Novel Bloom filter algorithms and architectures for ultra-high-speed network security applications

Arish Sateesan, Jo Vliegen, Joan Daemen, Nele Mentens

Introduction

A Bloom filter is a memory structure that is used to check whether input search data are present in a table of stored data (Lookup) and are extensively used in network security solutions that apply traffic flow monitoring or deep packet inspection.Speed of lookup affects the speed of many network applications.

The most important components determining the speed of Bloom filters are (1) Hash functions (Non-cryptographic hashes not preferred) and (2) Number of memory accesses (Limited or No use of external memory).

We take on the challenge of developing ultra-high-speed Bloom filters on FPGAs by proposing a new noncryptographic hash function, called Xoodoo-NC, derived from the cryptographic permutation Xoodoo. We adopt Bloom-1 architecture for a low latency bloom architecture.

Challenges

⇒Slower lookup architectures cause mismatch in throughput between the router and the network.
⇒High processing overhead not always mean fast. More area, power, and low frequency on hardware. ⇒Present architectures are fast?

CAM: Low latency, low operating frequency. Operating frequency of CAMs is not really suitable for Terabit networks.

Bloom Filters: High latency, high operating frequency. Low memory footprint but high computation overhead and presence of false positives. No. of memory accesses $\propto$ No. of Hash functions (k).

⇒The computation overhead of hash functions must also be reduced without adversely affecting the avalanche properties.

References

[1] Y. Qiao, T. Li, and S. Chen. Fast bloom filters and their generalization. *IEEE Transactions on Parallel and Distributed Systems*, 25(1):93–103, 2014.

[2] Joan Daemen, Seth Hoffert, Gilles Van Assche, and Ronny Van Keer. The design of xoodoo and xoofff. *IACR Transactions on Symmetric Cryptology*, 2018(4):1–38, Dec. 2018.

Algorithms & architectures

A high-speed non-cryptographic algorithm Xoodoo-NC is proposed and integrated to Bloom-1 [1] architecture.

Xoodoo-NC: Xoodoo-NC is a reduced-round,reduced-state version of Xoodoo [2], inheriting Xoodoo's desired avalanche properties, resulting in an ultra-low-latency non-cryptographic hash function.

⇒Xoodoo-NC is faster, thanks to low logical depth.

⇒Output size can be any multiple of 96 by concatenating each round outputs after 2 rounds.

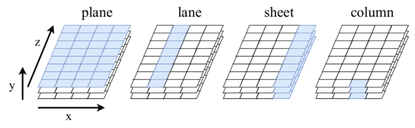

Figure 1: Xoodoo graphical representation

Xoodoo	Xoodoo-NC
384-bit state	96-bit state
x=4, y= 3, z=32	x=1, y=3, z=32
No. of rounds=12	No. of rounds=2 to 3

Table 1: Xoodoo Vs Xoodoo-NC

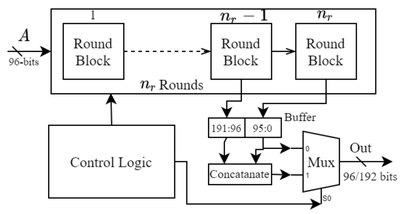

Figure 2: Xoodoo hardware block diagram

Bloom-1: A fast Bloom filter architecture Proposed by Y. Qiao et al., that only requires one read-out for each query and update operations.

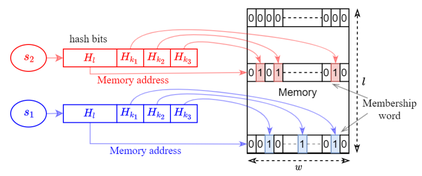

Figure 3: Bloom-1 representation

Algorithms & architectures

⇒Bloom-1 uses a single hash function ($log_2 l + k.log_2 w$) (Xoodoo-NC in our implementation).
⇒2-Dimensional memory array with all hash values mapped to a single membership word.
⇒Frequency of operation and latency doesn't change with k.

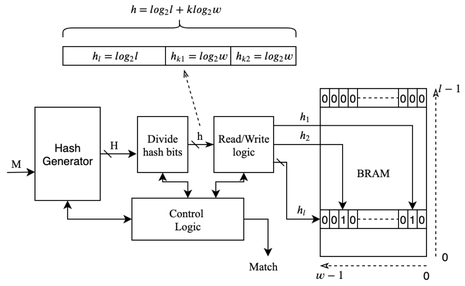

Figure 4: Bloom-1 hardware block diagram

Results & Analysis

⇒Experiments conducted on a Xilinx Virtex Ultrascale+ FPGA (XCVU7PFLVC2104-1-E)
⇒**Testing parameters:** Flow ID-96 bits
⇒**Hash functions used:** FNV-1a (64-bit & 128-bit, Pipelined and non-pipelined) and Xoodoo-NC (96-bit & 192-bit, Number of rounds 2 or 3)
⇒**Bloom-1 Parameters:** l=4096, w=64, m=262144
⇒Xoodoo-NC provides full bit-dependence and quasi-strict avalanche weight after 2.5 rounds, and ≈ 90% dependence after round 2.

Results & Analysis

Rounds	D_{av}	$\overline{w}_{av}$	H_{av}
2	84	35.408	80.332
2.5	96	47.324	95.864
3	96	47.309	95.867

Table 2: Avalanche scores of Xoodoo-NC

⇒Xoodoo-NC based bloom-1 outperforms the Xilinx IP CAM in terms of latency, resource utilization, and operating frequency for inserting 1024 elements(Fig 5).
⇒ Comparing with the Bloom-1 implementation with FNV-1a, Xoodoo-NC based implementation significantly better than the other 5).
⇒ This implementation has a latency of 3 clock cycles for update operation consisting of only one memory access, which is the lowest latency possible among available hardware implementations of bloom filters to the best of our knowledge.
⇒The proposed implementation has a lookup delay of 6.49ns, which makes Xoodoo-NC based bloom-1 the the fastest lookup architecture to the best of our knowledge.

Design	k	fpr	LUTs	FFs	BRAM	DSP	Max freq [MHz]	Lookup Delay [ns]
Xoodoo-NC	**12**	**$2.61x10^{-7}$**	**675**	**158**	**7.5**	**0**	**462.32**	**6.49**
FNV-1a	12	$2.61x10^{-7}$	1366	339	7.5	15	121.18	132.03
FNV-1a Pipelined	12	$2.61x10^{-7}$	2706	1562	7.5	180	104.87	38.14
CAM		0.0	6731	1145	320	0	112.75	17.74

Figure 5: Comaprison of Bloom-1 & CAM

Conclusion

⇒We targeted novel algorithms and architectures for high-speed Bloom filters on FPGA, used for fast lookups in network (security) applications.
⇒A new high-speed hardware-oriented non-cryptographic hash function called Xoodoo-NC is proposed and is integrated into a Bloom-1 architecture and evaluated on a Virtex UltraScale+ FPGA of Xilinx.
⇒To our knowledge, our work significantly outperforms all other existing solutions in terms of resource occupation, operating frequency, lookup delay and false positive rate compared to previously reported Bloom filter architectures and Contentaddressable Memories on FPGA.

Acknowledgement

This work is supported by the ESCALATE project, funded by FWO and SNSF (G0E0719N).

Novel Bloom filter algorithms and architectures for ultra-high-speed network security applications

Arish Sateesan, Jo Vliegen, Joan Daemen, Nele Mentens

Appendix

Evaluation: Figures and Tables

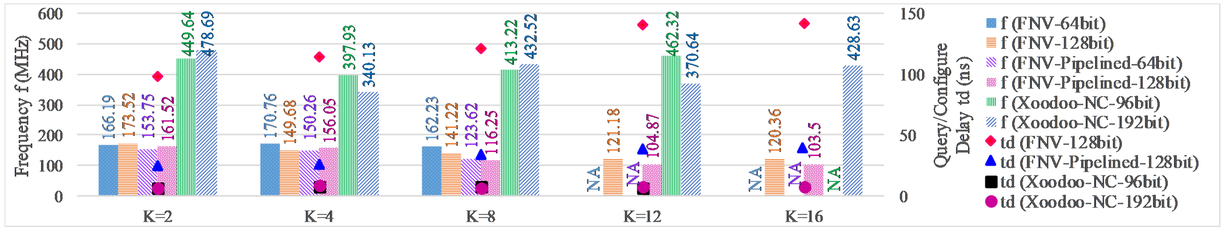

Figure 1: Maximum Operating Frequency and Delay of Bloom-1

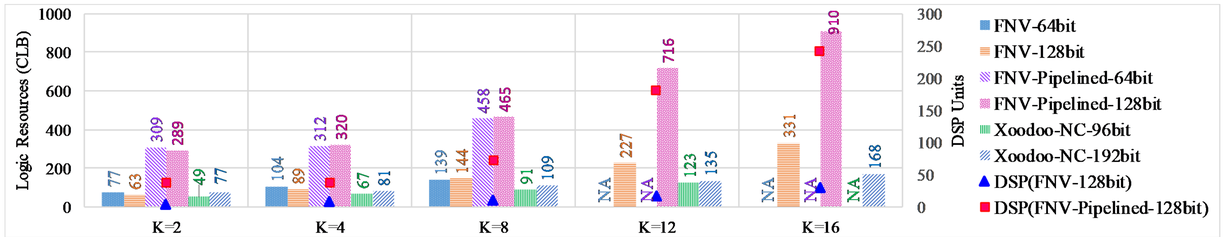

Figure 2: Resource Utilization of Bloom-1: Logic resources and DSP units

Design	k	fpr	LUTs	FFs	BRAM	LUTRAM	DSP	Max freq [MHz]	No. of cycles for lookup	Lookup Delay [ns]
Xoodoo-NC	**12**	**$2.61\text{x}10^{-7}$**	**675**	**158**	**7.5**	**0**	**0**	**462.32**	**3**	**6.49**
FNV-1a	12	$2.61\text{x}10^{-7}$	1366	339	7.5	0	15	121.18	16	132.03
FNV-1a Pipelined	12	$2.61\text{x}10^{-7}$	2706	1562	7.5	6	180	104.87	4	38.14
Direct CAM		0.0	6731	1145	320	1536	0	112.75	2	17.74
Custom Modified CAM		0.0	20'797	988	384	1536	0	225.07	3	13.33

Figure 3: COMPARISON OF BLOOM-1 BASED AND CAM BASED IMPLEMENTATIONS

Design	m	k	input size	LUTs	FFs	BRAMs	Frequency	fpr	No. of memory accesses	FPGA
[1](SBF)	4096	10	10 byte	1495	1297	5	73.51 MHz	$8.89\text{x}10^{-5}$	2	Virtex 2000E
[2](SBF)	16384	8	8 byte	1058	1058	4	200.6 MHz	$5.74\text{x}10^{-4}$	4	Virtex-4
[2](CBF)	16384	8	8 byte	1188	1188	4	201.6 MHz	$5.74\text{x}10^{-4}$	4	Virtex-4
Ours (xoodoo-NC)	**262144**	**12**	**12 byte**	**675**	**158**	**7.5**	**462.32 MHz**	**$2.61\text{x}10^{-7}$**	**1**	**Virtex UltraScale+**

Figure 4: COMPARISON WITH RELATED WORK

References

[1] J. Harwayne-Gidansky, D. Stefan, and I. Dalal. FPGA-based SoC for real-time network intrusion detection using counting Bloom filters. *IEEE Southeastcon 2009, Atlanta, GA*, pages 452–458, 2009.

[2] S. Dharmapurikar, P. Krishnamurthy, T. S. Sproull, and J. W. Lockwood. Deep packet inspection using parallel Bloom filters. *IEEE Micro, vol. 24, no. 1*, pages 52–61, 2004.

Appendix

Hardware block diagrams

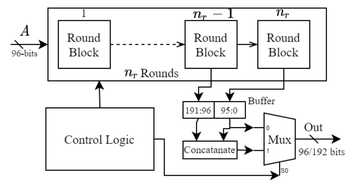

Figure 5: Xoodoo hardware block diagram

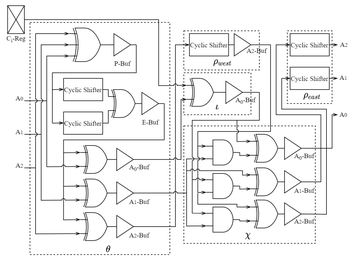

Figure 6: Xoodoo Round architecture

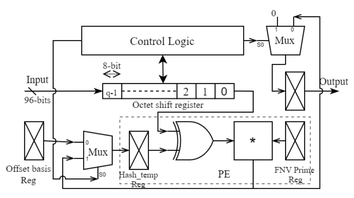

Figure 7: FNV-1a architecture

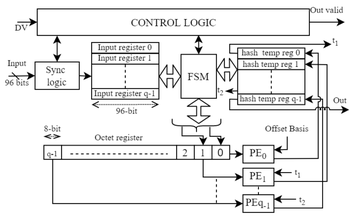

Figure 8: FNV-1a pipelined architecture

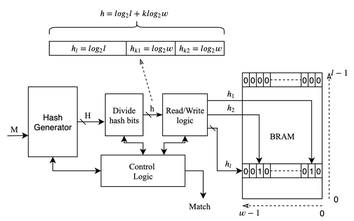

Figure 9: Bloom-1 architecture