High Performance Implementation of Bloom Filter on an FPGA for Database Applications

STUDENTS / UNIVERSITIES
Hakan Buğra Erentuğ / Sabancı University
Mert Tunç / Bilkent University
Sara Jiyan Sürer / Kadir Has University
Alperen Doğan / Sabancı University

SUPERVISOR(S)
Erdinç Öztürk
Kadir Eren Ünal

Abstract

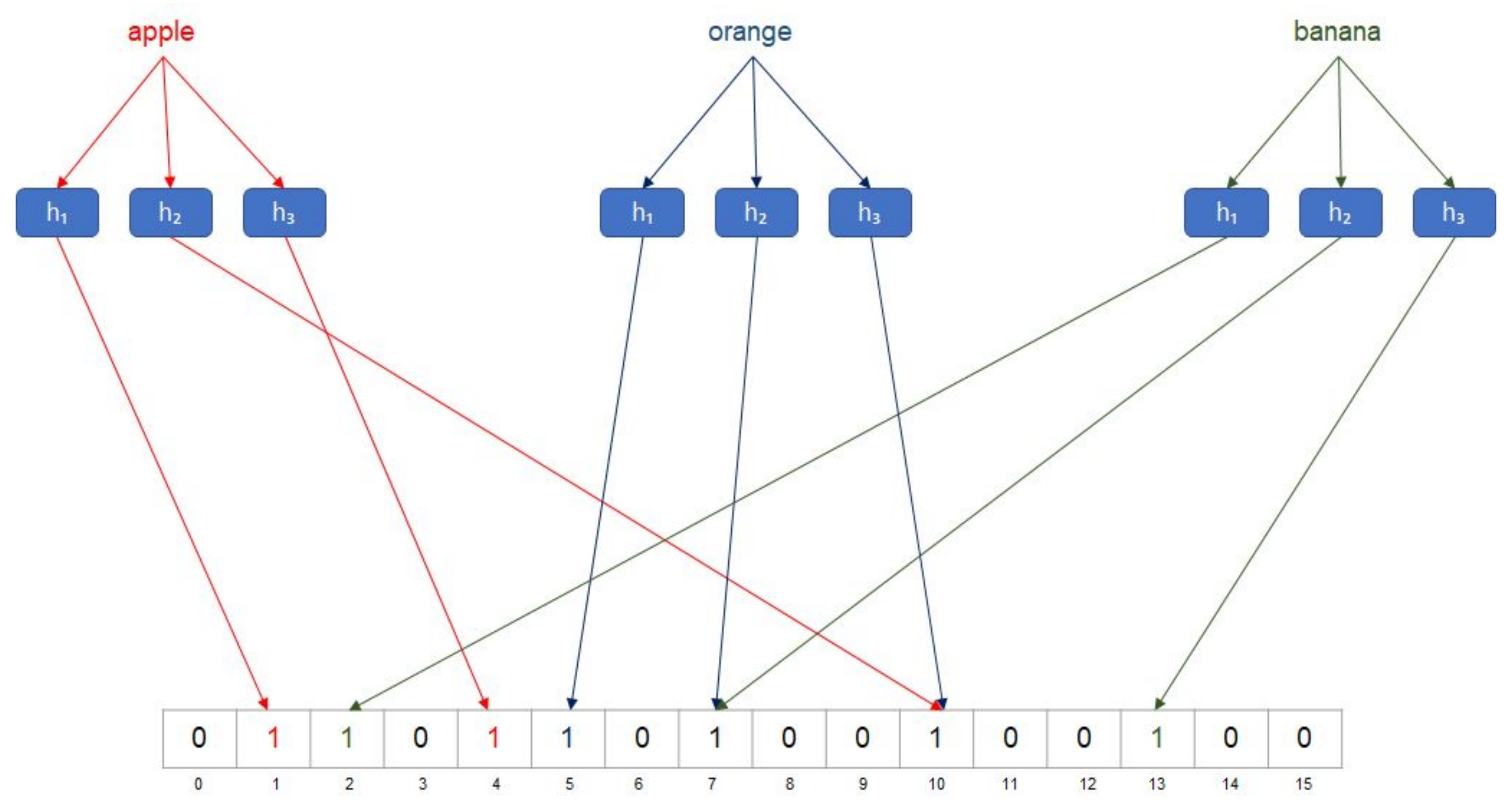

Figure-1

As the database systems increase their performance to meet the demand of today's network and storage applications, the need for fast and memory efficient database algorithms also become more significant. Bloom filters are probabilistic data structures that perform fast membership queries with sub-linear memory complexities. They are used in string matching, deep packet inspection and web cache applications. The fact that bloom filter mostly uses binary operations enables us to utilize the FPGA for high performance implementations compared to commercial processors. In our design, we have used partial bloom filters which resulted in lower false positive probability and less clock cycles required to execute query operations with a trade-off in memory. The parallel nature of the FPGA allowed us to achieve greater performance by using multiple bloom filters. As a result of the designed architecture, we have obtained an estimated throughput of 1.81 Gbps with merely 2.5 Kilobyte memory consumption.

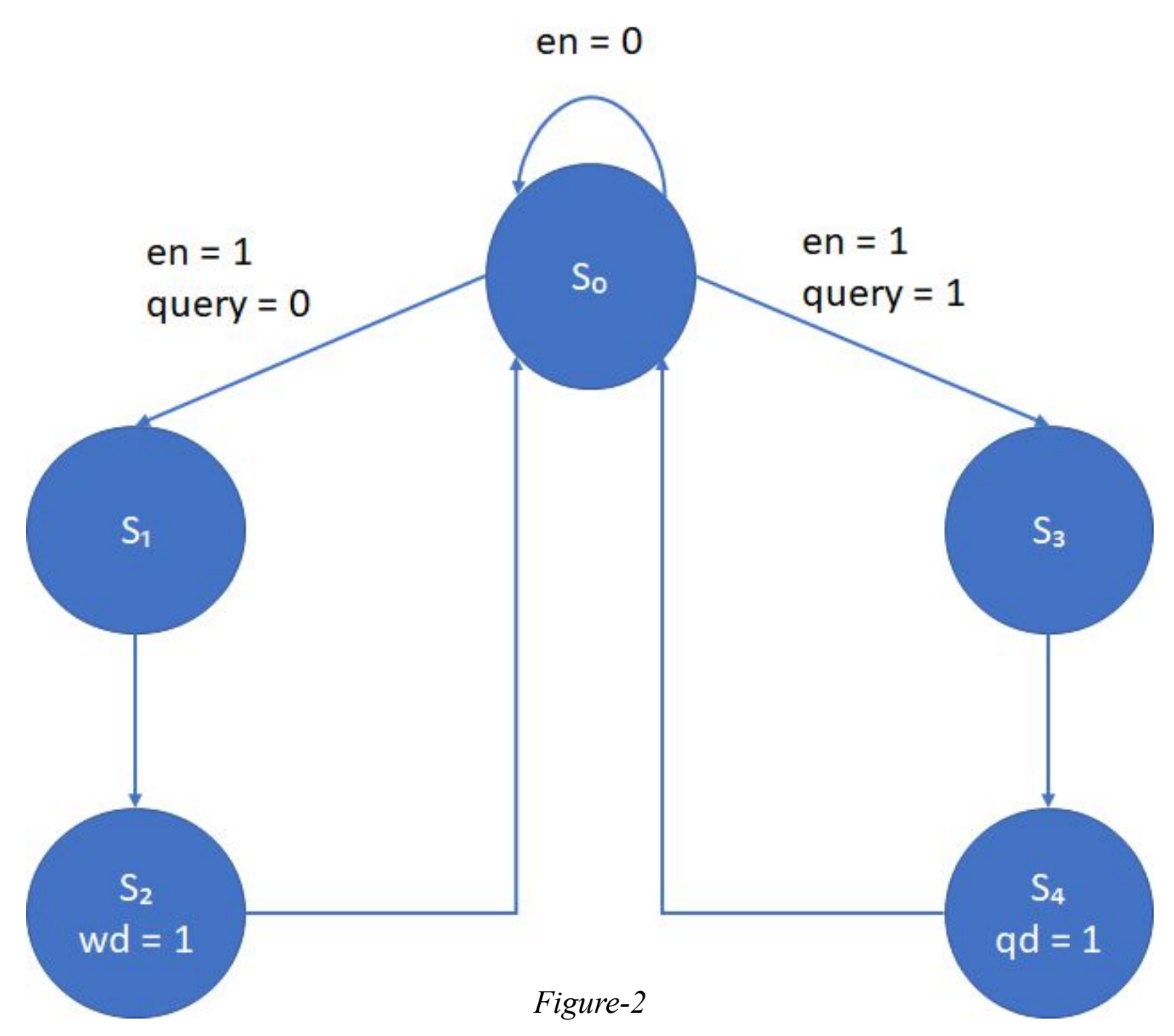

Figure-2

Objectives

- The aim of this project is to implement a high performance bloom filter on the FPGA environment for database applications.
- The implemented design will have a high throughput.

Project Details

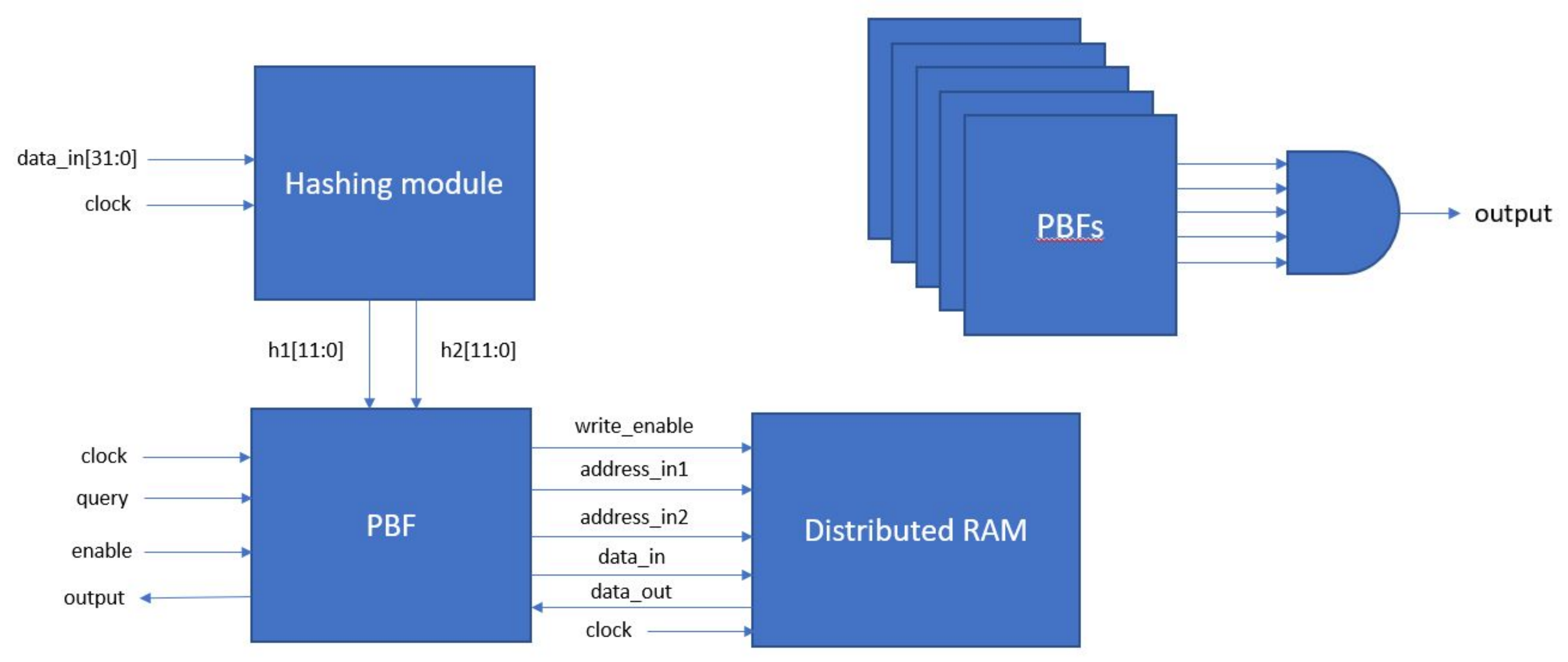

Figure-3

The project consists of three main pieces, hashing module, partial bloom filters and distributed RAM modules. This project is implemented on a NEXYS4 DDR FPGA.

The project utilizes partial bloom filters (PBF) to achieve high performance which is a commonly used technique in string matching designs. Having multiple bloom filters results in a lower false positive probability and less number of clock cycles to do a query operation but it requires additional memory space.

Project Details II

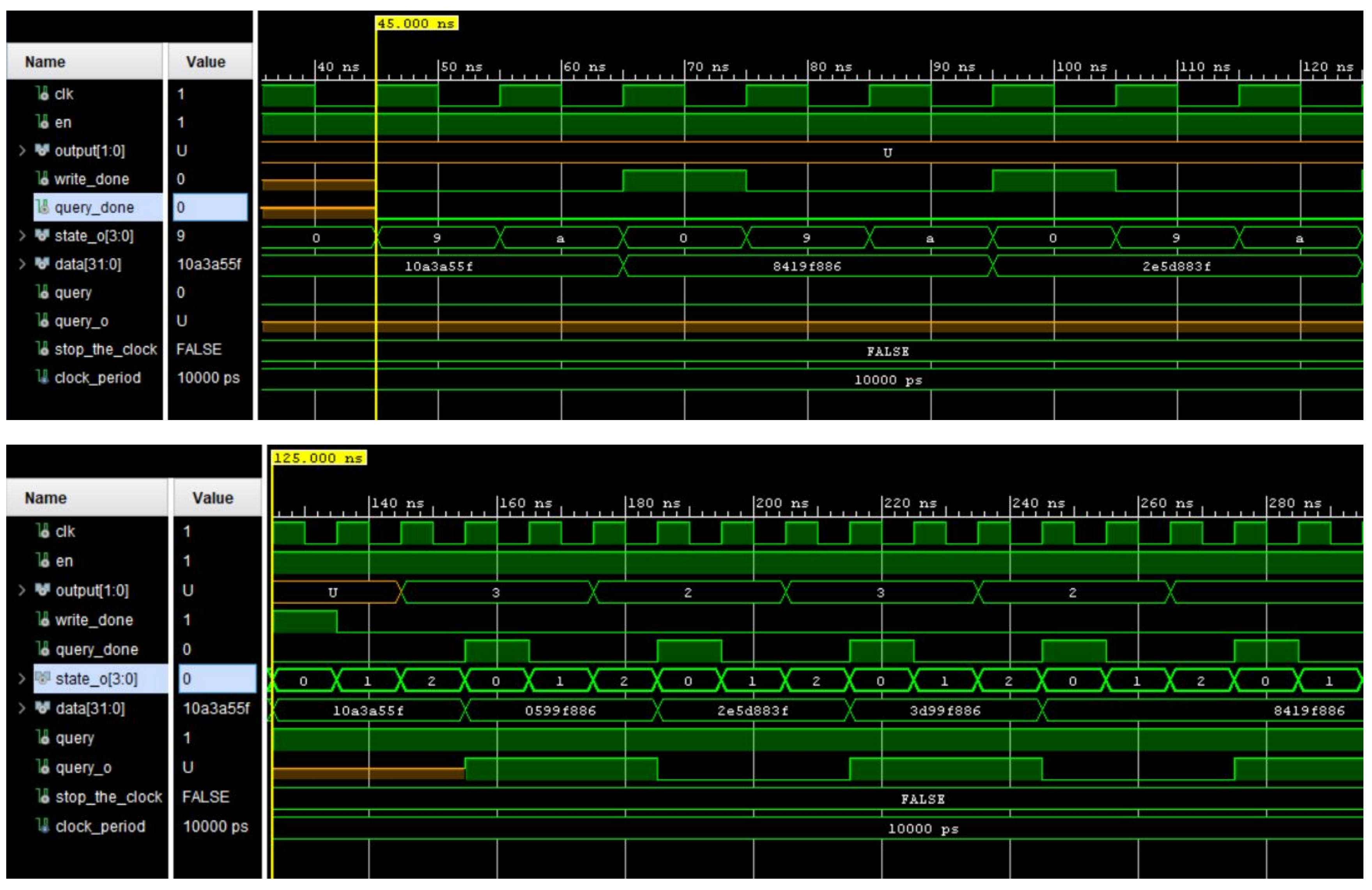

Figure-4

A single PBF in our implementation has the length of 4096 bits. It uses two hash functions to map the incoming data to the bit array. Bit extraction hashing is used due to its performance on hardware implementations as the hashing operation can be completed within the clock cycle.

When the number of elements inserted to the PBF is around 1400, the false positive probability of the PBF will be about ¼. This project uses five of the said PBFs and combines their outputs. Therefore, the false positive rate of the whole filter will be 0.1%.

Each PBF has its own dual port distributed RAM that is made of LUTs. With this configuration both insertion and query operations take 3 clock cycles.

Implementation results show that the design uses 1004 LUTs and 227 FFs which is about 2% of the onboard resources. The implementation runs at a clock frequency of 170 MHz which was increased from 162 MHz after basic floorplanning. Together with 32-bit input and the 3 clock cycles required for a query operation, the throughput of the implementation is about 1.81 Gbps.

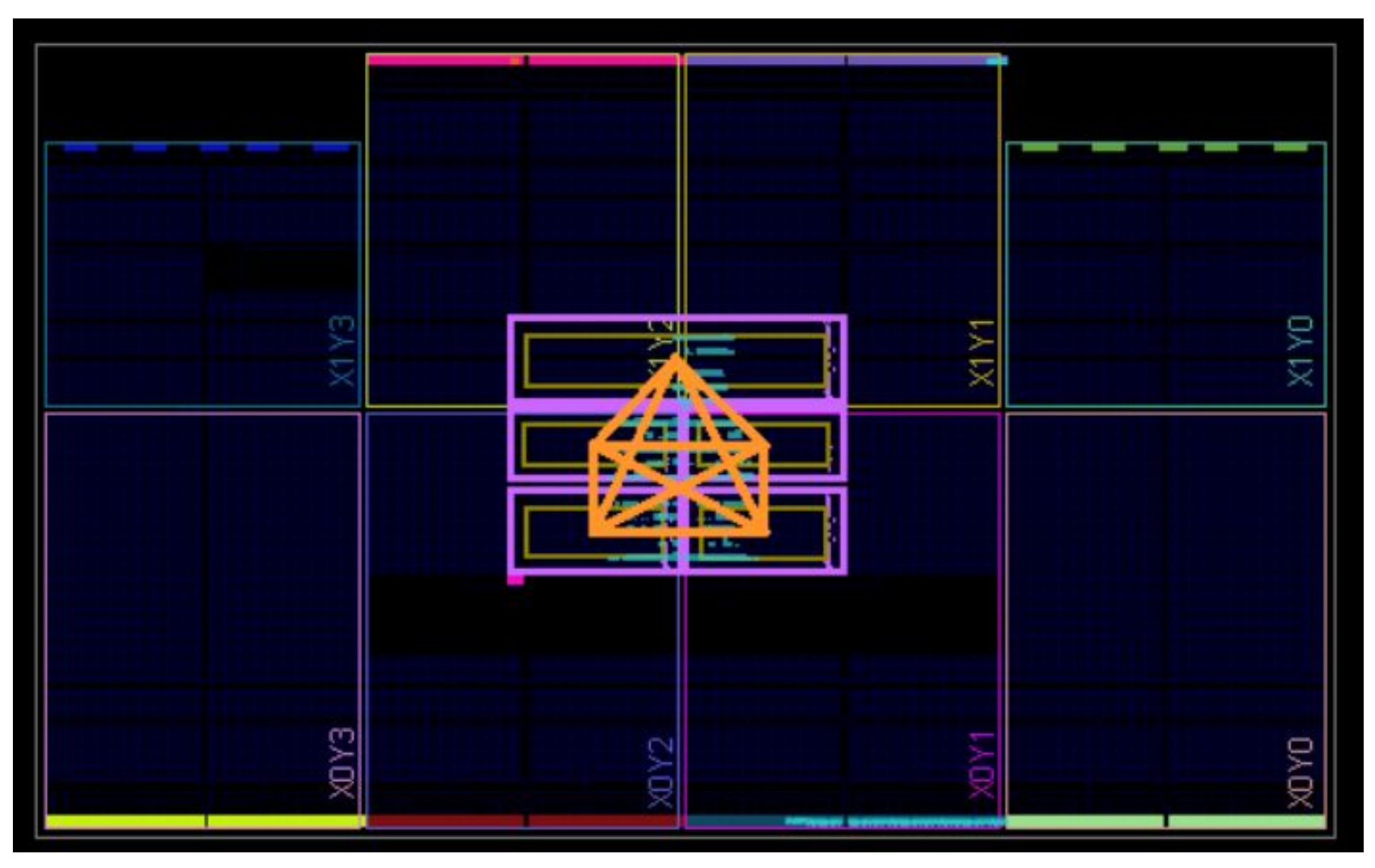

Figure-5

Conclusions

In this project we have implemented a high performance bloom filter on an FPGA. The bloom filter reached a false positive possibility of 0.1% when more than a thousand items were inserted.

Partial bloom filters have been used to reduce the number of memory accesses to a single RAM module during a query operation which improves the performance. Our implementation achieved an estimated throughput of 1.81 Gbps. Based on this project, the performance of different hash functions, database aggregation algorithms and memory structures can be compared in future research projects.

Since the interface options on the NEXYS board do not support the achieved throughput, a different FPGA with PCIE support can be used to test our implementation on realistic conditions. Furthermore, the performance of this project can be further improved by the implementation of a more complex floorplanning.

References

- Tong, D., & Prasanna, V. K. (2018). Sketch Acceleration on FPGA and its Applications in Network Anomaly Detection. *IEEE Transactions on Parallel and Distributed Systems*, 29(4), 929-942. doi:10.1109/tpds.2017.2766633
- Lyons, M. J., & Brooks, D. (2009). The design of a bloom filter hardware accelerator for ultra low power systems. *Proceedings of the 14th ACM/IEEE International Symposium on Low Power Electronics and Design* - ISLPED 09. doi:10.1145/1594233.1594330
- Dharmapurikar, Sarang; Attig, Michael; and Lockwood, John. (2004). Design and Implementation of a String Matching System for Network Intrusion Detection using FPGA-based Bloom Filters. *All Computer Science and Engineering Research*. Report Number: WUCSE-2004-12 .
- Fu, E., Ramakrishna, M., & Bahcekapili, E. (1997). Efficient Hardware Hashing Functions for High Performance Computers.