Complementary Synthesis for Pipelined Encoder

Abstract— Complementary synthesis automatically generates an encoder's decoder that recovers the encoder's inputs variables from its output variables. However, the Boolean function of the decoder are characterized by Craig interpolant, and thus include lots of random logic gates that make it unnecessarily large and difficult to be understood by human. By studying the structure of many encoders from real industrial projects, we found that most of them have a pipeline structure that can be exploited to overcome these two problems.

Thus, we propose a novel algorithm to first find out the encoder's pipeline registers in each pipeline stage, and then characterize the Boolean function of these pipeline registers and the encoder's input variables with support set from the next pipeline stage.

Experimental results on several complex encoders indicate that this algorithm can always correctly infer the encoder's pipeline structure, and generate the Boolean functions for the pipeline registers and input variables. Furthermore, the circuit area are significantly reduced, and the generated decoder's structure are much more easier to be understood.

I. Introduction

One of the most difficult jobs in designing communication and multimedia chips is to design and verify complex encoder and decoder pairs. The encoder maps its input variables $\vec{i}$ to its output variables $\vec{o}$, while the decoder recovers $\vec{i}$ from $\vec{o}$. Complementary synthesis [9, 7, 8, 6, 3, 4, 10] eases this job by automatically generating a decoder from an encoder, with the assumption that $\vec{i}$ can always be uniquely determined by a bounded sequence of $\vec{o}$. Thus, the decoder's Boolean function can be characterized with the algorithm proposed by Jiang et al. [2] based on Craig interpolant [1].

we can first says that there are pipeline struct, and then describe the problem of current algo on these

However, the decoders generated in this way have two major shortcomings:

1. Its circuit area is unnecessarily large because some common logic for two different input variables $i_1, i_2 \in \vec{i}$ are hidden deeply in the two Boolean function computed by Craig interpolants.

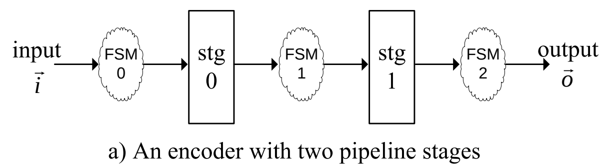

Fig. 1. The under-approximative approach checking if i_{p+l} can be uniquely determined

2. The decoder's circuit structure are lost, which make it very difficult to be understood by human engineers.

By studying the structure of many encoders from real industrial projects, we find that most of them have a pipeline structure that can be exploited to overcome these two problems. As shown in Figure [**?**],

II. Preliminaries

A. Propositional satisfiability

The Boolean value set is denoted as $B = \{0, 1\}$. A vector of variables is represented as $\vec{v} = (v, \dots)$. The number of variables in $\vec{v}$ is denoted as $|\vec{v}|$. If a variable v is a member of $\vec{v}$, then we say $v \in \vec{v}$; otherwise we say $v \notin \vec{v}$. For a variable v and a vector $\vec{v}$, if $v \notin \vec{v}$, then the new vector that contains both v and all members of $\vec{v}$ is denoted as $v \cup \vec{v}$. If $v \in \vec{v}$, then the new vector that contains all members of $\vec{v}$ except v, is denoted as $\vec{v} - v$. For the two vectors $\vec{a}$ and $\vec{b}$, the new vector with all members of $\vec{a}$ and $\vec{b}$ is denoted as $\vec{a} \cup \vec{b}$. The set of truth valuations of $\vec{v}$ is denoted as $[\![\vec{v}]\!]$, for instance, $[\![(v_1, v_2)]\!] = \{(0,0), (0,1), (1,0), (1,1)\}$.

The propositional satisfiability problem(abbreviated as SAT) for a Boolean formula F over a variable set V is to find a satisfying assignment $A : V \to B$, so that F can be evaluated to 1. If A exists, then F is satisfiable; otherwise, it is unsatisfiable.

Given two Boolean formulas ϕ_A and ϕ_B, with $\phi_A \wedge \phi_B$ unsatisfiable, there exists a formula ϕ_I referring only to the common variables of ϕ_A and ϕ_B such that $\phi_A \Rightarrow \phi_I$

and $\phi_I \wedge \phi_B$ is unsatisfiable. We call ϕ_I the **Craig interpolant** [1] of ϕ_A with respect to ϕ_B and use McMillan's algorithm [5] to generate it.

B. Finite state machine

The encoder is modeled by a finite state machine(FSM) $M = (\vec{s}, \vec{i}, \vec{o}, T)$, consisting of a state variable vector $\vec{s}$, an input variable vector $\vec{i}$, an output variable vector $\vec{o}$, and a transition function $T : [\![\vec{s}]\!] \times [\![\vec{i}]\!] \rightarrow [\![\vec{s}]\!] \times [\![\vec{o}]\!]$ that computes the next state and output variable vector from the current state and input variable vector.

The behavior of FSM M can be reasoned by unrolling transition function for multiple steps. The state variable $s \in \vec{s}$, input variable $i \in \vec{i}$ and output variable $o \in \vec{o}$ at the n-th step are respectively denoted as s_n, i_n and o_n. Furthermore, the state, the input and the output variable vectors at the n-th step are respectively denoted as $\vec{s}_n$, $\vec{i}_n$ and $\vec{o}_n$. A **path** is a state sequence $< \vec{s}_n, \ldots, \vec{s}_m >$ with $\exists \vec{i}_j \vec{o}_j (\vec{s}_{j+1}, \vec{o}_j) \equiv T(\vec{s}_j, \vec{i}_j)$ for all $n \leq j < m$. A **loop** is a path $< \vec{s}_n, \ldots, \vec{s}_m >$ with $\vec{s}_n \equiv \vec{s}_m$.

C. The halting algorithm to determine if an input variable can be uniquely determined by a bounded sequence of output variable vector

The first halting algorithm [8] iteratively unrolls the transition function. And for each iteration, it uses two approximative approaches to determine the answer. The first one is an under-approximative one that presented in C.1, while the second one is an over-approximative one presented in C.2. We will show in C.3 that these two approaches will eventually converge to a conclusive answer.

C.1 The under-approximative approach

As shown in Figure 2, on the unrolled transition functions, an input variable $i \in \vec{i}$ can be uniquely determined, if there exist three integers p, l and r, such that for any particular valuation of the output sequence $< \vec{o}_p, \ldots, \vec{o}_{p+l+r} >$, i_{p+l} cannot be 0 and 1 at the same time. This is equal to the unsatisfiability of $F_{PC}(p, l, r)$ in Equation (1).

$$F_{PC}(p,l,r) :=$$

$$\left\{ \begin{array}{cc} & \bigwedge_{m=0}^{p+l+r} \{(\vec{s}_{m+1}, \vec{o}_m) \equiv T(\vec{s}_m, \vec{i}_m)\} \\ \wedge & \bigwedge_{m=0}^{p+l+r} \{(\vec{s'}_{m+1}, \vec{o'}_m) \equiv T(\vec{s'}_m, \vec{i'}_m)\} \\ \wedge & \bigwedge_{m=p}^{p+l+r} \vec{o}_m \equiv \vec{o'}_m \\ \wedge & i_{p+l} \equiv 1 \wedge i'_{p+l} \equiv 0 \\ \wedge & \bigwedge_{m=0}^{p+l+r} assertion(\vec{i}_m) \\ \wedge & \bigwedge_{m=0}^{p+l+r} assertion(\vec{i'}_m) \end{array} \right\} \quad (1)$$

Here, p is the length of the prefix state transition sequence. l and r are the lengths of the two output sequences $< \vec{o}_{p+1}, \ldots, \vec{o}_{p+l} >$ and $< \vec{o}_{p+l+1}, \ldots, \vec{o}_{p+l+r} >$ used to determine i_{p+l}. Line 1 of Equation (1) corresponds to the left path in Figure 2, while Line 2 corresponds to the right path in Figure 2. These two paths are of the same length. Line 3 forces these two paths' output sequences to be the same, while Line 4 forces their i_{p+l} to be different. Line 5 and 6 are the assertion predicates given by the user that constrain the valid valuation on $\vec{i}$. PC in F_{PC} is the abbreviation of "parameterized complementary", which means $F_{PC}(p, l, r)$ is used to check whether the encoder's input can be uniquely determined with the three parameters p, l and r.

According to Figure 2, the first three lines of Equation (1) are two unrolled transition function sequences with the same output sequences. They can always be satisfied with the same input variable vectors and initial state vector. And the last two lines are constraints on input variable vectors. We always check their satisfiability before running our algorithm. So the unsatisfiability of $F_{PC}(p, l, r)$ always means $i_{p+l} \equiv i'_{p+l}$.

According to Equation (1), for $p' \geq p$, $l' \geq l$ and $r' \geq r$, the clause set of $F_{PC}(p', l', r')$ is a super set of $F_{PC}(p, l, r)$. So, the bounded proof of $F_{PC}(p, l, r)$'s unsatisfiability can be generalized to unbounded cases.

Proposition 1 *If $F_{PC}(p, l, r)$ is unsatisfiable, then i_{p+l} can be uniquely determined by $< \vec{o}_p, \ldots, \vec{o}_{p+l+r} >$ for all larger p, l and r.*

C.2 The over-approximative approach

For $F_{PC}(p, l, r)$ presented in last subsection, there are two possibilities:

1. i_{p+l} can be uniquely determined by $< \vec{o}_p, \ldots, \vec{o}_{p+l+r} >$ for some p, l and r;
2. i_{p+l} can't be uniquely determined by $< \vec{o}_p, \ldots, \vec{o}_{p+l+r} >$ for any p, l and r at all.

If it is the 1st case, then by iteratively increasing p, l and r, $F_{PC}(p, l, r)$ will eventually become unsatisfiable. But if it is the 2nd case, this method will never terminate.

So, to obtain a halting algorithm, we need to distinguish these two cases. One such solution is shown in Figure 3, which is similar to Figure 2 but with three additional constraints used to detect loops on the three state sequences $< \vec{s}_0, \ldots, \vec{s}_p >$,$< \vec{s}_{p+1}, \ldots, \vec{s}_{p+l} >$ and $< \vec{s}_{p+l+1}, \ldots, \vec{s}_{p+l+r} >$. It is formally defined in Equation (2) with the last three lines corresponding to the three new constraints used to detect loops.

$$F_{LN}(p,l,r) :=$$

$$\left\{ \begin{array}{cc} & F_{PC}(p,l,r) \\ \wedge & \bigvee_{x=0}^{p-1} \bigvee_{y=x+1}^{p} \{\vec{s}_x \equiv \vec{s}_y \wedge \vec{s'}_x \equiv \vec{s'}_y\} \\ \wedge & \bigvee_{x=p+1}^{p+l-1} \bigvee_{y=x+1}^{p+l} \{\vec{s}_x \equiv \vec{s}_y \wedge \vec{s'}_x \equiv \vec{s'}_y\} \\ \wedge & \bigvee_{x=p+l+1}^{p+l+r-1} \bigvee_{y=x+1}^{p+l+r} \{\vec{s}_x \equiv \vec{s}_y \wedge \vec{s'}_x \equiv \vec{s'}_y\} \end{array} \right\} \quad (2)$$

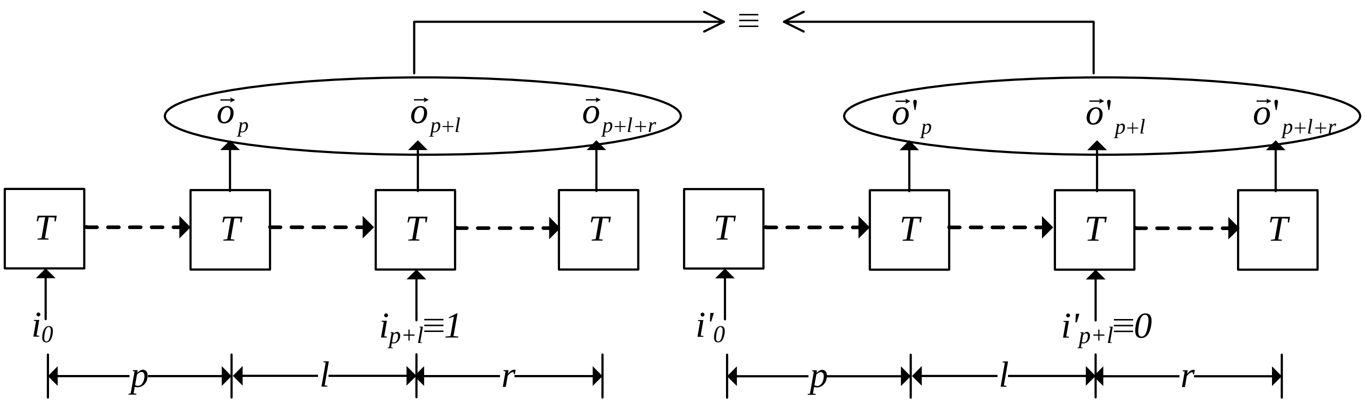

Fig. 2. The under-approximative approach checking if i_{p+l} can be uniquely determined

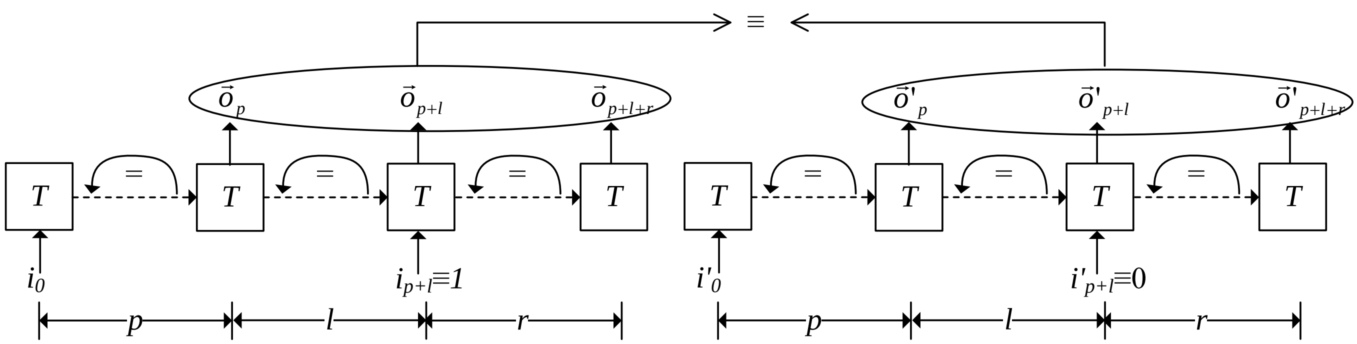

Fig. 3. The over-approximative approach checking if i_{p+l} can NOT be uniquely determined

LN in F_{LN} stands for "loop non-complementary", which means $F_{LN}(p, l, r)$ with three loops is used to check whether the input variable can NOT be uniquely determined.

When $F_{LN}(p, l, r)$ is satisfiable, then i_{p+l} can't be uniquely determined by $< \vec{o}_p, \ldots, \vec{o}_{p+l+r} >$. More importantly, by unrolling these three loops, we can generalize the satisfiability of $F_{LN}(p, l, r)$ to all larger p, l and r. This means:

Proposition 2 *If $F_{LN}(p, l, r)$ is satisfiable, then i_{p+l} cannot be uniquely determined by $< \vec{o}_p, \ldots, \vec{o}_{p+l+r} >$ for all larger p, l and r.*

C.3 The full algorithm

With Propositions 1 and 2, we can generalize their bounded proof to unbounded cases. This leads to the halting Algorithm 1 that search for p, l and r that enable an input variable i_{p+l} to be uniquely determined by the output sequence $< \vec{o}_p, \ldots, \vec{o}_{p+l+r} >$:

1. On the one hand, if there exists such p, l and r, then let $p' := max(p, l, r)$, $l' := max(p, l, r)$ and $r' := max(p, l, r)$. From Propositions 1, we know that $F_{PC}(p', l', r')$ is unsatisfiable. So eventually $F_{PC}(p, l, r)$ will become unsatisfiable in Line 4;

2. On the other hand, if there doesn't exist such p, l and r, then eventually p, l and r will be larger than the encoder's longest path without loop, which means that there will be three loops in $< \vec{s}_0, \ldots, \vec{s}_p >$,$< \vec{s}_{p+1}, \ldots, \vec{s}_{p+l} >$ and $< \vec{s}_{p+l+1}, \ldots, \vec{s}_{p+l+r} >$. This will make $F_{LN}(p, l, r)$ satisfiable in Line 6.

Both cases will lead to this Algorithm's termination. Please refer to [8] for more detail of this algorithm's correctness and termination proof.

III. How to Format the Page

A. Full-Size Camera-Ready Copy

Prepare Initial-Submission paper in full size format, on A4 size or 8 1/2" × 11"(21.5cm × 27.9 cm) paper.

B. Fonts

The best results will be obtained if your computer word-processor has several font sizes. Try to follow the font sizes specified in Table I as best as you can. As an aid to gauging font size, 1 point is about 0.35mm. Use a proportional, serif font such as Times of Dutch Roman.

C. Formats

In formatting your A4-size paper, set top margin to 29mm (1.14 inches), bottom margin to 30mm (1.18 inches), left and right margins to 15mm (0.59 inches) If you are using paper 8 1/2" × 11", set top margin to 17mm (0.67 inches), bottom margin to 24mm (0.94 inches), left margin to 18mm (0.71 inches) and right margins to 17mm (0.67 inches). The column width is 88mm (3.46 inches) with 5mm (0.2 inches) space between the two columns.

You should left- and right-justify your columns. On the last page of your paper, try to adjust the lengths of the two columns so that they are the same. Use automatic hyphenation if you have it and check spelling. Either digitize or paste down your figures.

IV. Figures and Tables

Position figures and tables at the tops and bottoms of columns. Avoid placing them in the middle of columns. Large figures and tables may span across both columns. Figure captions should be below the figures; table captions should be above the tables. Avoid placing figures and tables before their first mention in the text. Use the abbreviation "Fig.1", even at the beginning of a sentence.

The paper's graphics should have resolutions of 600dpi for monochrome, 300 dpi for grayscale, and 300 dpi for color. But, please do not include too much high-resolution images or photos in your paper.

Algorithm 1: *CheckUniqueness*(i): The halting algorithm to determine whether $i \in \vec{i}$ can be uniquely determined by a bounded sequence of output variable vector $\vec{o}$

Input: The input variable $i \in \vec{i}$.
Output: whether $i \in \vec{i}$ can be uniquely determined by $\vec{o}$, and the value of p, l and r.

p:= 0; l:= 0; r:= 0;
while 1 **do**
    p++; l++; r++;
    if $F_{PC}(p,l,r)$ *is unsatisfiable* **then**
        return (1, p, l, r);
    else if $F_{LN}(p,l,r)$ *is satisfiable* **then**
        return (0, p, l, r);

TABLE I
Fonts for Initial-Submission and Camera-Ready Papers

Font Size	Style	Text
14pt	bold	Paper title
12pt		Authors' names
10pt		Authors' affiliations, main text, equations, first letters in section titles[a]
10pt	italic	Subheddings
9pt	bold	Abstract
8pt		Section titles[a], table names[a], first letters in table captions[a], tables, figure captions, references, footnotes, text subscripts and superscripts
6pt		Table captions[a], table superscripts

[a] Uppercase

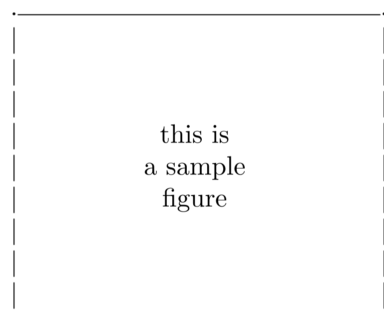

Fig. 4. This is a sample figure. Captions exceeding one line are arranged like this.

V. Helpful Hints

A. References

List and number all references at the end of the paper. When reffering to them in the text, type the corresponding reference number in the the citations consecutively. The sentence punctuation follows the parentheses. Do not use "Ref. [?]" or "reference [?]" except at the beginning of a sentence.

B. Footnotes

Number the footnotes separately in superscripts. Place the actual footnote at the bottom of the column in which it is cited. Do not put footnotes in the reference list.

C. Authors names

Give all authors' names; do not use "et al" unless there are six authors or more. Papers that have not been published, even if they have been submitted for publication, should be cited as "unpublished" [?]. Papers that have been accepted for publication should be cited as "in press" [?]. Capitalize only the first word in a paper title, except for proper nouns and element symbols.

For papers published in translation journals, please give the English citation first, followed by the original foreign language citations [?].

D. Notice for LaTeX users

If you use LaTeX to create your initial-submission paper, we recommend you to use `dvipdfm` to produce PDF files from dvi files. If you cannot use it, please use Type1 fonts instead of ugly Type3 fonts!

VI. Summary and Conclusions

This template can be downloaded through the ASP-DAC 2016 web site (http://www.aspdac.com/). If you have any problem, please contact ASP-DAC 2016 publication chair.

Acknowledgements

This article was written by referring to *"Author's guide – Preparation of Papers in Two-Column Format for VLSI Symposia on Technology and Circuits"*, the *"Preparation of Papers in Two-Column Format for the Proceedings of the 32nd ACM/IEEE Design Automation Conference"* written by Ann Burgmeyer, IEEE and *"the template for producing IEEE-format articles using LaTeX"*, written by Matthew Ward, Worcester Polytechnic Institute.

References

[1] W. Craig. Linear reasoning: A new form of the herbrand-gentzen theorem. *The Journal of Symbolic Logic*, 22(3):250–268, Sept. 1957.

[2] W.-L. H. Jie-Hong Roland Jiang, Hsuan-Po Lin. Interpolating functions from large boolean relations. In *Proceedings of 2009 International Conference on Computer-Aided Design*, ICCAD '09, pages 779–784. IEEE, 2009.

[3] H.-Y. Liu, Y.-C. Chou, C.-H. Lin, and J.-H. R. Jiang. Towards completely automatic decoder synthesis. In *Proceedings of the 2011 International Conference on Computer-Aided Design, ICCAD 2011*, ICCAD '11, pages 389–395, San Jose, CA, USA, 2011. IEEE Press.

[4] H.-Y. Liu, Y.-C. Chou, C.-H. Lin, and J.-H. R. Jiang. Automatic decoder synthesis: Methods and case studies. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 31(9):31:1319–31:1331, September 2012.

[5] K. L. McMillan. Interpolation and sat-based model checking. In F. S. Warren A. Hunt Jr., editor, *Computer Aided Verification, 15th International Conference, CAV 2003*, volume 2725 of *Lecture Notes in Computer Science*, pages 1–13. Springer-Verlag, Berlin Heidelberg, 2003.

[6] S. Shen, Y. Qin, K. Wang, Z. Pang, J. Zhang, and S. Li. Inferring assertion for complementary synthesis. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 31(8):31:1288–31:1292, August 2012.

[7] S. Shen, Y. Qin, K. Wang, L. Xiao, J. Zhang, and S. Li. Synthesizing complementary circuits automatically. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 29(8):29:1191–29:1202, August 2010.

[8] S. Shen, Y. Qin, L. Xiao, K. Wang, J. Zhang, and S. Li. A halting algorithm to determine the existence of the decoder. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 30(10):30:1556–30:1563, October 2011.

[9] S. Shen, J. Zhang, Y. Qin, and S. Li. Synthesizing complementary circuits automatically. In *Proceedings of the 2009 International Conference on Computer-Aided Design*, ICCAD '09, pages 381–388, San Jose, CA, USA, 2009. IEEE Press.

[10] K.-H. Tu and J.-H. R. Jiang. Synthesis of feedback decoders for initialized encoders. In *Proceedings of the 50th Annual Design Automation Conference, DAC 2013*, DAC '13, pages 1–6, Austin, TX, USA, 2013. ACM Press.