequent itemset in a database. In this project, we will implement a hardware accelerator for the Bit_Q_Apriori algorithm, an improvement by Xiaoqi et al. [2] over the traditional Apriori-like algorithm implemented by Baker and Prasanna [3] for association rule mining. The algorithm iterates over data in a database and generates bitset for frequent items in the dataset. The generated bitset can be used to identify a certain pattern in the dataset. In our project, we are using Bit_Q_Apriori algorithm to identify patterns in data extracted from ECG signals. Figure 1 shows different segments of typical ECG signals. There are two different types of features: Distance feature and Amplitude feature. Among all the features, we decided to use Q-R distance and R-T distance as exemplar ones.

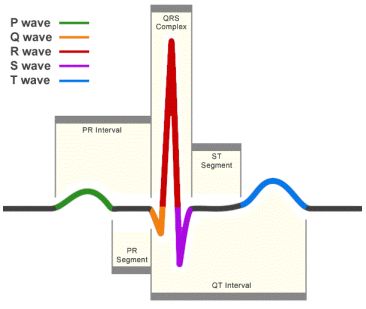

Figure 1: Different segments of ECG Signal

Current technologies in ECG monitoring and data analytics do not provide real-time signal processing and diagnosis. Currently two types of ECG monitoring are available. One is the bedside monitoring which requires the patient to be at hospital beside the machine. Another one is remote ECG telemonitoring that takes 3-4 minutes to record ECG and send to workstation for analysis. Current setups are not scalable as well. The FPGA based implementation will help to accelerate association rule mining processing and achieve better performance, energy efficiency and high scalability.

The report is organized as follows. Section II gives an overview of the algorithm using an example. Section III focuses on the design specification and algorithm implementation. The subsequent sections describe the Hardware Accelerator architecture and the optimization performed. Then we end with discussing results and conclusion.

II. BIT_Q_APRIORI ALGORITHM

Bit_Q_Apriori algorithm is a modified version of traditional Apriori algorithm that generates a set of frequent items from an overall database. The Algorithm can be divided into four steps:

Step 1: The attribute values are mapped to binary numbers. Values of attributes are discretized in several segments and represented in binary bits. The possible values of attributes are partitioned first into several intervals. Then number of partition is decided. Each partition is mapped to an integer. In Table 1, we created 2 partitions for patient, 3 partitions for Q-R interval and 4 partitions for R-T interval.

Table 1: (a) Original Transactions (b) (c) (d) : Portioning for Patient, Q-R interval and R-T interval (e) Transaction after mapping (f) Some 1-candidate and their bitsets

ID	Patient ID	Q-R	R-T
1	P1	0.62	5.5
2	P2	0.67	6.1
3	P1	0.71	6.7
4	P2	0.69	7.1

(a)

Interval	Integer	Binary
P1	1	01
P2	2	10

(b)

Q-R Interval	Integer	Binary
< 0.65	1	01
0.65~0.7	2	10
> 0.7	3	11

(c)

R-T Interval	Integer	Binary
< 6.0	1	001
6.0-6.5	2	010
6.5-7.0	3	011
> 7.0	4	100

(d)

ID	Patient	Q-R	R-T
1	01	01	001
2	10	10	010
3	01	11	011
4	10	10	100

(e)

1-Candidates	Binary bits	ID	bitset
Patient: P1	0100000	1,3	1010
Patient: P2	1000000	2,4	0101
Q-R: >0.7	0011000	3	0010
R-T: <6.0	0000001	1	1000

(f)

Step 2: 1-candidates are all the attributes that occurred at least once in original transaction. We specify each candidate with two parameters: Binary bits and bitset. Binary bits are generated using the binary representation of partitions. Number of bits in binary bits will depend on total number of bits needed to describe all partitions. In the example of Table 1, Patient, Q-R and R-T requires 2, 2 and 3 bits respectively. So, total number of bits representing binary bits of candidates will be 7. Therefore, Patient 1 is represented by 0100000. We generate 1-candidate's bitset by setting the bits of transaction it belongs to. For example, Patient 1 belongs to transaction 1 and 3 among all 4 transactions. So bitset of Patient 1 is 1010.

Step 3: In step 3, we generate the frequent items. The candidates that occur more than or equal a minimum support count are considered as frequent items. The frequency of an item is determined from its bitset. In each step of the algorithm, all possible combinations of attributes are checked with the minimum support count and added to frequent itemset. The algorithm stops when no new frequent items are generated. The binary bit of new generated item is bitwise OR operation of the corresponding candidates' bitsets. The bitset of new generated item is bitwise AND operation of the corresponding bitsets.

Step 4: In the process of creating new items, some overlapped or duplicate items are generated. In this step, those items are deleted.

In the example, result will be Patient 1 with Q-R interval 0.65-0.7 combination as it is the only 2-candiadte that occurred twice, which is equal to the minimum support count.

III. Design Specification

The high-level models are designed in Simulink using Synphony Model Compiler Blockset. It allows us to map our design for target FPGA and ASIC implementation. For FPGA, our target board was Xilinx Vertex 7 and for ASIC, we targeted 28nm library.

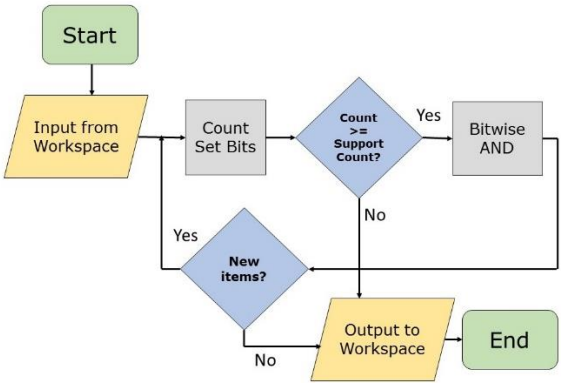

Figure 2: Flowchart of Bit_Q_Apriori Algorithm

Figure 2 shows the flowchart for the algorithm. There are two important processes: Counting the number of set bits and performing bitwise AND operation among the candidates. The algorithm is implemented in MATLAB first to generate a test bench to cross-check our results from hardware.

Our design experiences a variable number of input to each iteration because there is no way to know how many new candidates will be generated and pass the support count requirement. This creates a problem in specifying the loop latency. The only way to have a constant latency is to know the worst-case requirement based on the input data set. For basic architecture, we constrained our input to only 3 to reduce the worst-case memory and gate requirement. Moreover, we initially overestimated our target throughput as 100M samples per second. Due to the variable input constraint, we reduced our target throughput to 50M samples per second. The result section describes how we achieved this target using optimizations through interleaving and parallelism. Furthermore, our target was to implement the design for both FPGA and ASIC targets to support different application areas.

IV. Architecture Design & Optimization

This section describes the implementation of the algorithm for FPGA and ASIC targets. As mentioned previously, our architecture has two major sections; one for counting set bits and the other is for computing bitwise AND operation. Figure 3 shows a portion of the designed baseline architecture. This is the subsystem used for counting set bits of the candidates. The subsequent subsections discuss about the design and mapping of baseline architecture and optimized architectures.

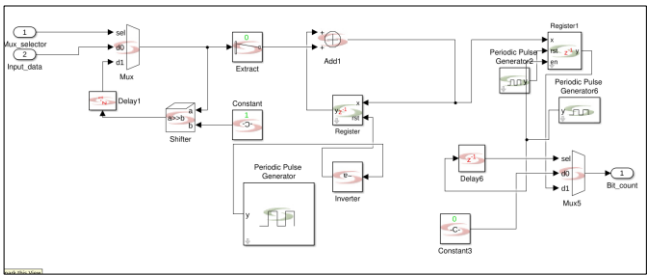

Figure 3: Subsystem for counting set bits

A. BASELINE ARCHITECTURE

We mapped our baseline architecture to a 28nm ASIC design using SHLS block. The implementation was done using a Vdd of 1V and LVT, RVT and HVT optimizations turned on. ASIC properties for baseline architecture are summarized in Table 2. As we can see from table, the baseline architecture is way behind our target throughput of 50 M Samples/s.

Table 2: ASIC Properties for Baseline Architecture

Property	Value
Area (um^2)	5725.264527
Dynamic Power (mW)	14.9
Leakage Power (mW)	0.0865469
Frequency (MHz)	781.25
Throughput (M sample/s)	1.08

We synthesized the circuit in Xilinx Virtex7 XC7VX485T FPGA. The resource utilization was determined. The tool estimated that the model can run at a frequency of 781.25 MHz. The specifications of the design are mentioned in the Table3. This is the minimum resource required to implement our architecture in FPGA. As we will see in next sections, the optimized designs are more resource hungry.

Table 3: Resource Utilization of FPGA synthesis for Basic Architecture

Property	Value
I/O Ports	22
DSP48s	0
Non-I/O Register Bits	151
Block RAMs	0
LUTs	328
Est. Frequency	300.4

B. ARCHITECTURAL OPTIMIZATIONS

Since the baseline architecture was unable to meet the target throughput, we used interleaving and parallelism to design the hardware accelerator. Architectural interleaving allows us to utilize the loop latency to feed in more data and increase throughput. The subsystem used for counting set bits has a loop latency of 10. To utilize this latency, we used an interleaving factor of 10.

As we couldn't achieve our target throughput using interleaving only, we decided to increase throughput further. We used parallelism with factor 2, 4 and 8 for further optimization. Parallelism allows us to slow down the circuit to reduce power density and increase throughput. For convenience, interleaving will be abbreviated as IL and Parallelism with factor 2,4 and 8 will be abbreviated as P2, P4 and P8 respectively in some of the tables and diagrams. Figure 4 shows the diagram for parallelism with factor 2.

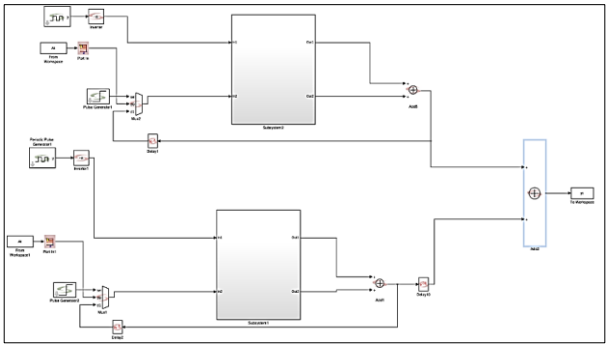

Figure4: Circuit with IL and P2 optimization

The optimized designs are mapped into ASIC and FPGA with the same configuration as the baseline architecture. For comparison purpose, we only use two of the optimized circuits: interleaved and interleaving with parallelism factor of 8. The circuit used for interleaving is the same as baseline circuit, only data for 10 different patients are fed into the circuit to utilize the loop latency of 10. Figure 2 shows the circuit with interleaving and parallelism factor 2 implemented. Table 4 and 5 summarizes the properties for ASIC design and resource utilization for FPGA.

Table 4: Properties of ASIC for Interleaved and Parallel Architectures

Property	Interleaved	IL + P8
Area (um^2)	41175.45196	502614.0728
Dynamic Power (mW)	15.1809	176.8347
Leakage Power (mW)	0.5721132	6.041
Frequency (MHz)	715	715
Throughput (M samples/s)	8	49.96

Output throughput is calculated using the following formula:

$$throughput = \frac{M}{N}\ (frequency)$$

Where M is the number of output generated in N cycles and frequency is the rate of input data.

Calculation of bitwise AND of all possible combination of candidate works as a bottleneck in the design. It occupies a good number of clock cycles which cannot be utilized for interleaving or any other optimization. In baseline architecture, output was generated at each 720 cycles and input frequency was tuned to 781.25 MHz.

$$throughput = \frac{1}{720}\ (781.25\ M) = 1.08\ M\ Samples/s$$

If we try to achieve our target throughput with the baseline architecture, the frequency required is 32 GHz, which is not realistic and the slack is also violated. As we will see in next section of report, using parallelization of factor 8 helped us achieve our target throughput of 50 M Samples per second.

Table 5. Resource Utilization of FPGA synthesis for Interleaved and Parallel Architectures

Property	Interleaved	IL+P8
I/O Ports	22	92
Non-I/O Register Bits	218	1724
Block RAMs	1	7
LUTs	207	1618
Est. Frequency (MHz)	715	715

V. Result & Analysis

Reports from FPGA and ASIC mapping provide important insights on the designed hardware accelerator. Our primary design target was to achieve the desired throughput. Figure 5 delineates how gradually we reached our target throughput by interleaving and parallelism.

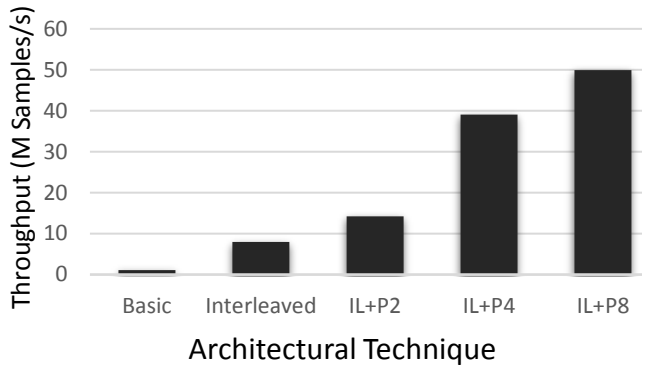

Figure 5: Throughput for different Architectural Technique

We started with a throughput of only 1.08 M Samples per second in our baseline architecture. An interleaving factor of 10 boosted the throughput to almost 8 times. With added parallelism of factor 2, 4 and 8, the throughput was improved almost 50 times than the baseline architecture.

If we analyze the area and power in Table 4, we can see that they increased significantly. For ASIC design, area and power are two important concerns. With increased parallelism, the number of component in design increases. Increased number of component results in a greater area and more power dissipation.

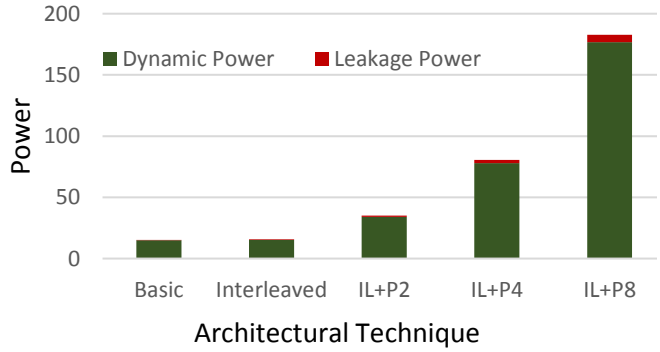

Figure 6: Dynamic and Leakage power for different architectural techniques.

Figure 6 shows the dynamic and leakage power for different architectural techniques. As expected, total power increases up to 13 times than the baseline architecture. Power density (Power/area) reduces in the IL+P8 design, which is an important benefit of using parallelism. Moreover, we know that leakage current increases in higher factor of parallelism; which is also evident from Figure 6

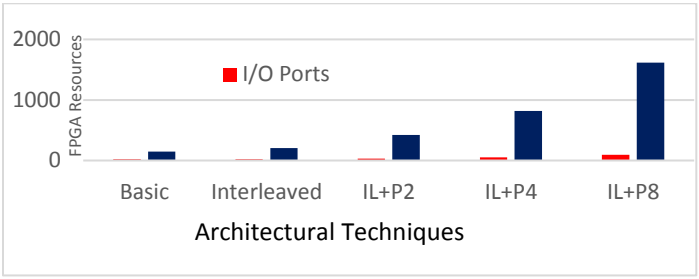

Figure 7: FPGA resource utilization for different architectural techniques.

Table 3, 5 and Figure 7 summarizes the resource utilization of FPGA implementation for basic and optimized design. As we incorporate different optimization, more resources are utilized due to increase in number of component in the circuit.

Existing software solutions running on general purpose CPU take 3-4 minutes to perform the data analysis. On the other hand, our solution takes few seconds to generate bitset and they are fed serially into the accelerator to generate and output of 50M samples per second. Moreover, the ASIC solution will provide a power and area optimized solution to existing design [5] for wearable ECG devices. The FPGA based solution gives us more flexibility that ASIC. So, if any better algorithm for feature extraction comes up, we can just modify the existing design and implement it in FPGA again. In ASIC, once the design in implemented in a chip, it can no longer be changed. So, based on application, any of the solutions can be opted. Our current design has scalability constraints. The current design can serve 240 patients at a time.

VI. Conclusion

We designed and optimized an implementation of Bit_Q_Apriori algorithm for ECG feature extraction. For future work, the design constraints can be challenged to support scalability. Further optimization of area and power can be done for ASIC design using Vdd optimization. Our design frequency is very high (715 MHz) which is not desirable for practical implementation. We can use retiming and pipelining to reduce this frequency. The primary motivation for design of the hardware accelerator was to contribute to the target application of diagnosis using ECG. We believe our solution has made a significant contribution towards future cloud based ECG monitoring [4] and wearable ECG devices [5].

REFERENCES

[1] Rakesh Agrawal, Tomasz Imieliński, Arun Swami, Mining association rules between sets of items in large databases, ACM SIGMOD Record, v.22 n.2, p.207-216, June 1, 1993 [doi>10.1145/170036.170072]

[2] Xiaoqi Gu, Yongxin Zhu, Shengyan Zhou, Chaojun Wang, Meikang Qiu, and Guoxing Wang. 2016. A Real-Time FPGA-Based Accelerator for ECG Analysis and Diagnosis Using Association-Rule Mining. ACM Trans. Embed. Comput. Syst. 15, 2, Article 25 (February 2016). DOI: http://dx.doi.org/10.1145/2821508

[3] Zachary K. Baker, Viktor K. Prasanna, Efficient Hardware Data Mining with the Apriori Algorithm on FPGAs, Proceedings of the 13th Annual IEEE Symposium on Field-Programmable Custom Computing Machines, p.3-12, April 18-20, 2005 [doi>10.1109/FCCM.2005.31]

[4] Shengyan Zhou, Yongxin Zhu, Chaojun Wang, Xiaoqi Gu, Jun Yin, Jiang Jiang, Guoguang Rong, An FPGA-Assisted Cloud Framework for Massive ECG Signal Processing, Proceedings of the 2014 IEEE 12th International Conference on Dependable, Autonomic and Secure Computing, p.208-213, August 24-27, 2014 [doi>10.1109/DASC.2014.45]

[5] Jun Dong, Jia-wei Zhang, Hong-hai Zhu, Liping Wang, Liu Xia, Zhen-jiang Li, A Remote Diagnosis Service Platform for Wearable ECG Monitors, IEEE Intelligent Systems, v.27 n.6, p.36-43, November 2012 [doi>10.1109/MIS.2012.4]