QBF-Based Boolean Function Bi-Decomposition

Huan Chen
University College Dublin
Dublin, Ireland

Mikoláš Janota
INESC-ID
Lisbon, Portugal

Joao Marques-Silva
University College Dublin
Dublin, Ireland

***Abstract*—Boolean function bi-decomposition is ubiquitous in logic synthesis. It entails the decomposition of a Boolean function using two-input simple logic gates. Existing solutions for bi-decomposition are often based on BDDs and, more recently, on Boolean Satisfiability. In addition, the partition of the input set of variables is either assumed, or heuristic solutions are considered for finding good partitions. In contrast to earlier work, this paper proposes the use of Quantified Boolean Formulas (QBF) for computing bi-decompositions. These bi-decompositions are optimal in terms of the achieved quality of the input set of variables. Experimental results, obtained on representative benchmarks, demonstrate clear improvements in the quality of computed decompositions, but also the practical feasibility of QBF-based bi-decomposition.**

I. Introduction

Bi-decomposition is arguably the most widely used form of Boolean function decomposition. Bi-decomposition [2]–[4], [6]–[8] consists of decomposing Boolean function $f(X)$ into the form of $f(X) = h(f_A(X_A, X_C), f_B(X_B, X_C))$, under variable partition $X = \{X_A|X_B|X_C\}$ wherein fewer number of variables are required in each sub-set X_A, X_B and X_C.

Decomposition of Boolean functions has been extensively studied, and initial work can be traced back to 1950s. Traditional approaches, e.g. [7], use BDDs as the underlying data structure. However, BDDs impose severe constraints on the number of input variables circuits can have. It is also generally accepted that BDDs do not scale for large Boolean functions. As a result, recent work [3], [6] proposed the use of Boolean Satisfiability (SAT) and Minimally Unsatisfiable Subformulas (MUS) to manipulate large Boolean functions. This resulted in significant performance improvements. In addition, [3], [6] proposed heuristic approaches for identifying variable partitions. Explicit (but heuristically restricted) enumeration of variable partitions [6] sometimes produces good solutions, corresponding to adequate values of disjointness and balancedness [3], [6]. However, it is in general difficult to guarantee the quality of variable partitions, since the number of possible partitions grows exponentially with the number of inputs. This prevents brute-force search [6] in practice.

The quality of Boolean function decomposition is often related with the quality of variable partitions [3], [4], [6], as an optimal solution requires fewer input variables and simpler sub-functions.

This paper addresses the problem of computing bi-decompositions with optimum variable partitions. The optimality of achieved variable partitions is measured in terms of existing metrics, namely disjointness and balancedness. The proposed solutions are based on novel QBF formulations for the problem of Boolean function bi-decomposition subject to target metrics (e.g. disjointness, balancedness, etc.). Besides the novel QBF formulations, the paper shows how bi-decomposition can be computed with *optimum* values for the target metrics. Experimental results, obtained on well-known benchmarks, demonstrate that QBF-based function bi-decomposition performs comparably with recent heuristic approaches [3], [6], while guaranteeing optimum variable partitions.

The paper is organized as follows. Section II covers the preliminaries. Section III reviews models for Boolean function bi-decomposition. Section IV proposes the new QBF-based models. Section V presents the experimental results. Finally, section VI concludes the paper and outlines future work. More details are in the extended version [2] of this paper.

II. Preliminaries

Variables are represented by set $X = \{x_1, x_2, \ldots, x_n\}$. The cardinality of X is denoted as $||X||$. A partition of a set X into $X_i \subseteq X$ for $i = 1, \ldots, k$ (with $X_i \bigcap X_j = \emptyset, i \neq j$ and $\bigcup_i X_i = X$) is denoted by $\{X_1|X_2|\ldots|X_k\}$. A Completely Specified Function (CSF) is denoted by $f : \mathcal{B}^n \rightarrow \mathcal{B}$. Similar to the recent work [3], [6], this paper assumes CSFs.

A. Boolean Function Bi-Decomposition

Definition 1: Bi-decomposition [8] for Completely Specified Function (CSF) $f(X)$ consists of decomposing $f(X)$ under variable partition $X = \{X_A|X_B|X_C\}$, into the form of $f(X) = f_A(X_A, X_C)$ <OP> $f_B(X_B, X_C)$, where <OP> is a binary operator, typically OR, AND *or* XOR.

This paper addresses *OR*, *AND* and *XOR* bi-decomposition because these three basic gates form other types of bi-decomposition [6]. Bi-decomposition is termed *disjoint* if $||X_C|| = 0$. A partition of X is trivial if $X = X_A \bigcup X_C$ or $X = X_B \bigcup X_C$ holds. Similar to earlier work [3], [6], this paper addresses non-trivial bi-decompositions.

B. Quantified Boolean Formulas

Quantified Boolean Formulas (QBF) generalize Boolean formulas by quantifying variables, either existentially ($\exists$) or universally ($\forall$). Throughout this paper, all Boolean variables in QBF are assumed to be quantified (or bound). QBF are assumed to be in the *prenex* form $Q_1 p_1 \ldots Q_n p_n . \phi$, with $Q_i \in \{\exists, \forall\}$, p_i are distinct Boolean variables, and ϕ is a propositional formula using only the variables p_i and the constants 0 (false), 1 (true).

C. Quality Metrics

The quality of variable partitions mainly impacts the quality of bi-decomposition [3], [4], [6], and indirectly impacts the decomposed network, e.g. delay, area and power consumption [4]. Similar to [3], [6], this paper measures the quality of variable partitions through two *relative* quality metrics, namely *disjointness* and *balancedness*. This paper develops QBF models for achieving disjointness and balancedness targets, or even the *optimum* solutions.

III. Related Work

Bi-Decompositions of Boolean functions are either based on BDDs or on SAT. A brief review of BDD-based approaches is given in [2]. This section briefly overviews earlier work on SAT-based bi-decomposition.

A. SAT-Based and MUS-Based Bi-Decomposition

With the objective of targeting bi-decomposition of large Boolean functions with a large number of inputs, and correspondingly large number of variable partitions, recent work proposed SAT-based bi-decomposition [6] and MUS-based bi-decomposition [3], [6]. SAT-based OR, AND and XOR bi-decompositions under known and unknown partition of variables were proposed in [6]. For example, the widely used OR bi-decomposition can be computed by SAT solving [6]. Given a non-trivial variable partition $X = \{X_A|X_B|X_C\}$, the following result holds:

Proposition 1: [6] A completely specified function $f(X)$ can be written as $f_A(X_A, X_C) \vee f_B(X_B, X_C)$ for some functions f_A and f_B if and only if the Boolean formula

$$f(X_A, X_B, X_C) \wedge \neg f(X'_A, X_B, X_C) \wedge \neg f(X_A, X'_B, X_C) \tag{1}$$

is unsatisfiable, where variable set Y' is an instantiated version of variable set Y.

An *instantiated* version x' of Boolean variable x can be viewed as a new Boolean variable x' that replaces x. This approach assumes that a variable partition $X = \{X_A|X_B|X_C\}$ is given. In practice, such variable partitions are generally unknown and must be automatically derived. One possible approach is to consider the following formulation [6]:

$$f(X) \wedge \neg f(X') \wedge \bigwedge_i ((x_i \equiv x'_i) \vee \alpha_{x_i}) \wedge \neg f(X'') \wedge \bigwedge_i ((x_i \equiv x''_i) \vee \beta_{x_i}) \tag{2}$$

where $x' \in X'$ and $x'' \in X''$ are instantiated versions of $x \in X$. α_{x_i} and β_{x_i} are *control* variables for enumerating variable partitions. By assigning different Boolean values to α_{x_i} and β_{x_i}, some of the clauses $((x_i \equiv x'_i) \vee \alpha_{x_i})$, $((x_i \equiv x''_i) \vee \beta_{x_i})$ are relaxed. The resulting clauses $(x_i \equiv x'_i)$ and $(x_i \equiv x''_i)$ impose equivalence relations for each pair of variables in sets X and X', and in X and X'', respectively.

The original work on SAT-based bi-decomposition [6] proposed the use of interpolation for computing the target functions f_A and f_B. Given that our work focuses on improving the identification of variable partitions, interpolation can also be used for computing functions f_A and f_B. Similarly to OR bi-decomposition, AND and XOR bi-decomposition can be computed by using SAT. Due to space limitations, this section omits the explanation of SAT-based AND and XOR bi-decompositions (e.g. see [6]). The approaches proposed in [6] are referred to as ***LJH*** in the remainder of the paper.

SAT-based bi-decomposition [3], [6] proposed a number of MUS-based techniques for computing good variables partitions. These include plain MUS computation and, more recently, group-oriented MUS computation. These approaches can be viewed as practical engineering solutions for bi-decomposition of *large* Boolean functions. Nevertheless, these approaches are heuristic and provide no guarantees regarding the quality of computed variable partitions.

IV. QBF-Based Bi-Decomposition

This section develops QBF models for computing Boolean function bi-decomposition with optimum variable partitions. The case for OR bi-decomposition is considered. The extended version [2] summarizes AND and XOR bi-decomposition.

Observe that, as described above, assignments to the α_{x_i} and β_{x_i} variables specify the sets X_A, X_B and X_C. A key observation is that formulation (2) above provides the basis for a natural (albeit incomplete) QBF formulation. Formulation (2) essentially quantifies existentially the α_{x_i} and β_{x_i} variables and, by requiring unsatisfiability, quantifies the X, X', X'' variables universally. This results in the following QBF formulation:

$$\begin{aligned} \exists_{\alpha_{x_i}, \beta_{x_i}}, \forall_{X, X', X''}. \neg[f(X) \wedge \neg f(X') \wedge \bigwedge_i ((x_i \equiv x'_i) \vee \alpha_{x_i}) \\ \wedge \neg f(X'') \wedge \bigwedge_i ((x_i \equiv x''_i) \vee \beta_{x_i})] \end{aligned} \tag{3}$$

This QBF formulation has a few important drawbacks: (i) the solution can be a trivial partition; and (ii) the quality of a non-trivial partition can be arbitrary. As a result, the ability to control the quality of the computed variable partition, requires extending (3) as follows:

$$\begin{aligned} \exists_{\alpha_{x_i}, \beta_{x_i}}, \forall_{X, X', X''}. \neg[f(X) \wedge \neg f(X') \wedge \bigwedge_i ((x_i \equiv x'_i) \vee \alpha_{x_i}) \\ \wedge \neg f(X'') \wedge \bigwedge_i ((x_i \equiv x''_i) \vee \beta_{x_i})] \wedge f_N(\alpha_X, \beta_X) \wedge f_T(\alpha_X, \beta_X) \end{aligned} \tag{4}$$

where $f_N(\alpha_X, \beta_X)$ requires a non-trivial variable partition, and $f_T(\alpha_X, \beta_X)$ requires the computed variable partition to respect target metrics, e.g. disjointness or balancedness.

1) Ensuring Non-trivial Partitions: Filtering of trivial partitions is achieved through constraints added to $f_N(\alpha_X, \beta_X)$. A *trivial partition* of X is such that either $X = X_A \bigcup X_C$ or $X = X_B \bigcup X_C$ holds. In other words, a *non-trivial* partition is such that $X_A \neq \emptyset \wedge X_B \neq \emptyset$. This condition is expressed with cardinality constraints $AtLeast1(\bigcup_{x \in X} \alpha_x) \wedge AtLeast1(\bigcup_{x \in X} \beta_x)$.

2) Targeting Disjointness: Constraints on variables α_{x_i} and β_{x_i} serve to require target values of disjointness. Observe that in model (4), the assignment $(\alpha_x, \beta_x) = (0, 0)$ denotes that $x \in X_C$. Improvements in disjointness consists of reducing the size of X_C. This is achieved by computing variable partitions with a sufficiently small number of pairs (α_x, β_x) with $(\alpha_x, \beta_x) = (0, 0)$. For a target level of disjointness ϵ, with $0 \leq \epsilon < 1$, let $k \in \mathbb{N}, k = \lfloor ||X|| \cdot \epsilon \rfloor$. Hence, the target constraint $f_T(\alpha_X, \beta_X)$ is defined as follows:

$$(\sum_{x \in X} \overline{\alpha_x} \cdot \overline{\beta_x}) \leq k \tag{5}$$

Clearly, k is discrete and finite, and so the optimum value can be computed by iteratively solving QBF (4) for different values of k.

Moreover, observe that Boolean function bi-decomposition exhibits key symmetry properties. For example, sets X_A and X_B are *indistinguishable*, and so the optimum solution is obtained even if constraint $||X_A|| \geq ||X_B||$ is included in the problem formulation. This constraint can either be added to $f_N(\alpha_X, \beta_X)$ or to $f_T(\alpha_X, \beta_X)$. In practice, this optimization reduces substantially the search space of the resulting QBF.

3) Targeting Balancedness: Similarly to the approach for disjointness, constraints on variables α_{x_i} and β_{x_i} serve to require target values of balancedness. Balancedness is improved if the difference between the number of variables x in set X_A, i.e. $(\alpha_x, \beta_x) = (1, 0)$, and the number of variables x in set X_B, i.e. $(\alpha_x, \beta_x) = (0, 1)$ is *minimized.* For a target level of balancedness ϵ, with $0 \leq \epsilon < 1$, let $k \in \mathbb{N}, k = \lfloor ||X|| \cdot \epsilon \rfloor$. Hence, the target constraint $f_T(\alpha_X, \beta_X)$ is defined as follows:

$$0 \leq (\sum_{x \in X} \alpha_x \cdot \overline{\beta_x} - \sum_{x \in X} \overline{\alpha_x} \cdot \beta_x) \leq k \tag{6}$$

Observe that, as before, the optimum value of k can be searched for, by iteratively solving the QBF (4) for different values of k. In addition, note that, in this case, the symmetry between X_A and X_B is automatically removed, by requiring $||X_A|| \geq ||X_B||$.

4) Integrating Disjointness & Balancedness: In practical settings, it is often the case that the objective is to achieve some simultaneous level of disjointness and balancedness.

Definition 2 (Cost of Disjointness and Balancedness): The cost of Disjointness and Balancedness is the arithmetic sum of weighted Disjointness and weighted Balancedness, which is expressed as the following cost function:

$$\sum \varpi_D \cdot Disjointness + \varpi_B \cdot Balancedness \tag{7}$$

where ϖ_D ($\varpi_D \in [0, 1]$) and ϖ_B ($\varpi_B \in [0, 1]$) are weights for Disjointness and Balancedness, respectively.

Observe that cost function (7) can be simplified if disjointness and balancedness are equally preferred, i.e. $\varpi_D = \varpi_B = 1$. Moreover, if $||X_A||$ is assumed to be no less than $||X_B||$, then the cardinality constraint can be simplified as follows.

$$0 \leq (\sum_{x \in X} \overline{\alpha_x} \cdot \overline{\beta_x} + \sum_{x \in X} \alpha_x \cdot \overline{\beta_x} - \sum_{x \in X} \overline{\alpha_x} \cdot \beta_x) \leq k \tag{8}$$

where $k \in \mathbb{N}, k = \lfloor ||X|| \cdot \epsilon \rfloor$.

5) Practical Implementation: In practice, the use of the 2QBF formula (4) is not straightforward because it requires auxiliary variables to encode it into CNF, as required by most QBF solvers. These auxiliary variables are existentially quantified in the innermost level of the QBF prefix and consequently result in a 3QCNF formula.

Consider a 2QBF formula $\exists_Z, \forall_X.\phi$, where ϕ is not in CNF. As indicated above, converting ϕ to CNF requires additional variables, which results in a 3QCNF formula. Instead of using a solver for QBF formulas with three levels of quantifiers, a different approach is used, which has been recently suggested in [5]. Consider the negation of $\exists_Z, \forall_X.\phi$, i.e. $\forall_Z, \exists_X.\neg\phi$. Observe that if $\exists_Z, \forall_X.\phi$ is valid, then $\forall_Z, \exists_X.\neg\phi$ cannot be valid. Thus, if the QBF solver provides a *counterexample* for why $\forall_Z, \exists_X.\neg\phi$ cannot be satisfied, it represents a model of $\exists_Z, \forall_X.\phi$. For QBF (4), the model represents the intended variable partition. As a result, the 2QBF formula to be used becomes:

$$\begin{aligned} &\forall_{\alpha_{x_i}, \beta_{x_i}}, \exists_{X, X', X''}.[f(X) \wedge \neg f(X') \wedge \bigwedge_i ((x_i \equiv x'_i) \vee \alpha_{x_i}) \\ &\wedge \neg f(X'') \wedge \bigwedge_i ((x_i \equiv x''_i) \vee \beta_{x_i})] \vee \neg f_N(\alpha_X, \beta_X) \vee \neg f_T(\alpha_X, \beta_X) \end{aligned} \tag{9}$$

6) Finding the Optimum: This section summarizes the approaches that can be used for computing the optimum disjointness or balancedness. An initial upper bound, on both disjointness and balancedness, can be obtained with the group-oriented MUS-based model [3]. Alternatively, the upper bound can be set to 1. Three strategies have been studied for computing the optimum disjointness and balancedness values.

Monotonically Increasing (MI) denotes iteratively increasing the value of k. Monotonically Decreasing (MD) denotes iteratively decreasing the value of k. Finally, dichotomic divide-and-conquer denotes binary search (Bin). In our experiments, the best results for disjointness were obtained using the sequence: MD $\rightarrow$ Bin $\rightarrow$ MI, where the number of iterations for each is heuristically chosen. For balancedness the best results for balancedness were obtained with MI.

V. Experimental Results

The tool, ***STEP** — **S**atisfiability-based func**T**ion d**E**com**P**osition*, implements the techniques proposed by this paper. ***STEP*** is implemented in C++, and uses ABC [1] for underlying circuit manipulation. The off-the-shelf 2QBF solver AReQS [5] and MUS solver MUSer [2] were used for QBF solving and MUS-based pre-processing of QBF searching bounds, respectively. The proposed QBF models for targeting solely *disjointness*, solely *balancedness*, and integrated disjointness and balancedness (with cost function '1$*$*disjointness* + 1$*$*balancedness*') were used for computing the *optimum* solutions; these are termed ***STEP-QD***, ***STEP-QB*** and ***STEP-QDB***, respectively. The tool ***Bi-dec*** implements OR bi-decomposition of ***LJH*** model [1] [6]. ***STEP-MG*** represents group-oriented MUS-based bi-decomposition [3].

Given a circuit, each Boolean function of Primary Output (PO) is decomposed into smaller sub-functions using the proposed and the earlier models. This section compares experimental results on the quality and performance of Boolean function bi-decomposition between different tools, namely ***Bi-dec*** (with its best quality mode, using command *'bi_dec [circuit.blif] or 0 1'*), ***STEP-MG*** (fastest mode of ***STEP***) and ***STEP-*{*QD,QB,QDB*}**. The experiments were performed on a Linux server with an Intel Xeon X3470 2.93GHz processor and 6GB RAM. Experimental data were obtained on industrial benchmarks ISCAS'85, ISCAS'89, ITC'99 and LGSynth. Circuits with *zero* decomposable PO functions were removed from the tables of results. For each circuit, the total timeout

[1]Unfortunately, AND and XOR bi-decompositions of LJH model is unavailable in tool *Bi-dec*. No result of LJH AND, XOR could be shown here.

TABLE I
COMPARISON OF QUALITY METRICS BETWEEN OR MODELS

Circuit	Circuit Statistics			$\mathcal{OR}$ LJH [6] vs. STEP-{QD,QB,QDB}					
				Disjointness		Balancedness		Disjointss+Balancedness	
	#In	#InM	#Out	STEP-QD better (%)	Both two are equal (%)	STEP-QB better (%)	Both two are equal (%)	STEP-QDB better (%)	Both two are equal (%)
C7552	207	194	108	30.00	70.00	50.00	50.00	0.00	100.00
s15850.1	611	183	684	0.00	100.00	7.69	92.31	7.69	92.31
s38584.1	1464	147	1730	18.60	81.40	70.15	29.85	41.76	58.24
C2670	233	119	140	8.33	91.67	48.72	51.28	11.54	88.46
i10	257	108	224	17.57	82.43	73.23	26.77	18.92	81.08
s38417	1664	99	1742	12.28	87.72	52.65	47.35	6.45	93.55
s9234.1	247	83	250	11.58	88.42	60.78	39.22	8.11	91.89
rot	135	63	107	4.17	95.83	72.92	27.08	8.33	91.67
s5378	199	60	213	10.38	89.62	82.24	17.76	5.00	95.00
s1423	91	59	79	3.85	96.15	42.31	57.69	5.00	95.00
pair	173	53	137	26.60	73.40	82.46	17.54	38.10	61.90
C880	60	45	26	33.33	66.67	81.25	18.75	0.00	100.00
clma	415	42	115	0.00	100.00	37.50	62.50	0.00	100.00
ITC_b07	49	42	57	7.69	92.31	84.62	15.38	0.00	100.00
ITC_b12	125	37	127	0.00	100.00	12.66	87.34	0.00	100.00
sbc	68	35	84	17.65	82.35	88.24	11.76	21.05	78.95
mm9a	39	31	36	30.00	70.00	38.10	61.90	0.00	100.00
mm9b	38	31	35	11.11	88.89	42.11	57.89	0.00	100.00

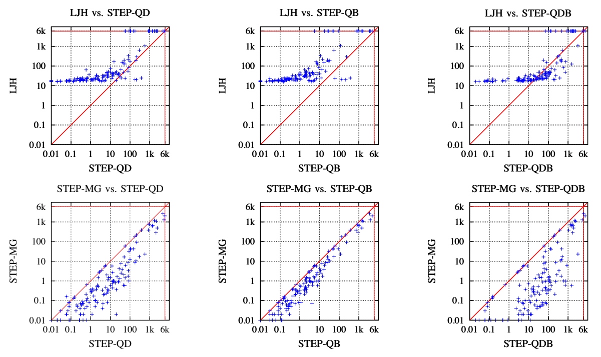

Fig. 1. CPU time comparison between models for all 145 circuits

was set to 6000 seconds. Each run of the QBF solver was given a timeout of 4 seconds. Due to space restrictions, only representative experimental results (for the *large* benchmarks, with **#InM** > 30) are shown.

A. Quality of Variable Partitions

The quality of variable partitions are essential to function bi-decomposition and determine the overall quality [2]–[4], [6]. ***STEP-*{*QD,QB,QDB*}** can guarantee the quality of variable partitions. For example, the new QBF models allow for controllable disjoint, balanced and customized, i.e. with user-specified cost functions, bi-decompositions. Similar to [3], [6], *disjointness* and *balancedness* were used to validate the quality of the results obtained with the new QBF models.

Table I shows the results of quality metrics between models for OR bi-decomposition. Columns **#In**, **#InM** and **#Out** denote the number of primary inputs, maximum number of support variables in POs, PO functions (to be decomposed), respectively. ***STEP-*{*QD,QB,QDB*}** is bootstrapped with the result of ***STEP-MG***. Hence, ***STEP-*{*QD,QB,QDB*}** cannot yield metrics worse than ***STEP-MG***. Moreover, the obtained experimental results of ***STEP-*{*QD,QB,QDB*}** for any PO, even if unable to prove the optimum, were no worse than ***Bi-dec***. As can be observed, ***STEP-QD***, ***STEP-QB***, and ***STEP-QDB*** are in many cases capable of improving the metrics computed by the other two tools. As can be concluded, the proposed QBF models were able to bi-decompose Boolean functions whenever possible, whereas earlier models fail to achieve the best decompositions in many cases.

TABLE II
PERCENTAGE OF SOLVED POS WITH STEP FOR OR BI-DECOMPOSITION

#Out	STEP-QD (%)	STEP-QB (%)	STEP-QDB (%)
38582	91.97	97.81	84.42

B. Practical Performance of QBF Models

Performance is also significant to function bi-decomposition as logic synthesis involves several iterations of function decompositions [3], [4]. Figure 1 shows scatter plots comparing the run times of ***STEP-*{*QD,QB,QDB*}** against the other tools for *all* 145 circuits. As can be observed, ***STEP-*{*QD,QB,QDB*}** outperforms ***Bi-dec***, but performs worse than ***STEP-MG***. Nevertheless, it is important to point out that both ***Bi-dec*** and ***STEP-MG*** compute *approximate* solutions, whereas ***STEP-*{*QD,QB,QDB*}** computes *exact* solutions. Table II shows the percentage of instances solved by ***STEP-*{*QD,QB,QDB*}**. As can be observed, ***STEP-QD*** solves close to 92% of the POs, ***STEP-QB*** solves close to 98% of the POs, and ***STEP-QDB*** solves close to 85% of the POs. These results are promising, given the current pace of improvement of QBF.

VI. CONCLUSIONS

Boolean function decomposition is ubiquitous in logic synthesis. This paper addresses Boolean function bi-decomposition and develops novel QBF models for finding *optimum* bi-decompositions according to the well-established metrics, namely disjointness and balancedness. In addition, the paper describes techniques for improving the models and, consequently, for QBF solving. A key example is breaking the symmetry between sets of variables in the computed bi-decomposition. Experimental results obtained on representative benchmark circuits, demonstrate that the new QBF models can be solved efficiently with modern 2QBF solvers [5], and perform comparably with state-of-the-art heuristic solutions for Boolean function bi-decomposition [3], [6].

Future work will address performance improvements, through tight integration of the new QBF models with heuristic SAT-based approaches.

Acknowledgement We would like to thank Prof. Jie-Hong Roland Jiang for kindly providing the tool *Bi-dec*. This work is partially supported by SFI PI grant BEACON (09/IN.1/I2618).

REFERENCES

[1] "Berkeley Logic Synthesis and Verification Group. ABC: A System for Sequential Synthesis and Verification, Release 70930," in *http://www.eecs.berkeley.edu/∼alanmi/abc/*.
[2] H. Chen, M. Janota, and J. Marques-Silva, "QBF-Based Boolean Function Bi-Decomposition." Computing Research Repository (CoRR), abs/1112.2313, December 2011.
[3] H. Chen and J. Marques-Silva, "Improvements to Satisfiability-Based Boolean Function Bi-Decomposition," in *VLSI-SoC*, 2011, pp. 142–147.
[4] M. Choudhury and K. Mohanram, "Bi-decomposition of large Boolean functions using blocking edge graphs," in *ICCAD*, 2010, pp. 586–591.
[5] M. Janota and J. Marques-Silva, "Abstraction-Based Algorithm for 2QBF," in *SAT*, 2011, pp. 230–244.
[6] R.-R. Lee, J.-H. Jiang, and W.-L. Hung, "Bi-decomposing large Boolean functions via interpolation and satisfiability solving," in *DAC*, 2008, pp. 636–641.
[7] A. Mishchenko, B. Steinbach, and M. Perkowski, "An algorithm for bi-decomposition of logic functions," in *DAC*, 2001, pp. 103–108.
[8] T. Sasao and J. T. Butler, "on bi-decomposition of logic functions," in *IWLS*, 1997, pp. 1–6.