An Exact Method to Enumerate Decomposition Charts for Index Generation Functions

Jon T. Butler
Department of Electrical and Computer Engineering
Naval Postgraduate School
Monterey, CA 93943–5121 U.S.A.
Email: jon_butler@msn.com

Tsutomu Sasao
Department of Computer Science and Electronics
Meiji University
Kawasaki-City, Kanagawa 214-8571 JAPAN
Email: sasao@cs.meiji.ac.jp

***Abstract*—In a previous paper, the balls-in-bins model was shown to efficiently enumerate random functions as a means to estimate the size of programmable architecture for the circuit needed to realize an index generation function. Because there are so many balls-in-bins instances, it is typically not possible to enumerate all. So, a Monte Carlo simulation is performed instead. In this paper, we show how to improve the balls-in-bins estimates by using a weighted approach. Additionally, we solve the the following - Problem: Derive an exact enumeration of all decomposition charts for the same analysis as in the previous paper. Our solution is based on the enumeration of integer partitions.**

***Index Terms*—index generation functions, decomposition chart, logic design, exact enumeration, partitions of integers.**

I. Introduction

Our goal is to understand the extent to which minimization techniques are effective in producing minimal and near-minimal circuits to realize index generation functions by the function decomposition technique [5]. Index generation functions are useful in access control lists, virus detection, packet classification, and in general pattern matching applications. While software can be used in these applications, hardware implementations of index generation functions are better, because they can perform at much higher speeds. Because memory is cheap, it can store much data - on the order of millions of words. A naïve use of typically many comparators could implement these applications. However, a memory-based system has been proposed by Sasao [4] that uses few comparators and a minimum amount of memory. It is fast and consumes relatively little power. Since it is memory-based, it can be easily modified to adapt the stored data to newly acquired data.

II. Index Generation Functions

Table I shows an example index generation function. Here, there are four variables, x_1, x_2, x_3, and x_4 and four **indices**, 1, 2, 3, and 4. If this represents a virus detection system, the four assignments of values to x_1, x_2, x_3, and x_4 represent potential viruses, and their corresponding indices represent addresses at which those viruses are processed. The assignments shown are called **registered vectors**. In general, there are k indices, and these have values $\{1, 2, \ldots, k\}$, one for each assignment. All assignments that do not correspond to an index map to 0 and are omitted from Table I. The assignments of values to x_1, x_2, x_3, and x_4, for these cases, are considered *not* to be viruses.

TABLE I: Example of an index generation function

x_1	x_2	x_3	x_4	f
0	0	0	1	4
0	1	1	1	2
1	0	1	0	3
1	1	0	0	1

III. Decomposition

Functional decomposition [1], [2] is a process in which a given function is divided into two (or more) subfunctions. The goal is to realize the subfunctions as physical circuits; for example, as LUTs in an FPGA. Fig. 1 shows the functional decomposition of $f(X)$, into two subfunctions $h(X_1)$ and $g(h, X_2)$, where $X = X_1 \bigcup X_2$. f, g, and h can be modeled as multiple-valued functions, and can be represented in the circuit by multiple binary-valued lines.

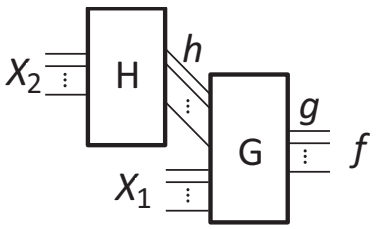


Fig. 1: Decomposition of a logic function.

Since we can choose h as the identity function (where the output of H is the same as its input), all functions have such a decomposition. However, this decomposition is trivial. An important tool in the design of circuits with *nontrivial* decompositions is the decomposition chart.

Definition 3.1. *Let $f(X)$ be a function, and let (X_1, X_2) be a partition of the variables X. Let $X_1 = (x_1, x_2, \ldots, x_{n_1})$ and $X_2 = (x_{n_1+1}, x_{n_1+2}, \ldots, x_n)$. A* **decomposition chart** *for f is an array with 2^{n_1} columns and 2^{n_2} rows, where columns*

2378-2226/18/$31.00 ©2018 IEEE
DOI 10.1109/ISMVL.2018.00032

are labeled by the 2^{n_1} assignments of values to X_1 and rows are labeled by the 2^{n_2} assignments of values to X_2. Each entry is the value of the index generation function, 1, 2, . . . , k, *or* 0. *Here,* 0 *is a special value that represents an assignment that is* not *stored. X_1 is the set of* **column variables**, *while X_2 is the set of the* **row variables**. *The number of column variables is $n_1 = |X_1|$, and the number of row variables is $n_2 = |X_2|$, for $n_1+n_2 = n$, the total number of variables. The* **column multiplicity** μ *is the number of distinct columns; it can range from 1 to 2^{n_1}. The number r of* **rails** *is the number of binary wires between circuits H and G needed to encode the μ columns. Specifically, $r = \lceil \log_2 \mu \rceil$.*

IV. The Balls-in-Bins Model

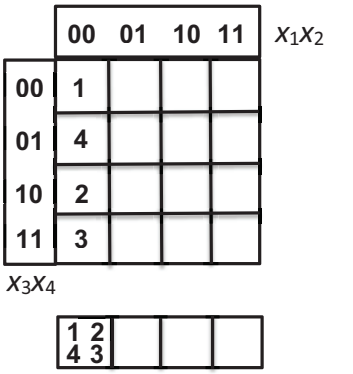
In [6], the "balls-in-bins" model was proposed as an approximation for the distribution of column multiplicities to decomposition charts of index generation functions. In this model, the bins are the columns (of which, there are 2^{n_1}), and the balls are the indices (of which, there are k). An element of this distribution of balls-in-bins is a choice with repetition of a column number for all of the balls. If two balls are assigned the same column number b, then those two balls are viewed as occupying the same bin b. For each distribution, we compute the number of bins with at least one ball. We seek the plot of the number of distributions of balls-in-bins with some specified number of columns with at least one nonzero value, which is a measure of μ, the column multiplicity. Although we prefer to enumerate *all* distributions of balls-in-bins, there are too many for the values of k and n_1 in which we are interested. Thus, we apply a Monte Carlo simulation, in which we choose randomly each distribution of balls-in-bins.

We note that this is an *approximation* to the distribution across decomposition charts. This is because two distributions of balls-in-bins do not, in general, represent the same number of distributions to decomposition charts. Fig. 2 shows an example of two balls-in-bins distributions and, for each, one of their corresponding decomposition charts. Here, $n_1 = 2$ and $k = 4$.

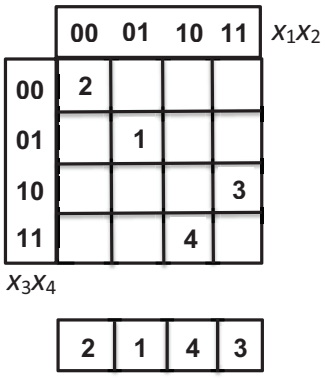
In Fig. 2a), all balls fall into the same bin (leftmost bin), and $\mu = 2$. This distribution corresponds to $4! = 24$ decomposition charts, one of which is shown. In Fig. 2b), all balls fall into different bins, and $\mu = 4$. This corresponds to $4^4 = 256$ decomposition charts, one of which is shown. When μ is smaller, balls tend to fall into the same bins, and there tend to be fewer distributions. Thus, in the balls-in-bins model, distributions with smaller μ are *overemphasized* compared to distributions with larger μ.

V. A Weighted Balls-in-Bins Model

However, we can correct this by *weighting* each balls-in-bins distribution with a value that is the number of decomposition charts to which it corresponds. In the example, the balls-in-bins distribution shown in Fig. 2a) would be weighted with 24, and the distribution shown in Fig. 2b) would be weighted with 256. By tallying the weights as each balls-in-bins distribution is generated, we compute a total weight, which can be



(a) Low Multiplicity Function



(b) High Multiplicity Function

Fig. 2: Comparing two distributions of balls-in-bins with their decomposition chart counterparts. Blank entries are 0.

used to normalize the final values. The normalization occurs as follows. Let the distribution of balls-in-bins be specified as $(b_0, b_1, \ldots, b_{n-1})$, where b_i is the number of balls that fall into the i-th bin. For example, the distributions in Fig. 2a) and 2b) correspond to $(4, 0, 0, 0)$ and $(1, 1, 1, 1)$, respectively. Because there are k balls,

$$k = \sum_{i=0}^{n-1} b_i.$$

In general, the weight W associated with balls-in-bins instance $(b_0, b_1, \ldots, b_{n-1})$ is

$$W = \prod_{i=0}^{n-1} \binom{2^{n_2}}{b_i} b_i! \ ,$$

where n_2 is the number of row variables (and 2^{n_2} is the number of rows). That is, W is the number of decomposition charts that corresponds to balls-in-bins distribution $(b_0, b_1, \ldots, b_{n-1})$. Table 4.2 of [6] shows a distribution of instances of balls-in-bins to column multiplicity μ for the

TABLE II: Comparing the accuracy of the balls-in-bins model with the weighted balls-in-bins model and the exact model ($n_1 = 8$, $k = 20$)

Rails r	Col. Mult. μ	Balls-in-Bins [6]	Wgted Balls -in-Bins	Exact # of Decomp. Ch.
4	12+1	0	0.6	0.0
4	13+1	1	0.0	0.4
4	14+1	16	7.7	11.1
4	15+1	265	194.0	208.0
5	16+1	3231	2660.7	2657.7
5	17+1	25670	22543.1	22567.7
5	18+1	130523	120832.2	121049.3
5	19+1	374304	369297.2	369000.7
5	20+1	465990	484464.4	484505.1
Total		1000000	999999.9	1000000.0

case of $k = 20$ and $n_1 = 8$ (8-bit numbers or 256 bins). For the weighted balls-in-bins model, we must specify how many rows are in the decomposition chart. We chose it to be the same as k or 20. In this case, the maximum μ is 21 (20 columns with nonzero indices plus 1 for the column with all 0's). Table II shows this data. That is, it shows, in the third column, data from a Monte Carlo simulation identical to the one used in Table 4.2 of [6]. The fourth column shows the distribution resulting from applying the weighted method described above. Each entry in this column should be viewed as an exact percentage of the total. For example, the entry 465,990, corresponds to exactly 46.599% of the *decomposition charts*, there being a total of 1,000,000 samples. In comparing the two columns, one can see that the weighted balls-in-bins method has similar values compared to the balls-in-bins model, but there is a noticeable difference between the two. This is the basis for our statement that the balls-in-bins method gives an *approximation* to the distribution of functions to column multiplicities in the decomposition chart approach to the design of index generation functions.

This data shows that there is an overemphasis of decomposition charts by the balls-in-bins model with smaller μ. Indeed, the balls-in-bins model function counts (third or middle column) are all larger than the weighted balls-in-bins function counts (fourth column) for all μ values except for the smallest and the largest μ value (12+1, 20+1). Both columns are derived from a Monte Carlo simulation, and, thus are approximate. However, we were able to derive exact function counts for various μ values, and these are shown in the rightmost column. The exact function counts show that the weighted balls-in-bins model is a closer approximation to exact function counts (rightmost column) than the original balls-in-bins model. We will explain the exact method (rightmost column) in the next section.

VI. An Exact Integer Partition Model

In this section, we propose a model based on integer partitions as a way to tractably compute *exact* values for the distribution of decomposition charts to the number of columns with at least one nonzero value. Because there are many instances of the balls-in-bins problem, we are not able to enumerate them all. This is why we did a Monte Carlo simulation. However, we can compute the number of decomposition charts corresponding to a given integer partition in a way similar to that shown in the previous section. This allows an exhaustive enumeration that produces an exact value, unlike the approximation derived from a Monte Carlo simulation of the balls-in-bins model or the weighted balls-in-bins model.

An approximation to the number of partitions [3] $p(n)$ is

$$p(n) \sim \frac{1}{4n\sqrt{3}} e^{\pi\sqrt{\frac{2n}{3}}}. \quad (1)$$

The exact number is 1, 2, 3, 5, 7, 11, 15, 22, 30, 42, 56, 77, 101, 135, 176, 231, 297, 385, 490, 627, 792, 1002, 1255, 1575, 1958, 2436, 3010, 3718, 4565, 5604 for $n = 1, 2, 3, ..., 30$. This relatively gradual increase suggests that an exhaustive enumeration is possible for many sizes of decomposition charts using the integer partition model. Indeed, there are ways to generate integer partitions randomly with uniform probability [3]. In this way, for really large values of n in (1), such that exhaustive enumeration is not possible, one can resort to Monte Carlo simulations.

To illustrate how to use integer partitions to compute the distribution of decomposition charts, consider, for example, the partition $4 + 2 + 2 + 1 = 9$ across decomposition charts with 12 columns and 20 rows. Here, 4 represents the occurrence of one column with four nonzero values, $2 + 2$ represents the occurrence of two columns each with two nonzero values, and 1 represents one column with one nonzero value. Since $4 + 2 + 2 + 1 = 9$, $k = 9$.

There are 12 ways to choose the column corresponding to the 4 (i.e. 4 nonzero values occur in one of 12 columns in 12 ways). There are $\binom{20}{4}$ ways to choose the 4 rows in which the nonzero values occur. Then, there are $9 \times 8 \times 7 \times 6$ ways in which 4 nonzero values are chosen. Thus, for the 4 part, there are $12 \times \binom{20}{4} \times 9 \times 8 \times 7 \times 6$ choices.

Now, consider the two 2 parts. There are $\binom{11}{2}$ ways to choose the columns in which the two 2's occur (i.e. after processing the 4 part, there are 11 columns in which to place the two 2 parts). For both columns, there are $\binom{20}{2}$ ways to choose where the nonzero values occur. Then, there are $5 \times 4 \times 3 \times 2$ ways to choose the nonzero values over the pair of 2's. Thus, for the pair of 2's, the number of choices is $\binom{11}{2} \times \binom{20}{2}^2 \times 5 \times 4 \times 3 \times 2$.

Finally, the 1 part of the partition can be placed in one of $\binom{9}{1}$ columns, and then in one of $\binom{20}{1}$ rows. The total number of ways to choose for the 1 part is $\binom{9}{1} \times \binom{20}{1}$ ways. Thus, the total number of decomposition charts associated with the partition $4 + 2 + 2 + 1$ is

$$\left(12 \times \binom{20}{4} \times 9 \times 8 \times 7 \times 6\right) \times \left(\binom{11}{2} \times \binom{20}{2}^2 \times 5 \times 4 \times 3 \times 2\right) \times \left(\binom{9}{1} \times \binom{20}{1}\right).$$

This equation can be rewritten as

$$\binom{12}{1}\binom{11}{2}\binom{9}{1}\binom{20}{4}\binom{20}{2}^2\binom{20}{1}9! = 7,540,158,181,248,000,000\ . \quad (2)$$

Note that this partition and all of its associated decomposition charts have $\mu = 5$ (there are 4 parts and at least one all-0 column). In the case of $n = 30$, there are only 5,604 partitions. We could easily enumerate such a number. However, the sum will be very large, as suggested by the very large number of decomposition charts indicated at (2).

The fifth or rightmost column in Table II shows an exact enumeration based on the integer partition method just described[1]. This data was calculated by a MATLAB program running on a 64-bit Intel Core®i5-4200U CPU with a 1.6GHz clock and 8 GB of RAM. In the exact method, we must choose the number of rows in the decomposition chart. We chose 20. The fourth column shows the values obtained by a Monte Carlo method using 1,000,000 samples, which were weighted values from the balls-in-bins method. These are close to the values produced by the exact integer partition method. The third column shows the values obtained by the balls-in-bins model using Monte Carlo simulation with 1,000,000 samples [6]. These values were noticeably larger for smaller values of μ. This coincides with our observation earlier that the balls-in-bins model overemphasizes the number of functions with smaller values of μ. It should be noted that the exact enumeration values were exact only within the precision achieved in our MATLAB program. We received warning messages stating that computation of certain combinatorial functions $\binom{n}{r}$ may not be accurate. Since the messages were warning messages (and not error messages), and the results were consistent, we ignored them.

In order to assess this last issue of inaccuracy, we repeated the experiments in Table II for the case of $n_1 = 6$ (reducing the number of columns from 256 down to 64). We chose the number of rows to be 20, as in Table II. For this case, the exact computation of the number of decomposition charts by MATLAB yielded no warning messages of inaccurate computations. We repeated the Monte Carlo simulation computations, also. The results are shown in Table III. As in the case of Table II, the decomposition charts in the case of the exact solution was chosen to be 20.

[1]To make the comparison easier, we have prorated the exact values to coincide with a distribution over 1000000 samples; e.g. 50% is 500000.

TABLE III: Comparing the accuracy of the balls-in-bins model with the weighted balls-in-bins model and the exact model ($n_1 = 6$, $k = 20$)

Rails r	Col. Mult. μ	Balls-in -Bins	Wgted Balls -in-Bins	Exact # Decomp. Ch.
4	9+1	0	0.0	0.0
4	10+1	4	0.5	1.3
4	11+1	35	19.6	25.2
4	12+1	503	297.1	322.0
4	13+1	3709	2738.8	2740.0
4	14+1	19759	15806.2	15720.8
4	15+1	71001	60862.8	60973.4
5	16+1	171579	158742.2	158435.3
5	17+1	273449	269983.3	269064.7
5	18+1	272387	283483.5	283813.1
5	19+1	151509	166728.2	167216.2
5	20+1	36065	41337.8	41688.2
Total		1000000	1000000.0	1000000.2

Table III shows the same overemphasis on the number of decomposition charts with small μ that occurred in the balls-in-bins model (middle column) as shown by Table II. Further, the approximate number of decomposition charts as computed by the Monte Carlo method over 1,000,000 samples closely matches the exact number computed by the methods described above (rightmost column). Another interesting observation is that reducing the number of columns (from 256 to 64) skews the distribution of decomposition charts so that there are fewer decomposition charts with larger μ. This is expected since, in the case of fewer columns, there is a greater tendency for columns to have more nonzero values than when there are many columns. Thus, we expect fewer decomposition charts with higher μ values; i.e. the distributions are "skewed" downward, so that there are fewer decomposition charts with larger μ values.

Table IV shows the exact values for the distributions of decomposition charts for $n_1 = 5$, 6, 7, 8, and 9 (number of columns is 32, 64, 128, 256, and 512, respectively), where the number of rows is 256. This shows that, as the number of columns increases, the fraction of decomposition charts shifts toward larger μ. This shows that, for fixed k, when the number of columns increases, the number of *distinct* columns increases, as shown by the increase in μ. This is an expected result, since, with more columns, the index values spread out across the columns, resulting in more distinct columns with larger k.

Table V shows the exact values for the distribution of decomposition charts with 32 columns, 8 rows, and $k = 8$, 16, 24, and 32 indices together with the values computed using the balls-in-bins model. Each entry in this table has the form A/B, where A is the value obtained from the balls-in-bins model, and B is the value obtained from the integer partition model, which yields exact values. As with previous analyses, the balls-in-bins model overemphasizes lower values of μ. That is, for each of the four values of k, the percentage

TABLE V: Distribution of decomposition charts with 32 columns versus μ using the exact model

Rails r	μ Col. Mult.	Number of Indices = k = Number of Columns 8	16	24	32
1	1+1	0.0/0.00	0.00	0.00	0.00
2	2+1	0.0/0.00	0.00	0.00	0.00
2	3+1	0.0/0.00	0.00	0.00	0.00
3	4+1	0.0/0.07	0.0/0.00	0.0/0.00	0.0/0.00
3	5+1	2.3/1.55	0.0/0.00	0.0/0.00	0.0/0.00
3	6+1	15.8/13.09	0.0/0.00	0.0/0.00	0.0/0.00
3	7+1	43.2/42.21	0.0/0.00	0.0/0.00	0.0/0.00
4	8+1	38.6/43.08	0.1/0.04	0.0/0.00	0.0/0.00
4	9+1	0.0/0.00	0.7/0.46	0.0/0.00	0.0/0.00
4	10+1	0.0/0.00	3.8/2.85	0.0/0.00	0.0/0.00
4	11+1	0.0/0.00	12.3/10.48	0.0/0.01	0.0/0.00
4	12+1	0.0/0.00	24.7/23.14	0.2/0.13	0.0/0.00
4	13+1	0.0/0.00	29.6/30.33	1.1/0.81	0.0/0.00
4	14+1	0.0/0.00	20.4/22.65	4.2/3.33	0.0/0.02
4	15+1	0.0/0.00	7.3/8.72	10.9/9.39	0.2/0.14
5	16+1	0.0/0.00	1.0/1.32	20.0/18.19	1.0/0.76
5	17+1	0.0/0.00	0.0/0.00	24.4/24.32	3.6/2.83
5	18+1	0.0/0.00	0.0/0.00	21.0/22.33	8.9/7.63
5	19+1	0.0/0.00	0.0/0.00	12.3/13.92	16.3/14.93
5	20+1	0.0/0.00	0.0/0.00	4.8/5.76	21.8/21.29
5	21+1	0.0/0.00	0.0/0.00	1.2/1.53	21.3/22.10
5	22+1	0.0/0.00	0.0/0.00	0.2/0.24	15.3/16.65
5	23+1	0.0/0.00	0.0/0.00	0.0/0.02	7.8/9.03
5	24+1	0.0/0.00	0.0/0.00	0.0/0.00	2.9/3.48
5	25+1	0.0/0.00	0.0/0.00	0.0/0.00	0.7/0.94
5	26+1	0.0/0.00	0.0/0.00	0.0/0.00	0.1/0.17
5	27+1	0.0/0.00	0.0/0.00	0.0/0.00	0.0/0.02
5	28+1	0.0/0.00	0.0/0.00	0.0/0.00	0.0/0.00
5	29+1	0.0/0.00	0.0/0.00	0.0/0.00	0.0/0.00
5	30+1	0.0/0.00	0.0/0.00	0.0/0.00	0.0/0.00
5	31+1	0.0/0.00	0.0/0.00	0.0/0.00	0.0/0.00
Total %		100.0/100.00	100.0/100.00	100.0/100.00	100.0/100.00
No. of Partitions		22	231	1575	8349

TABLE IV: Distribution of decomposition charts with 256 rows and $k = 20$ versus μ using the exact model.

Rails r	μ Col. Mult.	Number of Columns 2^{n_1} 32	64	128	256	512
3	7+1	0.0	0.0	0.0	0.0	0.0
4	8+1	0.1	0.1	0.0	0.0	0.0
4	9+1	4.8	0.0	0.0	0.0	0.0
4	10+1	92.0	0.1	0.0	0.0	0.0
4	11+1	1053.3	2.3	0.0	0.0	0.0
4	12+1	7513.6	40.8	3.2	0.0	0.0
4	13+1	34398.1	471.6	56.1	0.6	0.0
4	14+1	102960.6	3670.5	670.0	15.2	0.3
4	15+1	203207.2	19445.0	5513.9	265.5	10.2
5	16+1	264057.9	70188.5	30952.7	3178.1	252.6
5	17+1	223003.0	170853.7	115540.7	25421.1	4175.4
5	18+1	118918.5	273328.2	272424.3	129056.4	43902.6
5	19+1	37890.4	272879.1	364760.9	373874.8	263989.5
5	20+1	6456.9	152791.0	210078.2	468188.5	687669.5

of decomposition charts realizing the μ values shown is artificially high compared to the exact integer partition values. As before, MATLAB was used to compute the number of decomposition charts. In this case, we show the percentage of the total number of decomposition charts for each value of μ, the column multiplicity. In the case of Table V, all four values (k = 8, 16, 24, and 32) produced *no* accuracy warnings. It is interesting that for $k = 32$, 24, and 16, the vast majority of decomposition charts fall squarely within the r values of 5, 4, and 3, respectively, where r is the number of rails. This means that, for most decomposition charts in these ranges, there is little benefit to seeking one that will reduce the number of rails. In the case of $k = 8$, the situation is different. In this case, 43% of the decomposition charts have $r = 4$, while 57% have $r = 3$. Therefore, if one has a decomposition chart with $r = 4$, there is a good chance that one with $r = 3$ can be found. In this way, the number of rails is reduced by 1, as is the circuit complexity.

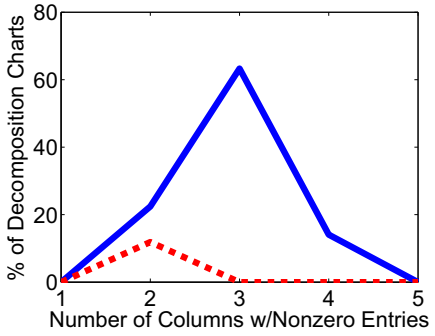


Fig. 3: 4×4 Decomposition Charts With $k = 4$

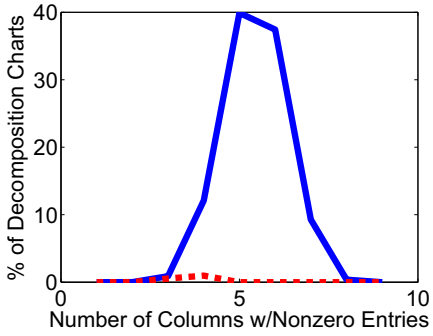


Fig. 4: 8×8 Decomposition Charts With $k = 8$

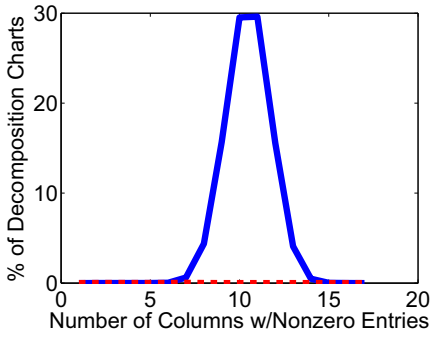


Fig. 5: 16×16 Decomposition Charts With $k = 16$

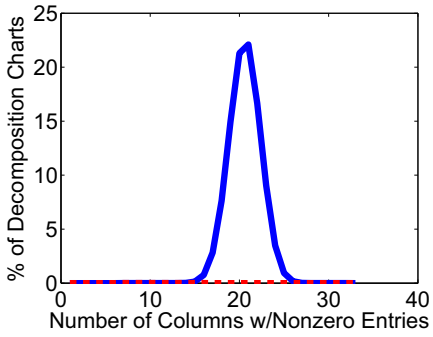


Fig. 6: 32×32 Decomposition Charts With $k = 32$

VII. Integer Partition Model Results

Fig. 3 shows data from the exact integer partition model for the case where there are four rows, four columns, and $k = 4$. The solid blue line corresponds to all partitions. The dashed red line corresponds to partitions in which the smallest part is 2. This line is well below the solid blue line, showing there are many fewer decomposition charts in this case. That is, in this case, all decomposition charts have the property that there are at least two indices in every column containing an index. Ideally, there should be one or no index in each column, as we show in Table I. Note also that the dashed red line showing decomposition charts in which the smallest part is 2 tend to contain fewer columns with indices. This is an expected result, since, with at least two indices per column, there will indeed be fewer columns.

Fig 4 shows the data from the integer partition model for the case where there are eight rows, eight columns, and $k = 8$. In this case, there are many fewer decomposition charts with two or more indices in columns that have indices (dashed red line) compared to the case of all decomposition charts. Figs. 5 and 6 show the cases of 16 rows, 16 columns, $k = 16$ and 32 rows, 32 columns, $k = 32$, respectively. These curves suggest a peak that occurs at a point that is doubled for each doubling of the values of n_1, n_2, and k.

VIII. Concluding Remarks

Our goal is to estimate the size of the programmable architecture needed to realize index generation functions, which are useful in virus detection and routing. In [6], the balls-in-bins model was used to efficiently estimate the size. In this paper, we use the integer partition model instead. It has two advantages. First, it is an exact enumeration, and second, it is efficient. As a result, for functions of small size, exhaustive enumeration is possible, versus Monte Carlo simulation, as in the balls-in-bins model. The integer partition model yields exact sizes, especially for smaller functions, versus an approximation, as in the balls-in-bins model.

Acknowledgment

This research is partly supported by the Japan Society for the Promotion of Science (JSPS) Grant in Aid for Scientific Research. Referees' comments helped to improve the paper.

References

[1] R. L. Ashenhurst, "The decomposition of switching functions," Inter. Symp. on the Theory of Switching, vol. 108, Issue 3, pp. 75–116, April 1957.

[2] H. A. Curtis, A New Approach to the Design of Switching Circuits, D. Van Nostrand Co. Princeton, NJ, 1962.

[3] E. M. Reingold, J. Nievergelt, and M. Deo, Combinatorial Algorithms: Theory and Practice, Prentice-Hall, Inc., Engelwood Cliffs, NJ, 07632, 1977, pp. 193–196.

[4] T. Sasao, Memory-Based Logic Synthesis, Springer, New York, Dordrecht, Heidelberg, London. 2011. pp. 92–118.

[5] T. Sasao, K. Matsuura, and Y. Iguchi, "A heuristic decomposition of index generation functions with many variables," The 12th Workshop on Synthesis And System Integration of Mixed Information Technologies (SASIMI-2004), pp. 23-28.

[6] T. Sasao and J. T. Butler, "Decomposition of index generation functions using a Monte Carlo method," Advanced Logic Synthesis, Springer, André Reis, ed., 2017.