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 and code 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

The productivity gap that exists between silicon density and design capabilities is too well known. Design reuse has been a prominent factor in bridging that gap, reducing design time and increasing overall productivity in not only software, but hardware as well [1] [2]. However, design reuse has a cost which depends on the time associate with reusing existing 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, forcing the designer to leave the design scope. Considering that there are no universally accepted standards, IP repositories may be unorganized with little or incomplete documentation. Searches may result in no matches found even though the circuit exists under another ”keyword”. In this case, the designer is simply *unaware* of the best way to retrieve the design that is desired. On the other hand, the circuit that is being searched for may not in fact exist at all.

Figure 1 shows the high-level concept of the proposed information reuse model. As the user makes changes to a reference circuit, the proposed tool will continuously monitor the circuit during the entry process. A vast number of comparisons are

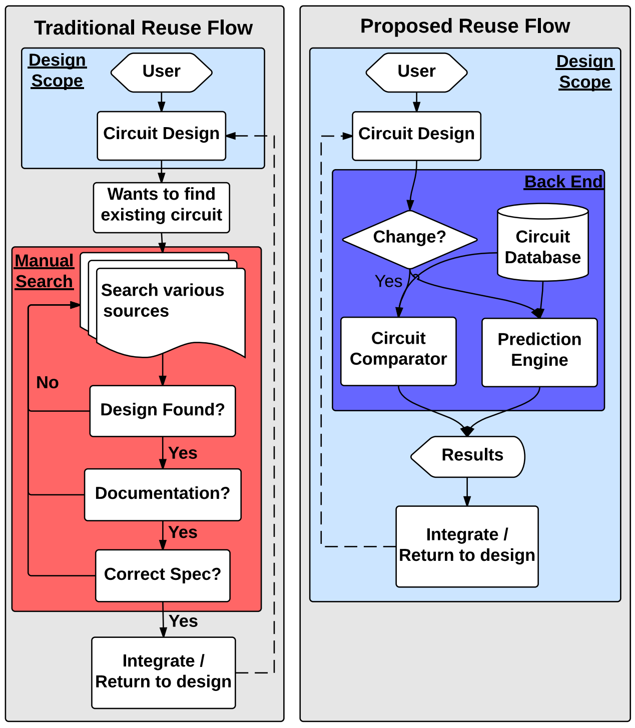

Fig. 1. Comparison between the proposed flow and the traditional flow for reusing designs

performed between the emerging design and 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. In addition, the underlying characteristics of the circuits are extracted and compared in order to predict the *intent* of the user. Candidate circuits and operations are then suggested to the user within the design environment in an unobtrusive manner. The results are ranked based on similarity such that the most relevant information is presented to the user first. The user can then integrate the information suggested into the design or interface with the tool to explore similar circuits and view design documentations.

In this paper, a methodology for comparing the similarity of two circuits using a q-gram *birthmarking* approach is pro-

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.

posed. A *birthmark* is defined by the inherent characteristics of a circuit and is constructed such that the structural and functional information of the circuit is captured. Structural information includes the components used and the overall layout of the circuit and functional information pertains to the dataflow of the circuit. 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 code prediction.

II. Related Work

With essentially every design out there at your fingertips, methods for analyzing and retrieving information from circuits become more prevalent and necessary. Reuse systems commonly use meta-data [3] and case-based reasoning (CBR) [4] as a way to help manage existing designs. Many of these approaches require the user to explicitly leave the design environment and manually search for designs to reuse. The user will also have to describe the desired circuit in a format specific to the reuse system. Furthermore, the reuse system has to be manually built, such as assigning meta-data to circuits, which can be extremely time consuming, especially for large repositories. On the other hand, [5] finds existing designs by using subgraph isomorphism between a query circuit and a database of circuits. This approach is ideal since no manual preprocessing is needed; however, the method of comparison is too restricting and fails to produce a match when two circuits are functionally equivalent but structurally different.

In terms of circuit comparison, there has been little work done in assessing the *similarity* between two circuits. Shi et al [6] used a modified version of an iterative graph similarity algorithm for molecular graphs in order to find similarities between pairs of nodes for placement on FPGAs. InVerS [7] determines a similarity factor between two netlists based on signatures of the nets obtained using a fast simulation technique and is focused more 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. Tamada et al [8] first introduced the idea of a birthmark which is a simplified representation of the software that is extracted from only the source itself. A wide variety of representations for birthmarks have been explored. Myles et al [9] modeled the instructions at the opcode level using a k-gram approach to detect software theft. On the other hand, Chen et al [10] used a sequence of system calls as their birthmark with sequence alignment techniques to assess the similarity of the software. One distinction between hardware and software is that software is inherently sequential whereas the structure of hardware is much more complex. Therefore, different techniques and methods are needed in order to apply the concepts and ideas of birthmarks to hardware designs.

III. Hardware Birthmarks

A hardware *birthmark* is defined here as a set of features or characteristics that describes the underlying structure and functionality inherent to a specific circuit. A birthmark is formally defined as follows:

Definition 1. Let x and z be two programs and let b be the function that extracts the birthmark from a given program. $b(x)$ 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 $b(x) = b(z)$

Due to the complex nature of digital circuits, the problem of analyzing the similarity of two designs is a difficult task. Circuit functions can be represented in countless different ways. In addition, different designers may also have different definitions of what is considered similar. Digital circuits are commonly described in a hardware description language (HDL) such as Verilog. All programming languages have an inherent structure to them and can typically be represented as an abstract syntax tree (AST). Comparing the AST directly can be computationally expensive for very large designs and unfruitful due to the diverse ways of describing a given function. Therefore, the AST is broken down into smaller parts and modeled using a path-based q-gram approach [11]. Figure 2 shows the overall system flow for the proposed reuse model.

IV. q-gram Hardware Birthmarks

A q-gram (n-gram) approach is commonly used in natural language processing (NLP) when trying to assess the similarity of two documents. The text of each document is decomposed into sequences of q length using a sliding window approach resulting in a model that represents the overall document. Similarity between two documents can be determined based on the number of shared q-grams found. The q-gram model can also be used for text prediction by using a Markov model [12]. This idea can be leveraged to predict the next most likely operation or design pattern to be used in the circuit. More ambitiously, this approach could potentially be used to predict the intent of the circuit the user is designing.

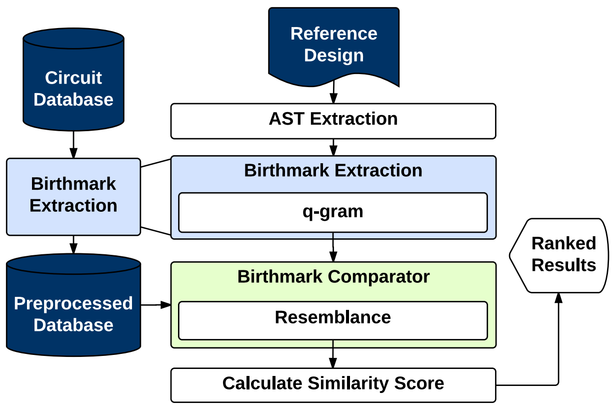

Fig. 2. Block diagram of overall system flow

A. Path-based q-gram

The idea of q-grams works well with sequential data such as text in a document. For non-sequential data such as graphs, q-gram can be extended by using a path-based q-gram approach. Instead of a sliding window of size q across a sequence, paths of length q are selected. Path-based q-grams are formally defined as follows:

Definition 2. Let $G(V, E)$ be a graph that represents the circuit. A path-based q-gram model of G is a collection of simple paths starting from $v \in V$ of length q, for all nodes in V,

Figure 3 shows a path-based q-gram model for a basic 1-bit counter with the bigrams (2-gram) and trigrams (3-grams) extracted. The nodes of the AST correspond to primitive operations such as add, shift, multiply, logic operators and more. Therefore, the elements of the q-gram are the operations themselves. Each operation is assigned a letter in the alphabet for simplicity.

1) Path-based q-gram Schemes: The accuracy or tightness of the match can also be decided 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. In a list-based scheme, the ordering of each element inside the q-gram is maintained. Figure 3 shows a strict list-based scheme where the order of the items in the q-gram is preserved. The set-based scheme is an extension of the list-based scheme where the frequency and the ordering of the elements are ignored. The frequency-based scheme maintains the frequency information but ignores the order of the elements.

Depending on the goal of the application, a different path-based q-gram scheme might be chosen. For example, the list-based scheme is useful for plagiarism and theft detection because it maintains ordering and frequency which can reduce the number of false-positives. On the other hand, if one was searching for a similar circuit in general, a looser search criteria might be desired. For the remainder of this paper, the frequency-based q-gram approach is used.

2) Extension to (q to 1)-gram: The notion of a q-gram is extended further to include the q-grams less than q. In other words, the q-gram model is a combination of n-grams where $1 < n \leq q$. This model was chosen because paths of length q may not exist in certain circuits. For example, the circuit in Figure 3 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

To calculate the similarity of the two birthmarks, Jaccard's index is used. Jaccard's index is commonly used to compare the similarity of two sample sets. It is also used as a way to quantitatively measure the *resemblance* of two documents [17]. Jaccards index is defined in Equation 1.

$$\frac{|b(x) \cap b(z)|}{|b(x) \cup b(z)|} \tag{1}$$

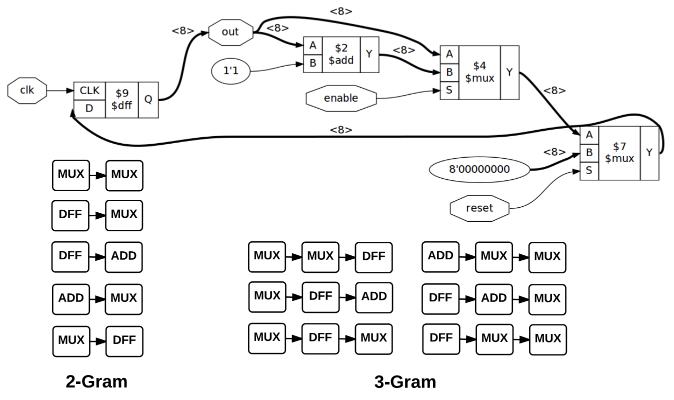

Fig. 3. Path-based 2-gram and 3-gram model for a basic 1-bit counter

Using Jaccards index, the q-gram models of two circuits are compared with each other such that the similarity between the two circuits are the number of q-grams that they share in common. Greater weight is placed on the q-grams that have a higher q.

C. Predicting Design Patterns

The q-gram model can be extended such that future elements in a sequence 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 q-gram observed and using a maximum likelihood estimator (MLE) to predict the next most likely function. More formally:

Definition 3. The MLE is defined to be

$$P(w_n | w_{n-q+1}^{n-1}) = \frac{C(w_{n-q+1}^{n-1} w_n)}{C(w_{n-q+1}^{n-1})} \tag{2}$$

where w is the element that represents an operation in the AST, $C(x)$ is the frequency of occurrence of sequence x, and n is the index of the element of interest.

In other words, the MLE is the probability that the next item in the sequence is w_n given the current sequence. This is computed by dividing the frequency of the concatenation of the current sequence and w_n by the frequency of the current sequence.

V. Results and Analysis

The concepts were implemented using the methods described above in C++ and Python. The open-source synthesis tool, Yosys [18], was used to extract the AST from a design described in Verilog. Verilog was chosen as the input to the system due to its wide use and preference in the hardware community. Furthermore, graphical environments such as LabView can be leveraged since their overall programming paradigm can be seen as an AST. 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.

TABLE I
RANKING RESULTS FOR VARIOUS MODULES OF DIFFERENT TYPES

Rank	mmuart	cf_fft_256_18	altera_sig_mult	generic_dpram
1	mmuart_transceiver*	cf_fft_256_16*	sig_altmult_add [13]	ram_sp_sr_sw [14]
2	rtfSimpleUartRx*	cf_fft_256_8*	unsigned_mult [13]	ram_sp_ar_sw [14]
3	rtfSimpleUart*	cf_fft_512_8*	qmult2*	true_dpram_sclk [13]
4	uart_rx_only	cf_fft_512_16*	clock_divider*	generic_mem_small*
5	tiny_spi*	cf_fft_512_18*	addsub [13]	generic_mem_medium*
6	RS232_uart_t300 ◇	spiraldft_4_4_stream_date [15]	counter [13]	dpram [16]
7	SPI_XIF*	usbf_top*	up_counter [14]	behave1p_mem*
8	rtfSimpleUartTx*	pipelined_fft_64*	clk_div [14]	behave2p_mem*
9	RS232_uart_t200 ◇	minimac*	ternary_addtree [13]	ram_dual [13]
10	RS232_uart_t500 ◇	dsp_core [16]	single_port_ram [13]	ram_infer [13]

Reference: (*) Opencores database, (◇) Trust-Hub database. The rest are circuits that had been manually designed and/or obtained internally.

To evaluate the validity of the birthmarking approach, 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 [13] [19] [14] [20] [16].

1) Circuit Similarity Matching: Table I presents the top ten existing circuits that are similar to the query in ranked order. Four different reference circuits are observed, each in one of four domains: communications, digital signal processing, arithmetic, and memory. To quantify and evaluate the performance of the ranking system, a mean average precision (MAP) was used to assess how well the birthmarking model ranks relevant circuits. The MAP of the results provided in Table I with a precision of 10 is 0.812. Figure 4 shows an autocorrelation between several domain specific circuits.

2) Evaluation of Model Using Cross Validation: The application of q-gram models to predict digital circuits is evaluated using two measures that are commonly used in assessing information retrieval applications: precision and applicability.

$$precision = \frac{P_c}{P_n} \tag{3}$$

$$applicability = \frac{P_n}{P_s} \tag{4}$$

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, the recall measure is not well suited because it measures the ratio of relevant items retrieved over all possible relevant items. The concept of relevance is obscure in this type of system because the interest is not to determine if a circuit is relevant or not, but rather how relevant.

A k-fold cross validation is performed on the q-gram model where $k = 10$. During the cross validation, both the precision and the applicability of the test set is calculated. Figure 5 shows the precision and applicability for various q. Due to the extension of n-grams smaller than q, the applicability hardly changes as q increases. This is desired because for almost every search query, a prediction can be generated. On the other hand, a higher q decreases performance of the system shown in Figure 6.

If the time between when the circuit snapshot was taken and when the result is ready is too long, the current state could have changed significantly such that the prediction is no longer valid. Generally, most studies [22] conclude that ten seconds is around the typical waiting time for any user. With the intent of balancing precision and performance, a 6-gram model will be used for the subsequent case studies and experiments.

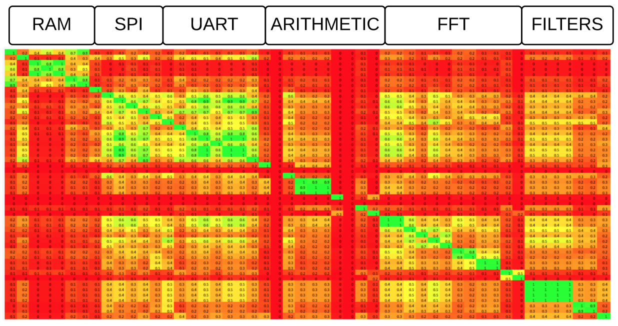

Fig. 4. Autocorrelation heatmap of a subset of circuits where red represent dissimilarity and green represents similarity.

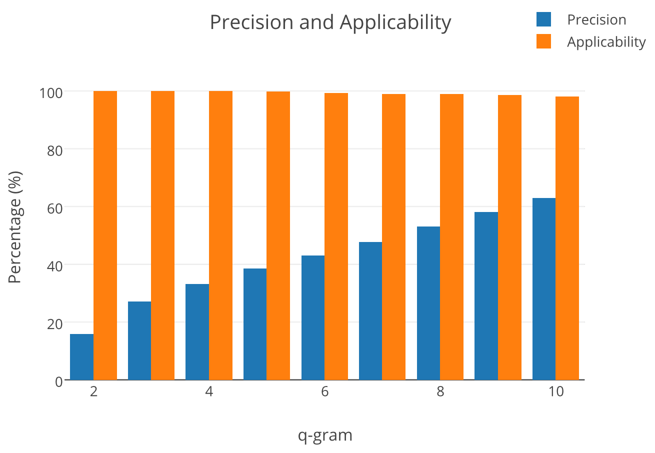

Fig. 5. The precision and applicability for different values of q obtained from cross validation of the circuit database.

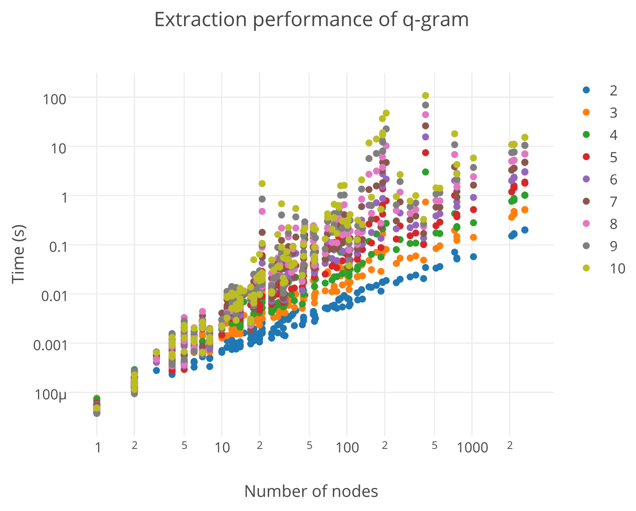

Fig. 6. Performance of extracting q-grams from circuits of different sizes with varying q values.

```
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;
5    wire signed [10:0] gx1,gx2,gx3, gy1, gy2, gy3; // Intermediate
6    wire signed [10:0] abs_gx,abs_gy;
7    wire [10:0] sum;
8
9    assign gx1=(p2-p0);
10   assign gx2=((p5-p3)<<1);              //Time Step 1
11   assign gx3=(p8-p6);
12
13   assign gy1=(p0-p6);
14   assign gy2=((p1-p7)<<1);
15   assign gy3=(p2-p8);
16
17   assign gx=(gx1+gx2+gx3);              //Time Step 2
18   assign gy=(gy1+gy2+gy3);
19
20   assign abs_gx = (gx[10]? ~gx+1 : gx); //Time Step 3
21   assign abs_gy = (gy[10]? ~gy+1 : gy);
22
23   assign sum = (abs_gx+abs_gy);         //Time Step 4
24
25   assign out = (|sum[10:8])?8'hff : sum[7:0];
26
27 endmodule
```

Fig. 7. The reference design of a Sobel operator

A. Case Study: Functional Block Prediction

A case study was designed to investigate the use of extending the q-gram model in attempting to aid the designer by predicting future operations. The reference circuit in design is a Sobel operator, commonly used in image processing applications to detect edges by approximating the gradient of the intensity function of the image. The module takes in a 3x3 pixel block from the image and approximates the derivatives for changes in both the horizontal and vertical direction by using a 3x3 spatial mask.

In Figure 7, the Sobel operator 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 II. At $T1$, a simple two block path consisting of a subtraction and a shift can be seen. With AS (add shift (line 10)) as the current q-gram, the tool predicted the next node as an add operation, reflected in 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 filter or DSP designs.

TABLE II
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)

B. Case Study: HDL Code Prediction

For HDL design, functional block prediction can have little meaning or effect since many of the operations can be combined into one line. Therefore, instead of predicting functional blocks, lines of similar HDL code can be suggested. Each functional node has a source attribute which indicates the associated line in the HDL source. The current line position is noted in the reference design and only the q-grams associated with the current line position are compared with the q-gram model of each circuit.

The lines of HDL predicted for the Sobel operator under design at $T1$ are shown in figures 8 and 9. The lines suggested in Figure 8 were extracted from a 4-point casual moving average filter where the operations are very similar to the reference. The code suggested in Figure 9 was extracted from a complete Sobel operator. Both suggested lines of code consist of an adder and a shifter block similar to the statement extracted from the incomplete reference circuit.

The limitation and difficulty in HDL code prediction from the q-gram model is that the lines of code associated with a particular sequence are usually different regardless which circuit the sequence was found in. For example, the bigram AS associated with the current line position in the reference design was found in two different circuits: `firfilter` and `sobel`. Line 3 in Figure 8 and lines 3 and 4 in Figure 9 are associated with that bigram. One way to provide more relevant code prediction is to rank the predicted HDL code based on how similar the reference circuit is to the circuit the code is extracted from.

```
...
    D0 <= D1;
    Dout <= ( (D0) + (D1) + (D2) + (D3) ) << 2
  end
...

```

Fig. 8. First-ranked predicted HDL code from a 4-point causal moving average filter

```
...
  wire [10:0] sum;
  assign gx=((p2−p0)+((p5−p3)<<1)+(p8−p6));
  assign gy=((p0−p6)+((p1−p7)<<1)+(p2−p8));
...

```

Fig. 9. Second-ranked predicted HDL code from Sobel operator

VI. 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 fine-grain reuse prediction can provide design patterns and hints as well as similar designs throughout the design phase.

Future work could explore various q-gram schemes such as using $k-AT$ trees [23] where instead of paths, a tree-based q-gram is constructed such that each vertex in the q-gram can be reached in q hops. Furthermore, the q-gram prediction model can be extended to module prediction based on hierarchy information. 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. Many of these ideas can be extended towards software productivity in term of extracting information out of existing software in order to aid programmers when coding.

The key challenge in terms of analyzing and extracting information out of digital circuits is how to best represent a hardware design such that meaningful data can be extracted and easily compared. Key contributions in this paper are

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

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 that can be applied and 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] 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.

[7] K.-h. Chang, D. A. Papa, I. L. Markov, and V. Bertacco, "Invers: an incremental verification system with circuit similarity metrics and error visualization," pp. 487–494, 2007.

[8] 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.

[9] G. Myles and C. Collberg, "K-gram based software birthmarks," *Proceedings of the 2005 ACM symposium on Applied computing - SAC '05*, p. 314, 2005.

[10] D. Kim, S.-j. Cho, S. Han, M. Park, and I. You, "Open Source Software Detection using Function-level Static Software Birthmark," *Journal of Internet Services and ...*, vol. 4, no. November, pp. 25–37, 2014.

[11] 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.

[12] 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.

[13] Altera. [Online]. Available: https://www.altera.com/support/support-resources/design-examples/design-software/verilog.html

[14] Asic world. [Online]. Available: http://www.asic-world.com/examples/verilog/index.html

[15] Spiral. [Online]. Available: http://www.spiral.net/hardware/dftgen.html

[16] Scu-rtl benchmark. [Online]. Available: http://ftp.engr.scu.edu/pub/smourad/scu-rtl-bench/

[17] A. Z. Broder, "On the resemblance and containment of documents," in *Compression and Complexity of Sequences 1997. Proceedings*. IEEE, 1997, pp. 21–29.

[18] C. Wolf, "Yosys open synthesis suite," http://www.clifford.at/yosys/.

[19] Dsp examples. [Online]. Available: http://people.ece.cornell.edu/land/courses/ece5760/DE2/fpgaDSP.html

[20] Doulous. [Online]. Available: http://www.doulos.com/knowhow/verilog_designers_guide/models/

[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] F. F.-H. Nah, "A study on tolerable waiting time: how long are web users willing to wait?" *Behaviour & Information Technology*, vol. 23, no. 3, pp. 153–163, 2004.

[23] G. Wang, B. Wang, X. Yang, and G. Yu, "Efficiently indexing large sparse graphs for similarity search," *Knowledge and Data Engineering, IEEE Transactions on*, vol. 24, no. 3, pp. 440–451, March 2012.