High-speed Hardware Implementation of 8-bit per Item Frequent Items Counter

Katsumi Inoue*, Trong-Thuc Hoang†, Xuan-Thuan Nguyen†, Hong-Thu Nguyen†, and Cong-Kha Pham‡
*Advanced Original Technologies Co., Ltd (AOT), Tokyo, Japan
†‡The University of Electro-Communications (UEC), 1-5-1 Chofugaoka, Chofu-shi, Tokyo 182-8585, Japan
Email: *ugg44151@nifty.com; †{thuc, xuanthuan, hongthu}@vlsilab.ee.uec.ac.jp; ‡phamck@uec.ac.jp

In this paper, the high-speed architecture of Frequent Items Counting (FIC) is proposed. FIC is a problem of counting frequently appeared items in the itemset. The task is a must-have function in almost every data mining algorithms such as frequent elements [1], iceberg queries [2], and top-k queries [3]. For related works, the space-saving method was the primary method used to solve the FIC problem in software applications [1], [3]. It is an approximation method with the idea of selecting and monitoring only a few best candidates. The algorithm was also implemented in hardware as in [4]. However, due to the approximation approach, the architectures in [4] cannot produce the full FIC table. Therefore, the goal of the proposed FIC architecture in this paper is to produce the completed FIC table. Hence, the proposed implementations did not deploy an approximation method such as space-saving algorithm, but the tuple-scan approach. The tuple-scan approach can produce the completed FIC table in a single pass of itemset by maintaining an array of count-register.

The overview of the proposed Field Programmable Gate Array (FPGA) system is given by Fig. 1. The database processor, the memories, the direct memory access (DMA), and the FIC IP controls the overall operation, store the itemset, transfers data, and executes the FIC task, respectively. The chosen FPGA board was the Altera Arria V SoC development kit with the 5ASTFD5K3F40I3 chip series. The kit had the limitation of 256-bit data-width, thus resulting in 256-bit data-width of the FIC IP and its DMA. Therefore, with 8-bit/item, the FIC IP can process 32 items by each clock. Fig. 2 describes the architecture of the FIC core module. The key idea is to use binary decoders to generate a matrix of binary values, with each column represents for one input items binary value. Then, the rows of the matrix are summed by the population-count $\sum$ modules to retrieve the input items counting results. Finally, the current 256 counting results are added up with the previous results from the count-register array. There are seven options of count-register bit-width from 8-bit to 32-bit counters as can be seen in Table I. Fig. 3 gives the design of an 8-to-256 binary decoder. The binary decoder works in the same way of an address decoder in a conventional RAM design. With 8-bit input data from 0 to 255, the 256-bit output is generated with only one asserted bit and 255 de-asserted bits as shown in the truth table in Fig. 3. With 8-bit input data, the binary decoder design is divided into three stages of the combination as seen in Fig. 3.

After successful implemented on FPGA, the FIC core modules were further synthesized by Synopsys tools with the process library of SOTB (Silicon On Thin Buried-oxide) 65nm. The experimental results were reported in Table I along with others results of our previous work [5] and an equivalent software. The software was based on the well-known R-Studio tool, and it can produce full FIC table using the function `count()` in the `plyr` library with no other optimizations and runs on one core of CPU. The ASIC results in Table I were extracted after the IC Compiler step. The counting speed of Million Items Per Second (MIPS) of FPGA systems were practically mesured, while those of the ASIC circuits were theoretically calculated based on their F_{max}.

As can be seen in Table I, it is clear that with the smaller count-register bit-width option, the fewer resources costs and the better timing performances. For specific, when changing from 8-bit to 32-bit counters option, FPGA results had the costs of Adaptive Look-Up Tables (ALUTs) and registers increased about 35% and 60%, respectively; while their F_{max} and average MIPS results decreased about 16%. The SOTB-65nm builds achieved the speed about 75% of the FPGA results. FPGA's results in comparison with the previous work [5] at the same 32-bit count-register setting, the ALUTs requirement was decreased about 30%, the registers number was increased about twice, and the average MIPS was about three times better. In comparison with the software, the 8-bit and 32-bit counters versions of FPGA implementations gain the better MIPS performances about 185 times and 155 times, respectively. To conclude, the achieved throughput rates in this paper were far more than those of the best-published results in both terms of software and hardware implementations.

References

[1] S. Das *et al.*, "Thread Cooperation in Multicore Architectures for Frequency Counting Over Multiple Data Streams," in *Int. Conf. on PVLDB*, vol. 2, no. 1, Aug. 2009, pp. 217–228.
[2] K. AlSabti, "Efficient Computing of Iceberg Queries Using Quantiling," *Journal of King Saud Univ. - Comp. and Info. Sciences*, vol. 18, pp. 53–75, 2006.
[3] A. Metwally *et al.*, "An Integrated Efficient Solution for Computing Frequent and Top-k Elements in Data Streams," *ACM Trans. on Database Systems*, vol. 31, no. 3, pp. 1095–1133, Sep. 2006.
[4] J. Teubner *et al.*, "Frequent Item Computation on a Chip," *IEEE Trans. on Knowledge and Data Engi.*, vol. 23, no. 8, pp. 1169–1181, Aug. 2011.
[5] Trong-Thuc Hoang *et al.*, "FPGA-based Frequent Items Counting Using Matrix of Equality Comparators," in *IEEE Int. Midwest Symp. on Cir. and Sys. (MWCAS)*, Boston, USA, Aug. 2017, pp. 285–288.

TABLE I: The experimental results of the proposed FIC architecture with FPGA implementations and SOTB-65nm synthesis.

FPGA implementation (Altera Arria V SoC: 5ASTFD5K3F40I3)								
	Proposed architecture (Binary Decoders)							Previous work [5] (Equality Comparators)
Count-Register	8-bit	12-bit	16-bit	20-bit	24-bit	28-bit	32-bit	32-bit
ALUTs	27,290	29,164	30,095	31,625	32,225	33,155	36,692	51,094
Registers	10,458	11,478	12,498	13,518	14,538	15,558	16,578	8,417
F_{Max} (MHz)	149.44	137.75	137.02	134.47	130.60	125.30	125.10	40.85
Average MIPS	4,638.62	4,275.76	4,253.1	4,173.95	4,053.82	3,889.31	3,883.1	1,280

SOTB-65nm synthesis								Software-based (R-Studio)	
Count-Register	8-bit	12-bit	16-bit	20-bit	24-bit	28-bit	32-bit		
Standard cells	70,349	75,290	82,178	91,046	98,649	116,919	125,279	**Platform**	Windows 10 PC, 4GHz Core-i7 Intel CPU, 32GB RAM
Area (μm×μm)	569×551	595×580	622×605	648×630	674×652	709×691	732×713		
Power (mW)	7.3178	8.1773	9.1761	9.9226	11.0924	11.6577	12.5734		
Number of CMOS	813,595	889,099	975,160	1,050,372	1,134,045	1,232,400	1,348,297		
F_{Max} (MHz)	105	97	97	96	95	94	94	**Average MIPS**	25
Theoretically MIPS	3,360	3,104	3,104	3,072	3,040	3,008	3,008		

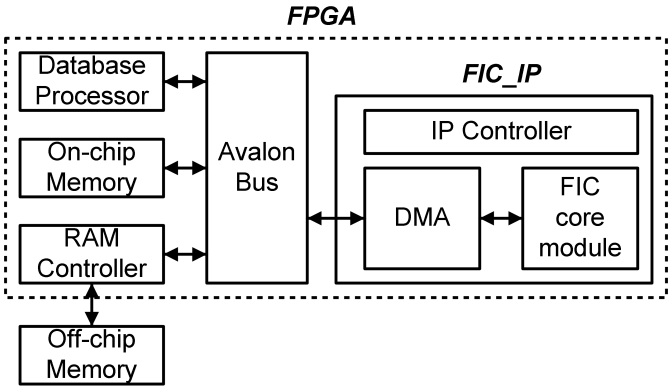

Fig. 1: The system overview of the proposed FIC implementation on FPGA.

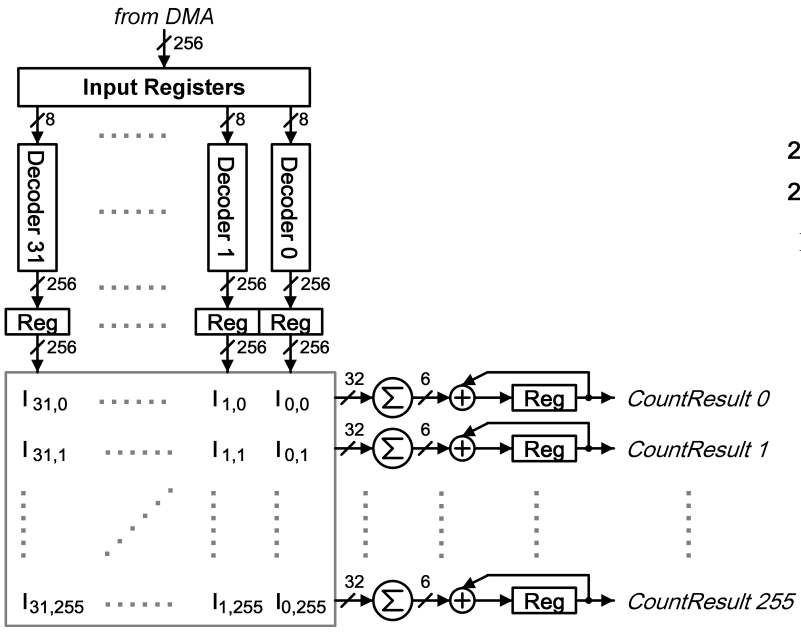

Fig. 2: FIC core module architecture.

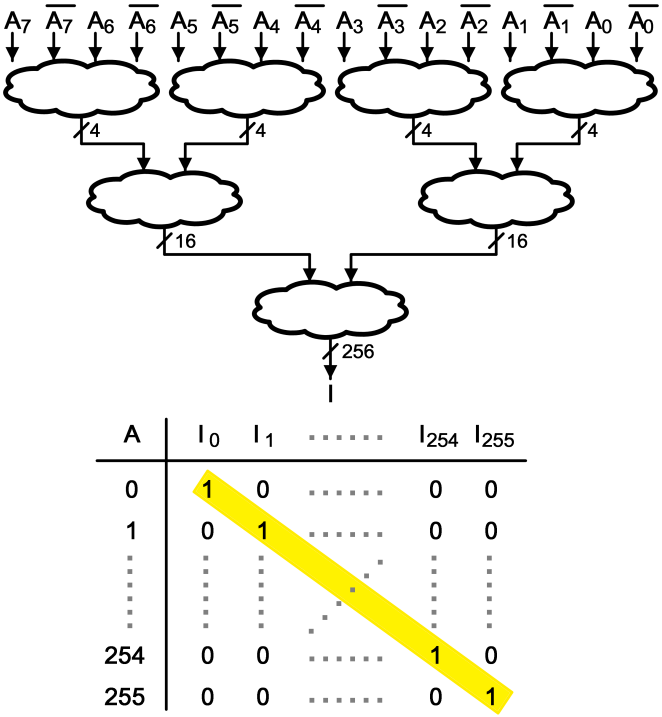

Fig. 3: 8-to-256 binary decoder design.