Online Appendix to:
Area-Efficient Near-Associative Memories on FPGAs

UDIT DHAWAN and ANDRÉ DEHON, University of Pennsylvania

A. DISTRIBUTION OF DEGREES

The problem of expressing the probability distribution of the degrees of the G slots can be modeled as the Birthday Problem.[3] In a generalized Birthday Problem, one needs to find the probability that in a group of N people, n have the same birthday (any particular date). Although an exact answer can be found using the binomial distribution, this probability can be nicely approximated by the Poisson distribution as follows:

$$P(n) = e^{-\lambda}\frac{\lambda^n}{n!}, \tag{19}$$

where λ is the expected value of the number of birthdays on a single day ($\lambda = \frac{N}{365}$).

In the case of dMHC, the problem is that given cM slots in a G table (equivalent to the number of days in a year), what is the probability that exactly d key–value pairs will use a particular G slot (defined earlier as its degree) when M key–value pairs (equivalent to the number of people) are being stored in the dMHC? Therefore, in this case,

$$\lambda = \frac{M}{cM} = \frac{1}{c}. \tag{20}$$

Now, the probability distribution for the degrees of the G slots can be expressed using the Poisson distribution as follows:

$$P(degree = d) = P(d) = e^{-\lambda}\frac{\lambda^d}{d!}. \tag{21}$$

Since the expected value of the Poisson distribution is λ, the expected value of the degree of any G slot is $\lambda = \frac{1}{c}$, as expected.

While inserting a new key–value pair in a dMHC(k,c), we preferably look for a degree-0 G slot so that we do not victimize any of the existing key–value pairs. The probability that we can find at least one degree-0 G slot (out of k slots) can be expressed as follows:

$$\begin{aligned} P(\text{at least 1 degree-0}) &= 1 - P(\text{all non-zero degrees}) \\ &= 1 - (1 - P(0))^k \\ &= 1 - (1 - e^{-\lambda})^k. \end{aligned}$$

For $k = 4, c = 2$, this is 0.974 (probability that we do not victimize any G slot while inserting a new key–value pair). If we vary the sparsity factor and the number of hash functions, we can plot the probability that we do not victimize any G slot for different cases as shown in Figure 13.

[3]McKinney, E. H. (1966). Generalized Birthday Problem. *American Mathematical Monthly*, 73(4): 385–387.

 1936-7406/2015/01-ART3 $15.00
DOI: http://dx.doi.org/10.1145/2629471

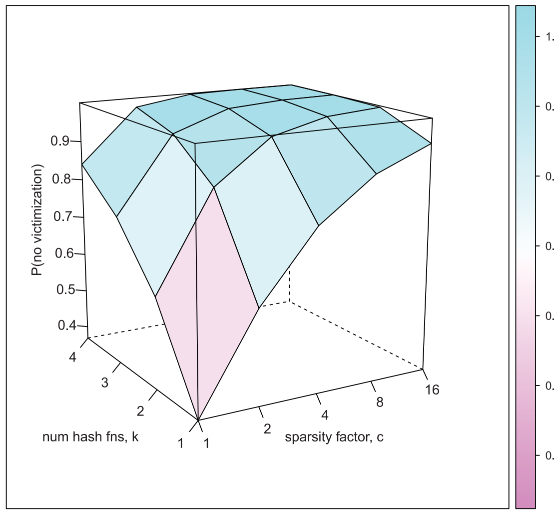

Fig. 13. Probability of no victimization.

We mentioned in Section 4 that we only use 2 bits for storing the degree of a G slot. We can compute the probability that a G slot has a degree 2 or more as follows:

$$P_{degree-0-or-1} = P(degree = 0) + P(degree = 1) = \left(1 + \frac{1}{c}\right) e^{-\frac{1}{c}}, \tag{22}$$

from which we conclude the following:

$$P_{not-degree-0-or-1} = 1 - \left(1 + \frac{1}{c}\right) e^{-\frac{1}{c}} = \frac{1}{2c^2} - \frac{1}{3c^3} + \frac{1}{8c^4} \cdots < \frac{1}{2c^2}. \tag{23}$$

Then the probability that all of the k G tables have this slot with a degree 2 or larger is

$$P_{all-2-or-more} < \left(\frac{1}{2c^2}\right)^k. \tag{24}$$

B. DEGREE OF VICTIMS

We can extend the Poisson distribution to also express the distribution of the degrees of the victimized G slots, in case we do not find a degree-0 G slot. For this, we can find the probability that the degree of a G slot is d given that it is being used to store at least one key–value pair. That is,

$$\begin{aligned} P_{victim}(degree = d) &= P(d|d \geq 1) \\ &= \frac{P(d)}{(1 - P(0))} \\ &= \frac{e^{-\lambda}}{1 - e^{-\lambda}} \frac{\lambda^d}{d!}. \end{aligned}$$

This is shown in Figure 14. As can be seen in the figure, with a high probability, we victimize only one key–value pair. We can now find the expected value of the degree of a G slot upon victimization, which comes out to 1.27.

Also, as mentioned earlier, we only keep 2 bits for maintaining the degree of G slots. That is, we saturate the degree at 3. This works well if the probability that the degree

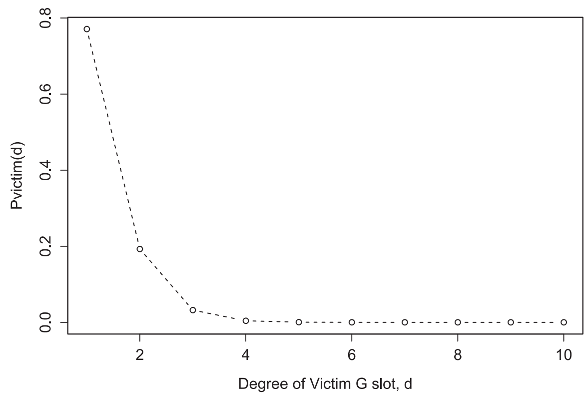

Fig. 14. PDF for degrees of victims, $c = 2$.

Table IV. 6-LUTs Consumed by dMHC(k,c)

dMHC Variant	6-LUTs for XOR-Reduce
Flat	$(w_k + w_v + 2\log_2(cM) + 2) \times \left(\lfloor \frac{k}{6} \rfloor + 1\right)$
2-level	$(2\log_2(cM) + 2) \times \left(\lfloor \frac{k}{6} \rfloor + 1\right)$
Fast-match	$(w_k + 2\log_2(cM) + 2) \times \left(\lfloor \frac{k}{6} \rfloor + 1\right)$
Fast-value	$(w_v + 2\log_2(cM) + 2) \times \left(\lfloor \frac{k}{6} \rfloor + 1\right)$

of a G slot is greater than 3 is very low. For $k = 4, c = 2$:

$$P(d > 3 | d \geq 1) = \sum_{d \geq 4} P_{victim}(d) = 0.0044. \tag{25}$$

C. LUT CONSUMPTION FOR DMHC (K,C)

In the main text, we focused on reducing the area in terms of BRAMs in the design. In addition to BRAMs, logic resources are consumed for implementing the hash functions and the match logic in the dMHC. Since the hash values are computed on the input key, it is the same for all dMHC variants. For an M-entry dMHC(k,c) storing w_k-bit keys and w_v-bit values, the 6-LUTs required for computing the hash values can be expressed as

$$LUT_{hash} = k \times \log_2(cM) \times \left\lceil \left(\left\lceil \frac{w_k}{\log_2(cM)} \right\rceil + 1 \right) \Big/ 6 \right\rceil. \tag{26}$$

After fetching the G table entries, the next step is to combine the k pieces together. Since the G table entry is unique to the dMHC variant, we show the 6-LUTs required for each variant in Table IV.

Finally, LUTs are consumed in the logic to match the stored key against the input key as shown here:

$$LUT_{match} = \left\lceil \frac{w_k}{3} \right\rceil + \left\lceil \frac{\lceil \frac{w_k}{3} \rceil}{5} \right\rceil. \tag{27}$$