A q-gram Birthmarking Approach to Predicting Reusable Hardware

Kevin Zeng and Peter Athanas
NSF Center for High-Performance Reconfigurable Computing (CHREC)
Bradley Department of Electrical and Computer Engineering
Virginia Tech, Blacksburg, USA
Email: {kaiwen and athanas}@vt.edu

***Abstract*—Designer productivity is a growing concern as overall hardware complexity rises. Design reuse, a key component in productivity, is underutilized. Not only can existing designs be reused, but also the patterns and information contained within them as well. With the increase in the number of circuits available, there requires a need to analyze and retrieve designs with ease in order to accelerate design entry. In this paper, a birthmarking approach using q-grams is presented. Using this technique, design patterns regarding existing circuits can be captured and used to not only suggest similar and reusable designs, but functional blocks throughout the design phase, with little to no effort from the user. Preliminary experiments and case studies of the q-gram birthmarking technique were performed on over 250 circuits from various sources in order to show the feasibility of the proposed methods.**

I. Introduction

Design reuse has been a prominent factor in bridging the productivity gap, reducing design time, and increasing overall productivity of designing hardware [1] [2]. However, the cost of design reuse depends on the time associate with reusing intellectual properties (IP). This includes the search for IP to the integration of the IP into the overall design. If a significant portion of the reduce project time is associated with the cost of reuse, further improvements can be made.

Traditionally, the reuse flow requires the circuit designer to explicitly know when and where to search for existing circuits. Considering that there are no universally accepted standards, IP repositories may be unorganized and scattered across various sources. Searches may result in no matches found even though the circuit exists under another ”keyword”. The designer is simply *unaware* of the best way to retrieve the desired circuit. Yet, the circuit that is being searched for may not in fact exist at all.

The tool proposed in this paper will continuously monitor the circuit during the entry process. When the reference circuit has been modified, the design is compared against an archive of existing designs. If there exists a proper community structure of easing contributions and automating access, the number of existing designs can rapidly rise to tens of thousands or more. Candidate circuits and reusable subcomponents are suggested to the user within the design environment in a ranked order. This entire flow is automated and requires no manual intervention from the user. The only time the user interfaces with the proposed tool is when a circuit of interest appears in the list of suggested designs.

This work was supported in part by the I/UCRC Program of the National Science Foundation within the NSF Center for High-Performance Reconfigurable Computing (CHREC), Grant No. IIP-1266245.

In this paper, a methodology for analyzing and comparing circuits using a q-gram *birthmarking* approach is proposed. A *birthmark* is a representation defined by the inherent characteristics of a circuit and is constructed such that the structural and functional information of the circuit is captured. The q-gram *birthmarking* approach can be extended and used for analyzing and retrieving relevant information to help accelerate the design entry process. An assessment is performed on the quality of the results when comparing a reference circuit against a compiled database of cores from various archives for circuit suggestion and functional block prediction.

II. Related Work

Reuse systems commonly use meta-data [3] and case-based reasoning [4] to manage existing designs. This involves manually searching for designs and manually building the database. Graph-based methods have been used to search for existing designs in a database [5] [6]. This approach is ideal since no manual preprocessing is needed; however, the method of comparison only captures structural similarity and not functional similarity.

There has been little work done in assessing the *similarity* between two circuits. Shi et al [7] finds similarities between pairs of nodes in the circuit. InVerS [8] uses fast simulations to find similarities, but is used primarily for incremental verification. Many of these endeavors do not give a good sense of how similar two circuits are overall.

Birthmarking approaches have been commonly used for assessing the similarity of software [9] [10]. Tamada et al [9] first introduced the idea of birthmarks in order to detect theft of Java programs. Many of these concepts can be leveraged and applied to hardware designs [11].

III. Hardware Birthmarks

A hardware *birthmark* is defined here as a set of features that describes the underlying structure and functionality inherent to a specific circuit. Let x and z be two circuits and let b be the

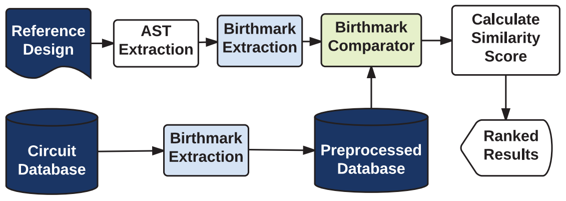

Fig. 1. Block diagram of overall system flow

function that extracts the birthmark from a given circuit. $b(x)$ is a birthmark of x if:

1) $b(x)$ is obtained from x itself
2) if x and z are copies of each other, then $b(x) = b(z)$

Digital circuits are commonly described in a hardware description language (HDL) such as Verilog. All programming languages have an inherent structure and can typically be represented as an abstract syntax tree (AST). Comparing the AST directly can be expensive for very large designs and unfruitful due to the diverse ways of describing a function. Therefore, the AST is broken down into smaller parts and modeled using a path-based q-gram approach [12]. Figure 1 shows the overall system flow for the proposed reuse model.

q-grams are commonly used to assess the similarity of two documents. The text of each document is decomposed into sequences or tokens of q length using a sliding window resulting in a model that represents the overall document. The q-gram model can also be used for text prediction by using a Markov model [13]. This idea can be leveraged to predict the next most likely operation to be used in the circuit.

A. Path-based q-grams

The idea of q-grams works well with sequential data. For non-sequential data such as graphs, q-gram can be extended by using a path-based q-grams approach. Let $G(V, E)$ be a graph that represents the circuit. A path-based q-gram model of G is a set of simple paths starting from $v \in V$ of length q, for all nodes in V.

Figure 2 shows three different path-based q-gram models for a basic 8-bit counter. The nodes of the AST correspond to primitive operations such as add, shift, multiply, logic operators, etc. Therefore, the elements of the q-grams are the operations themselves. Each operation is assigned a letter in the alphabet for simplicity.

1) Path-based q-gram Schemes: The bounds of the match can be adjusted based on the representation of the q-gram. Three different schemes for path-based q-grams were explored: list-based, set-based, and frequency-based. Examples

TABLE I
DETAILED DESCRIPTION OF THE DIFFERENT BIRTHMARK SCHEMES

Scheme	q-gram	Description
List	AXAM	q-gram extracted from circuit
Frequency	AAMX	Ordering of elements is insignificant (sorted)
Set	AMX	Both ordering and frequency of element is ignored. Only one A is taken into account

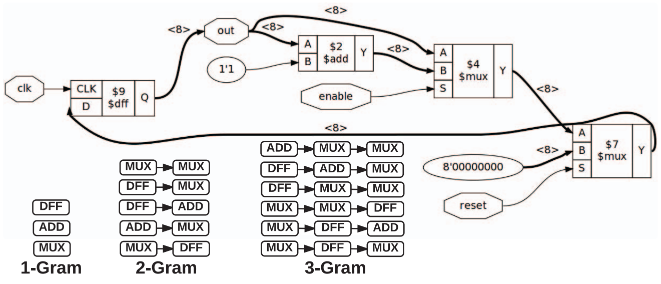

Fig. 2. Path-based 3-gram model of basic 8-bit counter. (q to 1)-gram is shown because it includes all the q-grams that are less than and equal to 3.

of each scheme are detailed in Table I. Figure 2 shows a strict list-based scheme where the order of the elements in the q-gram is preserved. Depending on the goal of the application, a different path-based q-gram scheme might be chosen. Based on the results of the matches, the frequency-based q-gram approach gave the best results for this specific application.

2) Extension to (q to 1)-gram: The notion of the q-gram model is extended further to include tokens with length less than q because paths of length q may not exist in certain circuits. In other words, the q-gram model is a combination of tokens where $1 \leq len(token) \leq q$. For example, the circuit in Figure 2 does not have a 5-gram since the longest path from any node in the circuit is 4. By taking the smaller q-gram models into account, this problem can be alleviated.

B. Computing Similarity Between Birthmarks

Jaccard's index is commonly used to compare the similarity of two sample sets [14]. Greater weight is placed on tokens that have a higher q, $1 \leq len(token) \leq q$. The modified Jaccard's index is described in Algorithm 1. The intuition is that the more tokens the circuits share, the more similar the two circuits are.

Algorithm 1 Calculating similarity between two birthmarks

```
1:  function BIRTHMARK_SIMILARITY($b(x), b(y)$)
2:      $intersection = 0$, $union = 0$
3:      for each $q$-gram $g \in b(x)$ do
4:          $union = union + len(g)$
5:          if $g \in b(y)$ then
6:              $intersection = intersection + len(g)$
7:          end if
8:      end for
9:      for each $q$-gram $g \in b(y)$ do
10:         $union = union + len(g)$
11:     end for
12:     $sim = intersection/(union - intersection)$
13: end function
```

C. Predicting Design Patterns

The q-gram model can be extended such that future elements in a token can be predicted by utilizing Markov models. The intuition behind using a Markov model is that future states can be predicted with certain probability based on the what the current states are. This is typically done by recording the frequency of each token observed and using a maximum likelihood estimator (MLE) to predict the next most likely function. More formally:

Definition 1. The MLE is defined to be

$$P(w_n|w_{n-q+1}^{n-1}) = \frac{C(w_{n-q+1}^{n-1}w_n)}{C(w_{n-q+1}^{n-1})} \tag{1}$$

where w is the element that represents an operation in the AST, $C(x)$ is the frequency of occurrence of token x, and n is the index of the element of interest. In other words, the MLE is the probability that the next element in the token is w_n given the current token.

IV. Results and Analysis

The concepts were implemented using the methods described above in C++ and Python. The open-source synthesis tool, Yosys [15], was used to extract the AST from a design described in Verilog. Once the AST is obtained, the q-gram model is extracted and analyzed such that reusable designs as well as design patterns can be suggested to the user.

To evaluate the validity of the birthmark, a database of circuits was constructed. The circuits in the database were extracted from a variety of sources such as OpenCores and Trust-Hub, totaling more than 250 circuits with varying levels of complexity and size [16] [17] [18] [19] [20].

1) Evaluation of Model Using Cross Validation: q-gram models that predict subsequent operations are typically evaluated using two measures that are commonly used in assessing information retrieval applications: precision and applicability.

$$precision = \frac{P_c}{P_n} \qquad applicability = \frac{P_n}{P_s} \tag{2}$$

Precision indicates the number of correct predictions over all predictions. Applicability is the number of predictions made out of the total number of attempts made to search for the prefix [21]. Typically, recall is used instead of applicability; however, for a prediction system, recall measures the ratio of relevant items retrieved over all possible relevant items. The concept of relevance is obscure because the interest is not to determine if a circuit is relevant or not, but rather how relevant.

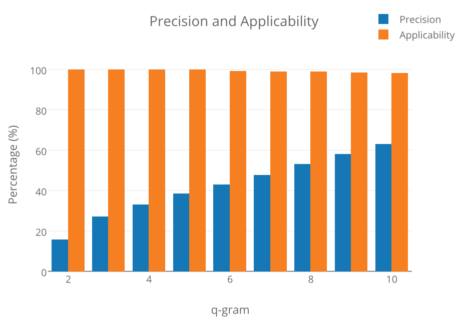

Fig. 3. The precision and applicability for different values of q obtained from cross validation of the circuit database.

TABLE II
Ranking results for various modules of different types

	mmuart	cf_fft_256_18	generic_dpram
1	mmuart_transceiver*	cf_fft_256_16*	ram_sp_sr_sw [18]
2	rtfSimpleUartRx*	cf_fft_256_8*	ram_sp_ar_sw [18]
3	rtfSimpleUart*	cf_fft_512_8*	true_dpram_sclk [16]
4	uart_rx_only	cf_fft_512_16*	generic_mem_s*
5	tiny_spi*	cf_fft_512_18*	generic_mem_m*
6	RS232_uart_t300 ◇	sdft_4_4_sd [22]	dpram [20]
7	SPI_XIF*	usbf_top*	behave1p_mem*
8	rtfSimpleUartTx*	pipelined_fft_64*	behave2p_mem*
9	RS232_uart_t200 ◇	minimac*	ram_dual [16]
10	RS232_uart_t500 ◇	dsp_core [20]	ram_infer [16]

Reference: (*) Opencores database, (◇) Trust-Hub database. The rest are circuits that had been manually designed and/or obtained internally. The reference circuit used is in the first row.

A k-fold cross validation is performed on the q-gram model with $k = 10$, where one-tenth of the dataset is used as test and the other is used as training data. The result of the cross validation is seen in Figure 3. Due to the extension to include tokens smaller than q, the applicability hardly changes as q increases. This is desired because for almost every search query, a prediction can be generated.

As q increases, not only does precision increase, but execution time. If the result of the comparison takes too long, the current circuit state could have changed significantly such that the prediction is no longer valid. With the intent of balancing precision and performance, a 6-gram model will be used for the subsequent case studies and experiments.

2) Circuit Similarity Matching: Table II presents the top ten circuits that are similar to the query in ranked order. Three different reference circuits in different domains were observed. A mean average precision (MAP) was used to assess how well the birthmarking model ranks relevant circuits. The MAP of the results in Table II with a precision of 10 is 0.749, showing an improvement in finding similar designs compared to [11].

A. Case Study: Functional Block Prediction

A case study was designed to investigate the use of extending the q-gram model in attempting to predict the next most likely design or module. In other words, based on the IPs and modules being used, what is the most likely module to be next in the datapath. This then becomes a system-level design. The raise in abstraction level is a common method to increasing overall productivity of a designer. A more ideal use case would be to use this approach in a graphical design environment such as LabVIEW where the components are essentially drag, drop, and connect.

Designs containing commonly used IPs are limited. Therefore, the q-gram model is built not based on IPs and modules used, but the basic functional blocks of the design, such as adders, shifters, logic operators, etc in order to show the potential feasibility of this approach. The reference circuit in design is a Sobel operator, commonly used in image

TABLE III
FUNCTIONAL BLOCK PREDICTION AT DIFFERENT TIME STEPS

Timestep	Q-Gram	Prediction 1	Prediction 2
1	AS	A (Adder)	M (Mux)
2	ASAA	M (Mux)	L (Logic)
3	ASAA	E (Equality)	X (Multiplier)
4	SAANLAMA	M (Mux)	L (Logic)

processing applications to detect edges. The q-gram model was trained on all the circuits in the database.

In Figure 4, the Sobel code is partitioned into five time steps in order to mimic different stages of the circuit under design. The top two results of the functional block prediction can be seen in Table III. At $T1$, a simple two element path consisting of a subtraction and a shift can be seen. With AS (add shift (line 9)) as the current q-gram, the tool predicted the next node as an add operation, reflected by the addition on line 17.

It is important to note that the circuits used to train the q-gram can heavily influence the prediction model. Models can be trained and used depending on the domain of the application. For example, for a filter design, one would want to use a model that best captures design patterns and trends associated with filters or DSP designs.

```
 1 module sobel_t7(p0, p1, p2, p3, p5, p6, p7, p8, out);
 2   input  [7:0]  p0,p1,p2,p3,p5,p6,p7,p8;  // 8 bit pixels inputs
 3   output [7:0]  out;                      // 8 bit output pixel
 4   wire signed [10:0]  gx,gy, gx1,gx2,gx3, gy1,gy2,gy3;
 5   wire signed [10:0]  abs_gx,abs_gy;
 6   wire [10:0]  sum;
 7
 8   assign gx1=(p2-p0);
 9   assign gx2=((p5-p3)<<1);                //Time Step 1
10   assign gx3=(p8-p6);
11   assign gy1=(p0-p6);
12   assign gy2=((p1-p7)<<1);
13   assign gy3=(p2-p8);
14
15   assign gx=(gx1+gx2+gx3);                //Time Step 2
16   assign gy=(gy1+gy2+gy3);
17   assign abs_gx = (gx[10]? ~gx+1 : gx);   //Time Step 3
18   assign abs_gy = (gy[10]? ~gy+1 : gy);
19
20   assign sum = (abs_gx+abs_gy);           //Time Step 4
21   assign out = (|sum[10:8])?8'hff : sum[7:0];
22 endmodule
```

Fig. 4. The reference design of a Sobel operator

V. CONCLUSION

A birthmarking technique based on the q-gram model for extracting and analyzing circuits was presented. The q-gram model was used to show that reuse prediction can provide patterns and similar designs throughout the design phase. Many of these ideas can also be extended towards software productivity, hardware security, and reverse engineering.

This paper presented a birthmarking technique using q-grams as a means to efficiently extract and compare hardware designs. Methods for analysing circuit information in order to predict similar designs and future operations, given snapshots of the reference design, were described.

Much of this work is still in its infancy. Preliminary experiments and results show that this approach yields promising results in suggesting similar circuits as well as predicting subsequent operations. The intuition is the designer is unaware of emerging circuits and design patterns. The goal of this work is to empower designers with the combined knowledge of all the circuits in the world.

REFERENCES

[1] H. Foster, "Why the design productivity gap never happened," in *Computer-Aided Design (ICCAD), 2013 IEEE/ACM International Conference on*, Nov 2013, pp. 581–584.
[2] S. Sikand. Design reuse, verification reuse, and dependency management. [Online]. Available: http://icmanage.com/component/technicalpapers/technicalpapers.html?id=18&Itemid=185
[3] A. Reutter and W. Rosenstiel, "An efficient reuse system for digital circuit design," in *Proceedings of the conference on Design, automation and test in Europe*. ACM, 1999, p. 9.
[4] P. Oehler, I. Vollrath, P. Conradi, and R. Bergmann, "Are you readee for ips," in *Proceedings of the 2nd Workshop Reuse Techniques for VLSI Design*, 1998.
[5] J. Whitham, "A Graph Matching Search Algorithm for an Electronic Circuit Repository," *Univ. of York*, 2004.
[6] K. Zeng and P. Athanas, "Enhancing productivity with back-end similarity matching of digital circuits for ip reuse," in *Reconfigurable Computing and FPGAs (ReConFig), 2013 International Conference on*, Dec 2013, pp. 1–6.
[7] X. Shi, D. Zeng, Y. Hu, G. Lin, and O. R. Zaiane, "Enhancement of incremental design for FPGAs using circuit similarity," *Proceedings of the 12th International Symposium on Quality Electronic Design, ISQED 2011*, pp. 243–250, 2011.
[8] K.-H. Chang, D. Papa, I. Markov, and V. Bertacco, "Invers: An incremental verification system with circuit similarity metrics and error visualization," pp. 487–494, March 2007.
[9] H. Tamada, M. Nakamura, A. Monden, and K.-i. Matsumoto, "Design and evaluation of birthmarks for detecting theft of java programs." in *IASTED Conf. on Software Engineering*, 2004, pp. 569–574.
[10] G. Myles and C. Collberg, "K-gram based software birthmarks," *Proceedings of the 2005 ACM symposium on Applied computing - SAC '05*, p. 314, 2005.
[11] K. Zeng and P. Athanas, "Discovering reusable hardware using birthmarking techniques," in *Information Reuse and Integration (IRI), 2015 IEEE International Conference on*, Aug 2015, pp. 106–113.
[12] X. Zhao, C. Xiao, X. Lin, and W. Wang, "Efficient graph similarity joins with edit distance constraints," in *Data Engineering (ICDE), 2012 IEEE 28th International Conference on*, April 2012, pp. 834–845.
[13] P. F. Brown, P. V. Desouza, R. L. Mercer, V. J. D. Pietra, and J. C. Lai, "Class-based n-gram models of natural language," *Computational linguistics*, vol. 18, no. 4, pp. 467–479, 1992.
[14] A. Z. Broder, "On the resemblance and containment of documents," in *Compression and Complexity of Sequences 1997. Proceedings*. IEEE, 1997, pp. 21–29.
[15] C. Wolf, "Yosys open synthesis suite," http://www.clifford.at/yosys/.
[16] Altera. [Online]. Available: https://www.altera.com/support/support-resources/design-examples/design-software/verilog.html
[17] Dsp examples. [Online]. Available: http://people.ece.cornell.edu/land/courses/ece5760/DE2/fpgaDSP.html
[18] Asic world. [Online]. Available: http://www.asic-world.com/examples/verilog/index.html
[19] Doulous. [Online]. Available: http://www.doulos.com/knowhow/verilog_designers_guide/models/
[20] Scu-rtl benchmark. [Online]. Available: http://ftp.engr.scu.edu/pub/smourad/scu-rtl-bench/
[21] Z. Su, Q. Yang, Y. Lu, and H. Zhang, "Whatnext: A prediction system for web requests using n-gram sequence models," in *Web Information Systems Engineering, 2000. Proceedings of the First International Conference on*, vol. 1. IEEE, 2000, pp. 214–221.
[22] Spiral. [Online]. Available: http://www.spiral.net/hardware/dftgen.html