Frequent Items Counter Based on Binary Decoders

Katsumi Inoue\textsuperscript{1,2a)}, Trong-Thuc Hoang\textsuperscript{2b)}, and Cong-Kha Pham\textsuperscript{2c)}

\textsuperscript{1}Advanced Original Technologies Co., Ltd (AOT), Tokyo, Japan
\textsuperscript{2}The University of Electro-Communications (UEC)
1-5-1 Chofugaoka, Chofu, Tokyo 182-8585, Japan
a) ugg44151@nifty.com  b) thuc@vlsilab.ee.uec.ac.jp  c) phamck@uec.ac.jp

Abstract: In this paper, the hardware design of frequent items counter is proposed. The key idea is to create a matrix of binary-value by using an array of binary-decoder to decode all of the input items in parallel. After that, an array of population-count modules are applied to the rows of the matrix to generate counting results. The architecture was implemented with five options of bit/item from 6-bit/item to 10-bit/item, and seven options of count-register bit-width from 8-bit counters to 32-bit counters. Therefore, there were 35 different versions of implementation presented in this paper. Those implementations were built on the Field-Programmable Gate Array (FPGA) board of Altera Arria V SoC development kit. Also, they were synthesized to chips with the process technology of 65nm Silicon On Thin Buried-Oxide (SOTB). The experimental results of the proposed architecture achieved outstanding timing performances compared to other attempts to date.

Keywords: Binary Decoder, FPGA, Frequent Items Counter, SOTB.

Classification: Integrated circuits (Logic)

References

[1] T. Arici, \textit{et al.}: “A Histogram Modification Framework and Its Application for Image Contrast Enhancement,” IEEE Trans. on Image Processing 18 (Sep. 2009) 1921 (DOI: 10.1109/TIP.2009.2021548).
[2] J. E. Duarte-Sanchez, \textit{et al.}: “Hardware Accelerator for the Multifractal Analysis of DNA Sequences,” IEEE/ACM Trans. on Computational Biology and Bioinformatics PP (May 2017) 1 (DOI: 10.1109/TCBB.2017.2731339).
[3] C. Estan and G. Varghese: “New Directions in Traffic Measurement and Accounting,” Conf. on App., Tech., Archi., and Protocols for Comp. Comm. 32 (2002) 323 (DOI: 10.1145/633025.633056).
[4] S. Das, \textit{et al.}: “Thread Cooperation in Multicore Architectures for Frequency Counting Over Multiple Data Streams,” Int. Conf. on Very Large Data Bases Endowment (PVLDB) 2 (Aug. 2009) 217 (DOI: 10.14778/1687627.1687653).
[5] S. Das, \textit{et al.}: “CAM Conscious Integrated Answering of Frequent Elements and Top-k Queries Over Data Streams,” 4th Int. Workshop on Data Management on New Hardware (2008) 1 (DOI: 10.1145/1457150.1457152).
1 Introduction

Frequent Items Counter (FIC) is the problem of counting items’ appearances in the given itemset. The problem is commonly found in various tasks related to many fields of research. For example, the FIC task can be applied to those applications of image histogram [1], DNA sequences analysis accelerator [2], and network traffic measurement [3] to speed up the statistic analysis process of a vast amount of pixels, DNA’s nucleobases, and IP-packet flows, respectively. The FIC function is also one of the most fundamental functions utilized in various data mining algorithms such as frequent element [4–8], Top-K query [5], quantile query [9], iceberg data cube [10], and iceberg query [11]. Throughout the time, the requirements of FIC increased rapidly. Nowadays, FIC implementation is required not only for the timing performances but also be restricted to one data-pass in some applications such as data streaming.
FIC implementation was well studied because of its vital role. However, due to the nature of the CPU-based approach, parallel computation of FIC was hard to achieve on software [4]. When the limitations of CPU-based architecture are reached, any optimization at the software level will have a minor effect or become irrelevant. Because of this, on the trend of the rapid development of database research area, an efficient hardware solution was becoming more and more necessary. Furthermore, compared to CPU-based designs, hardware implementations have substantial advantages of parallel processing, low-power consumption, and free from memory-buffer like stack or Random Access Memory (RAM).

The primary goal of this paper is to present a high-speed hardware FIC architecture. Moreover, because FIC can be utilized in many kinds of applications, the proposed design has multiple options to fit the broader requirements. For specific, there were five options of the number of bits per input item, i.e., 6-bit/item, 7-bit/item, 8-bit/item, 9-bit/item, and 10-bit/item, and seven options of the count-register bit-width, i.e., 8-bit, 12-bit, 16-bit, 20-bit, 24-bit, 28-bit, and 32-bit counters. As a result, there were 35 different FIC implementations presented in this paper. The count-register bit-width determines the maximum number that the counters can hold, and the number of bit/item determines how many types of items can be in the given itemset.

The idea of the proposed architecture is the usage of a binary-decoder array to generate a matrix of binary-value from the input items, with each binary-decoder creates one column of the matrix. After that, population-count modules are applied to the rows of the matrix to generate a column of counting results. A Field-Programmable Gate Array (FPGA) test system was built on the Altera Arria V SoC development kit with the FPGA chipset of 5ASTFD5K3F40I3. Furthermore, 35 FIC implementations were synthesized at the Application-Specific Integrated Chip (ASIC) level by the Synopsys tools using the 65nm Silicon On Thin Buried-oxide (SOTB) process technology. The experimental results showed that when increasing the number of bit/item and the count-register bit-width, the resources costs were increased and the counting speeds were dropped. For specific, for SOTB-65nm results, one more bit/item and four more bits of count-register bit-width caused the counting speeds decreased about 630 Million Items Per Second (MIPS) and 70 MIPS, respectively, while both results of area and power increased about \(1.74\times\) and \(1.09\times\), respectively. The MIPS performances of the SOTB-65nm layouts were about 75% of those of the FPGA implementations. For the FPGA results, one more bit/item and four more bits of count-register bit-width caused the counting speeds decreased about 900 MIPS and 200 MIPS, respectively, and logic utilization increased about \(1.82\times\) and \(1.08\times\), respectively.

The remainder of this paper is organized as follow. Section 2 describes the proposed FIC architecture. Section 3 provides the experimental results along with the comparison with other recent works. Section 4 discusses the achieved results further. Finally, Section 5 gives the conclusion for the paper.
Fig. 1: The block diagram of the FPGA test system.

2 Frequent Items Counter Architecture

The proposed architecture was implemented with five options of bit/item and seven options of count-register bit-width. The five options for bit/item are 6-bit, 7-bit, 8-bit, 9-bit, and 10-bit. The number of bit/item adjusts the maximum number of item types in the itemset. For example, 6-bit/item and 8-bit/item mean that there are at the maximum of 64 and 256 different types of items in the itemset, respectively. For the count-register bit-width, the seven options are 8-bit, 12-bit, 16-bit, 20-bit, 24-bit, 28-bit, and 32-bit. The bit-width of count-register defines the maximum counting results that the counters can hold. For example, with the options of 8-bit and 16-bit counters, the count-register can hold the counting results up to 255 and 65,536, respectively. To summarize, the various design options of the proposed architecture can satisfy the broader needs of FIC applications. The five options of bit/item and the seven options of count-register bit-width made 35 different versions of implementation presented in this paper. The completed test system was built on the Altera Arria V SoC FPGA development kit. Moreover, the FIC core modules were synthesized at the ASIC level using the SOTB-65nm process.

2.1 System Overview

The block diagram of the FPGA test system is described in Fig. 1. As shown in the figure, the system deployed the Database Processor (DBP), memories, and FIC_IP to control the operation, store the itemset and counting results, and perform the FIC task, respectively. Besides the FIC core module, the FIC_IP also included a Direct Memory Access (DMA) and a controller to maximize the data transfer rates and communicate with the DBP, respectively. The off-chip memory stored the completed itemset, and the on-chip memory was used as a data buffer; the on-chip memory was also used to store the final counting results. To optimize the transfer rates and resources cost, the interfaces of the Avalon Bus were all set to 256-bit data-width because of the kit’s bandwidth limitation. Therefore, depends on the bit/item scenario, the DMA can transfer a different number of items by each operating clock. For instance, with the fixed input of 256-bit, the DMA can provide the FIC core module with 32 and 25 input items per clock when the bit/item scenarios are 8-bit/item and 10-bit/item, respectively.

A typical operation of the proposed FPGA system begins with the command from the DBP to the FIC_IP by the information of the itemset in the memories, i.e., addresses and data length. After that, by using the IP controller, the DBP starts the DMA to begin the data transfer process. The FIC
core module keeps counting up the results along with the DMA’s activity. When the data transfer process is completed, the counting results in the FIC core module are given back to the DBP to complete the system operation.

2.2 FIC Core Module
The architecture of the FIC core module is given by Fig. 2. Because the proposed design aims to produce the full FIC results, the idea of counters array was deployed as seen in the figure. In Fig. 2, the array of counters have $2^N$ counters from the $Result-1$ to the $Result-2^N$, with $N$ is the number of bit/item. They keep counting up the results for all item types while the system was scanning throughout the itemset. Therefore, the completed FIC results can be answered by a single data-pass.

As mentioned earlier, depending on the number of bit/item, there were different numbers of input items that can be transferred to the FIC core module by one operating clock. Therefore, the number of binary-decoder was also changed corresponding to the number of bit/item as seen in Fig. 2. For example, 42 and 36 binary-decoder were required in the implementations with the design options of 6-bit/item and 7-bit/item, respectively. The idea for the binary-decoder array is to achieve the parallel processing for all of the input items, thus optimizes the system operating speed.

![Fig. 2: The architecture of the FIC core module with the N-bit/item option.](image)

![Fig. 3: The design of the 8-to-256 binary-decoder.](image)
Fig. 4: The design of the population-count module $\sum_{-32}^{0}$ (a) that contains the submodules of $\sum_{-16}^{5}$ (b), $\sum_{-8}^{4}$ (c), and $\sum_{-4}^{3}$ (d).

The array of binary-decoder produces the $I_{x,y}$-matrix of binary-value as shown in Fig. 2; the $x$-value is from 1 to floor$(256/N)$, and the $y$-value is from 1 to $2^N$. Each binary-decoder generates a matrix column, which is its corresponding input item’s binary-value. In the $2^N$ bits of binary-value, there is only one bit that has the value of ‘1’, and the other $(2^N-1)$ bits are zeros. Therefore, $2^N$ counting results can be calculated by performing the sum functions on the rows of the $I_{x,y}$-matrix. As seen in Fig. 2, the sum tasks are done by the population-count $\sum$ modules. After the sum, the current counting results are added up with the previous counting results stored in the count-register array. At the end of the operation, the count-register array returns the final counting results to the DBP. Furthermore, at the beginning of the operation, they can be reset to their initial values of zeros by a DBP command.

The design of the binary-decoder is the same as the design of a general address-decoder commonly used in RAMs. Fig. 3 shows an example design of the 8-to-256 binary-decoder. As seen in Fig. 3, 256-bit of $I$-value is generated based on the 8-bit input $A$-value, and there is only one asserted bit in the $I$-value as seen in the truth table. As shown in the Fig. 3, the design of the 8-to-256 binary-decoder has three stages of combinational logic. The first stage is made up of every two input bits with their inverted bits. For instance, the $A_0$, $\overline{A_0}$, $A_1$, and $\overline{A_1}$ make the combination values of $(A_0 \ AND \ A_1)$, $(A_0 \ AND \ \overline{A_1})$, $(\overline{A_0} \ AND \ A_1)$, and $(\overline{A_0} \ AND \ \overline{A_1})$. The second and the third stages are made up by having each group of four and 16 distinct values is combined altogether, respectively.

For the design of the population-count $\sum$ module, there were a number of different versions of the design because of the bit/item scenarios. For example, if the $N$-value equals to eight, a $32 \times 256$ $I_{x,y}$-matrix will be created,
thus leading to the need for 256 $\sum_{-32}^{6}$ modules. Then, each $\sum_{-32}^{6}$ module is applied to a row of the $32 \times 256$ $I_{x,y}$-matrix to sum all of the 32 individual values to a 6-bit sum-result; the 6-bit sum-result holds the value from 0 to 32. Fig. 4 gives the details of the $\sum_{-32}^{6}$ module architecture. For other bit/item design options, different architectures of the $\sum$ modules are applied, but they all share the same design idea shown in Fig. 4. The design of the $\sum$ module in this paper aims to optimize both area and timing performances. As given in Fig. 4, the proposed architecture utilized only Full Adder (FA) and Half Adder (HA) in a parallel setup.

3 Experimental Results

3.1 The FIC Core Module Results

The proposed FIC architecture was implemented in 35 different versions corresponding to the number of bit/item and count-register bit-width. All 35 implementations were built on the FPGA board of Altera Arria V SoC development kit with the FPGA chipset of 5ASTFD5K3F40I3. The report of logic utilization is the number of Adaptive Logic Modules (ALM). ALM is the fundamental block of Altera’s FPGAs. An ALM of an Arria V FPGA contains an 8-bit Adaptive Look-Up Table (ALUT) and four registers [12].

Fig. 5: The ALM (a) and $F_{\text{Max}}$ (b) results on FPGA.

Fig. 6: The MIPS (a) and ALM/MIPS ratio (b) results on FPGA.
Fig. 5 and Fig. 6 show the experimental results of 35 FPGA implementations. The ratio of ALM/MIPS was also presented in the semi-log graph of Fig. 6(b) to analyze the trade-off between timing performances and resources costs. The ratio represents the quality of an FPGA implementation; a better implementation comes with a smaller ratio result. According to the figures, it is clear that when increasing the count-register bit-width or the number of bit/item, the timing performances were dropped and the logic utilizations were increased. As can be seen in Fig. 5(a), the changes of logic utilizations were almost linear. To be specific, the ALM results were increased about $1.82 \times$ and $1.08 \times$ for one more bit/item and four more bits of count-register bit-width, respectively. For the timing performances, although the maximum operating frequencies ($F_{\text{Max}}$) in Fig. 5(b) fluctuated non-linearly, the MIPS results in Fig. 6(a) were almost linear. For one more bit/item and four more bits of count-register bit-width, the MIPS values were decreased by about 900 and 200, respectively. For the quality ratio of ALM/MIPS, according to Fig. 6(b), it is clear that the changes were also almost linear with the $2.18 \times$ and $1.13 \times$ increments by one more bit/item and four more bits of count-register bit-width, respectively.

After FPGA implementation, the 35 FIC core modules were synthesized at the ASIC level using the SOTB-65nm technology. The synthesis results were reported by the Synopsys tools, and they are plotted in Fig. 7, Fig. 8, and Fig. 9. The results of area and power were received at the IC compiler reports. The $F_{\text{Max}}$ results were retrieved by testing those layouts including their parasitic delay information with the PrimeTime tool. For the MIPS results, because the layouts were not fabricated into actual chips, they did not have the practical MIPS results as the FPGA implementations. Therefore, the MIPS results in Fig. 8(b) were theoretically calculated. The theory MIPS result is computed by multiplying the corresponding $F_{\text{Max}}$ value with the number of input items that the FIC core module can receive by each clock. To visualize the trade-off between the timing performances, the area costs, and the power consumptions of the SOTB-65nm layouts, the two ratios of area/MIPS and power/MIPS were proposed in the semi-log graphs of Fig. 9(a) and Fig. 9(b), respectively.

Fig. 7: The area (a) and power (b) results on SOTB-65nm.
Fig. 8: The \( F_{\text{Max}} \) (a) and MIPS (b) results on SOTB-65nm.

According to the figures, when increasing the number of bit/item and counter-register bit-width, the results of area and power were increased, and the performances of \( F_{\text{Max}} \) and MIPS were decreased. To be specific, both area and power results were increased about 1.74\( \times \) and 1.09\( \times \) for one more bit/item and four more bits of count-register bit-width, respectively. On the other hand, for one more bit/item and four more bits of count-register bit-width, the timing performances were decreased by about 630 MIPS and 70 MIPS, respectively. For the two quality ratios plotted in Fig. 9(a) and Fig. 9(b), their changes throughout all layouts were almost the same and linear. Therefore, it can be said that both ratios were increased about 2.06\( \times \) and 1.12\( \times \) when adding one more bit/item and four more bits of count-register bit-width, respectively. Comparing to the FPGA results, the SOTB-65nm syntheses gained about 75\% of the MIPS results.

3.2 Comparison with Other Recent Works

Table I gives the comparison between other recent works and the proposed architectures at the 32-bit counters option. In the table, to make a fair comparison, the \( F_{\text{Max}} \) results of [13] and [14], which were respectively built on a 65nm FPGA and a 40nm FPGA, were scaled into equivalent 20nm chips’ performances by using the scaling equations from work [15]. After that, their MIPS results were also scaled proportionally according to their scaled \( F_{\text{Max}} \).
values. The reason for the 20nm scaling is because the work [15] did not have the 28nm scaling coefficients; therefore, the 20nm scaling results were selected instead. Furthermore, because the work [15] also did not have the 40nm scaling coefficients, the scaling result of the work [14] in Table I was the mean value of the two scaling results from the 45nm and 32nm scaling coefficients.

The implementations in 2010 [13] and 2011 [14] were based on an approximation design approach; thereby they were different with direct computation implementations like the work in 2017 [16] and our work. A direct computation method can produce all of the FIC results correctly, while an approximation method can answer only a small number of top-count results approximately. Because of that, the performances of an approximation design did not depend on the number of item types, while those of a direct computation design profoundly did. In other words, direct computation architectures such as the work [16] and our work had to rely on the number of bit/item, while such approximation implementations like [13] and [14] had not to. To conclude, for the comparison, performances of a direct computation design are compared with those of an approximation design at the same number of output FIC results.

According to Table I, it is clear that the proposed architecture achieved the outstanding MIPS results in comparison with other works. The proposed architecture respectively had more than 4,900 MIPS and 2,500 MIPS for the 6-bit/item and 10-bit/item design options, while the work [16] achieved only 1,280 MIPS for the 8-bit/item configuration, and the works [13] and [14] cannot reach to 200 MIPS for all of their settings.

For the resources cost comparison, the differences between Xilinx’s and Altera’s FPGAs must be clarified first. The fundamental resources block of Altera’s and Xilinx’s FPGAs is the ALM and slice, respectively. For Arria V FPGAs, an ALM contains four registers and one 8-bit ALUT [12]. For Virtex-5 FPGAs and Virtex-6 FPGAs, a slice contains four 6-bit Look-Up Tables (LUT) and four registers [17], and four 6-bit LUTs and eight registers [18], respectively. As a result, it can be said that a Xilinx’s LUT roughly equals to 3/4 of an Altera’s ALUT. Hence, a Xilinx’s slice roughly equals to three Altera’s ALMs in term of LUTs. Also, in term of registers, one slice of Virtex-5 and Virtex-6 equals to one and two Arria V’s ALMs, respectively. To summarize, based on the above analyses, it is clear that the proposed implementations achieved the best resources costs in Table I compared to other works at the same number of output FIC results.

4 Discussion

FIC designs have two main approaches of approximation and direct computation. The main advantage of the approximation approach is a large number of item types processing, while that of the direct computation one is a completed FIC output results. Most software FICs favored an approximation approach [4–8]. The reason is that, besides the main advantage, an approximation software FIC also gives a short execution time. On the other hand,
Table I: The results of the 32-bit count-register FPGA implementations in comparison with other recent works.

| Design       | [13] (2010) | [14] (2011) |
|--------------|-------------|-------------|
| Approach     | Approximation |            |
| Method       | Pipelined model |         |
| FPGA         | Virtex-5 (65nm) | Virtex-6 (40nm) |
| bit/item     | 32           |            |
| counters     | 32           |            |
| Number of output results | 32 | 64 | 128 | 64 | 128 | 256 | 512 | 1,024 |
| LUT          | 8,720       | 16,887     | 32,023     | 16,846     | 34,108     | 71,302     | 135,424     | 269,285     |
| Register     | 8,312       | 6,880      | 12,033     | 5,377      | 10,529     | 20,804     | 41,441      | 82,657      |
| Slide        |             |            |            |            |            |            | 69,433      |            |
| $F_{\text{Max}}$ |             |            |            |            |            |            |            |            |
| Origin:      | 80          | 110        | 115        | 105        | 110        | 140.28     | 121.15      |            |
| Scaled:      | 185.33      | 140.28     | 146.28     | 133.90     | 140.28     | 121.15     |            |            |
| MIPS         |             |            |            |            |            |            |            |            |
| Origin:      | 80          | 110        | N/A        | 105        | N/A        |            |            |            |
| Scaled:      | 185.33      | 140.28     | N/A        | 133.90     | N/A        |            |            |            |

| Design       | [13] (2010) | [16] (2017) | Proposed (2018) |
|--------------|-------------|-------------|-----------------|
| Approach     | Approximation | Direct computation |           |
| Method       | Pipelined model | Equality comparator | Binary decoder |
| FPGA         | Virtex-5 (65nm) | Arria V SoC (28nm) |             |
| bit/item     | 32           | 8           | 6              | 7             | 8             | 9             | 10           |
| counters     |              |             |                | 32            |              |               |              |
| Number of output results | 256 | 256 | 64 | 128 | 256 | 512 | 1,024 |
| LUT          | 62,260       | N/A         |               |              |              |               |              |
| ALUT         | N/A          | 51,094      | 10,312       | 19,246       | 36,692       | 64,709       | 83,709       |
| Register     | 22,335       | 8,417       | 4,916        | 8,890        | 16,578       | 30,914       | 58,563       |
| Slide        | N/A          | N/A         | 45,258       | 7,202        | 13,436       | 25,990       | 47,055       | 80,975       |
| ALM          |              |             | 45,258       | 7,202        | 13,436       | 25,990       | 47,055       | 80,975       |
| $F_{\text{Max}}$ |             |            |              |              |              |              |            |            |
| Origin:      | 80          | 110        | 115        | 105        | 110        | 121.15     |            |            |
| Scaled:      | 185.33      | 140.28     | 146.28     | 133.90     | 140.28     | 121.15     |            |            |
| MIPS         |             |            |            |            |            |            |            |            |
| Origin:      | 80          | 110        | N/A        | 105        | N/A        |            |            |            |
| Scaled:      | 185.33      | 140.28     | N/A        | 133.90     | N/A        |            |            |            |
because the direct computation approach was hard to be implemented in parallel on software, its software implementations usually present a poor timing performance. Therefore, the majority of software FICs were based on an approximation method [4–8]. For hardware FIC, there were several attempts to make a hardware implementation recently. However, those implementations were usually a further development from its software’s achievements [13,14]. As a consequence, those designs also relied on an approximation architecture rather than a direct computation one. Thus, those architectures failed to utilize the hardware nature advantage of parallel processing. For this reason, a direct computation hardware FIC architecture such as the proposed architecture in this paper could exploit the parallel advantage of hardware to make an excellent timing performance.

As seen in Table I, it was clear that the proposed architecture achieved the outstanding MIPS results. However, the primary drawback of the design, also of the direct computation approach, was the dependence on the number of item types, or the number of bit/item. Because of this, the applications range was limited. However, depending on specific requirements, the proposed architecture still can satisfy some database applications. Furthermore, the proposed architecture is the best suit for those applications that need a high-speed performance rather than a large number of item types processing. For instance, such applications are the image histogram application, which needs 8-bit or 10-bit per color-plane, and the application of DNA-sequence statistic analysis, which requires only 3-bit/item for the five scenarios of nucleobases, i.e., C, G, A, T, and null.

According to the ALM changes in Fig. 5(a), the ALM result of the architecture at the settings of 12-bit/item and 32-bit counters can be estimated to 262,000 ALMs. Therefore, 12-bit/item versions of the proposed architecture can be implemented on those FPGAs like Arria V, Arria 10, Stratix V, and Stratix 10. Moreover, a multi-FPGA system can be developed to increase the number of bit/item further. Besides the overwhelming timing results, such a multi-FPGA system still has the economic benefits compared to a conventional software system that based on a sever [6,8] or multiple PCs [7].

5 Conclusion

High-speed direct-computation FIC architecture using binary-decoder was proposed in this paper. The architecture was implemented with various options of bit/item and count-register bit-width. All implementations were built on the two platforms of FPGA kit and ASIC with the SOTB-65nm process. The experimental results showed that the proposed architecture achieved the timing performances exceed far more than those of other recent works.

Acknowledgement

This work is supported by VLSI Design and Education Center (VDEC), the University of Tokyo in collaboration with Synopsys, Inc., and Cadence Design Systems, Inc.