New & Improved Models for SAT-Based Bi-Decomposition

Huan Chen
Complex & Adaptive Systems Laboratory
School of Computer Science and Informatics
University College Dublin, Ireland
huan.chen@ucd.ie

Joao Marques-Silva
Complex & Adaptive Systems Laboratory
School of Computer Science and Informatics
University College Dublin, Ireland
jpms@ucd.ie

ABSTRACT

Boolean function bi-decomposition is pervasive in logic synthesis. Bi-decomposition entails the decomposition of a Boolean function into two other simpler functions connected by a simple two-input gate. Existing solutions are based either on Binary Decision Diagrams (BDDs) or Boolean Satisfiability (SAT). Furthermore, the partition of the input set of variables is either assumed, or an automatic derivation is required. Most recent work on bi-decomposition proposed the use of Minimally Unsatisfiable Subformulas (MUSes) or Quantified Boolean Formulas (QBF) for computing, respectively, variable partitions of either approximate or optimum quality. This paper develops new group-oriented MUS-based models for addressing both the performance and the quality of bi-decompositions. The paper shows that approximate MUS search can be guided by the quality of well-known metrics. In addition, the paper improves on recent high-performance approximate models and versatile exact models, to address the practical requirements of bi-decomposition in logic synthesis. Experimental results obtained on representative benchmarks demonstrate significant improvement in performance as well as in the quality of decompositions.

Categories and Subject Descriptors

B6.3 [**Logic Design**]: Design Aids—*automatic synthesis*

General Terms

Algorithms, Design

Keywords

bi-decomposition, logic synthesis, satisfiability

1. INTRODUCTION

Boolean function bi-decomposition is ubiquitous in logic synthesis. It is arguably the most widely used form of functional decomposition. Bi-decomposition [5–7, 12, 14, 15] 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 $X = \{X_A|X_B|X_C\}$, where $f_A(X_A, X_C)$ and $f_B(X_B, X_C)$ are functions simpler than $f(X)$. In practice variable partitions are often not available, and so automatic derivation of variable partitions is required.

The quality of bi-decomposition is mainly determined by the quality of variable partitions, as an optimal solution results in simpler sub-functions f_A and f_B. Typically, two *relative* quality metrics [5,6,12,13], namely *disjointness* and *balancedness*, are used to evaluate the resulting variable partitions, for which smaller values represent preferred bi-decompositions. In practice, disjointness is in general preferred [5, 12], since it represents the reduction of common variables to f_A and f_B, which in turn often simplifies the resulting Boolean function. Similar to recent work on functional decomposition [5,6,12,13], this paper addresses these two relative metrics, namely disjointness and balancedness. *Absolute* quality metrics are an alternative to relative quality metrics, and include total variable count (Σ) and maximum partition size (Δ) [7]. Nevertheless, absolute quality metrics scale worse with the number of inputs [7].

Research on decomposition of Boolean functions can be traced back to 1950s [2, 8]. Traditional approaches [4, 11, 15, 16] are based on BDDs for bi-decomposing Boolean networks. However, BDDs impose severe constraints on the number of input variables functions can have. Also, approaches based on BDD-based bi-decomposition lack mechanisms for deriving variable partitions; hence the partition of the input set of variables needs to be assumed. Moreover, it is also generally accepted that BDDs do not scale for large Boolean functions. As a result, recent work [5,6,12,13] proposed SAT-based models for addressing performance and quality in decompositions.

SAT-based bi-decomposition has a number of advantages and also disadvantages. In terms of the quality of computed partitions, SAT-based models can either be *approximate* or *exact*. Approximate solutions [5, 12] proposed Boolean Satisfiability (SAT) and Minimally Unsatisfiable Subformulas (MUS) for manipulating Boolean functions. These approaches achieve significant performance gains. However, the approximate models find solutions with brute-force search [12], heuristic shuffling of CNF IDs [12], heuristic searching of seed variable partitions [5] and interfacing of standalone MUS solvers [5]. These solutions prevent *controllable* quality in practice. In contrast, exact models [6] use QBF for manipulating Boolean functions and constraints. This re-

GLSVLSI'12, May 3–4, 2012, Salt Lake City, Utah, USA.

sults in guaranteed quality of bi-decompositions. However, solving the QBF formulas representing the exact models can have significant computational cost.

This paper develops extensions to earlier work on approximate [5,12] and exact [6] SAT-based bi-decomposition, and proposes a new heuristic model. The paper has three main contributions. First, the paper develops a new approximate Group-oriented MUS-based model that in practice yields good quality metrics. This new model offers significant performance improvement over recently proposed exact solutions [6] as well as to the majority of approximate solutions [5, 12]. Moreover, this new model yields bi-decompositions of quality better than existing approximate models [5, 12]. The second contribution addresses techniques for achieving target quality metrics with exact bi-decomposition models [6]. The third contribution proposes a flow for aggregating all existing bi-decomposition models [5, 6, 12], as well as the new models, with the purpose of achieving a wide range of trade-offs between quality of decomposition metrics and runtime performance.

The paper is organized as follows. Section 2 provides the preliminaries. Section 3 reviews Satisfiability-based models for bi-decomposition. Section 4 proposes the new models and the practical extensions. Section 5 presents the experimental results. Finally, section 6 concludes the paper and outlines future work.

2. 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 [5,6,12], this paper assumes CSFs.

2.1 Boolean Function Bi-Decomposition

DEFINITION 1. *Bi-decomposition [15] 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 types form other types of bi-decomposition [12]. 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 [5, 6, 12, 13], this paper addresses *non-trivial* bi-decompositions.

2.2 Boolean Satisfiability

A formula in Conjunctive Normal Form (CNF) $\mathcal{F}$ is defined as a set of sets of literals defined on X, representing a conjunction of disjunctions of literals. A literal is either a variable or its complement. Each set of literals is referred to as a clause c. Moreover, it is assumed that each clause is non-tautological. Additional SAT definitions can be found in standard references (e.g. [3]).

DEFINITION 2 (MUS). *[5] $\mathcal{M} \subseteq \mathcal{F}$ is a Minimally Unsatisfiable Subformula (MUS) iff $\mathcal{M}$ is unsatisfiable and $\forall_{c \in \mathcal{M}}$, $\mathcal{M} \setminus \{c\}$ is satisfiable.*

DEFINITION 3 (GROUP-ORIENTED MUS). *(E.g. [5]) Given an unsatisfiable CNF formula $\mathcal{C} = \mathcal{D} \cup \bigcup_{G \in \mathcal{G}} G$, where $\mathcal{G} = \{\mathcal{G}_1, \ldots, \mathcal{G}_k\}$, and $\mathcal{D}$ and each $\mathcal{G}_i$ are disjoint sets of clauses, a group-oriented MUS of $\mathcal{C}$ is a subset $\mathcal{G}'$ of $\mathcal{G}$ such that $\mathcal{D} \cup \bigcup_{G \in \mathcal{G}'} G$ is unsatisfiable and, for every $\mathcal{G}'' \subset \mathcal{G}'$, we have that $\mathcal{D} \cup \bigcup_{G \in \mathcal{G}''} G$ is satisfiable.*

2.3 Quality Metrics

The quality of variable partitions mainly impacts the quality of bi-decomposition [5–7, 12], and indirectly impacts the decomposed network, e.g. delay, area and power consumption [7]. Similar to [5, 6, 12, 13], this paper measures the quality of variable partitions through two *relative* quality metrics, namely *disjointness* and *balancedness*. Assume a variable partition $\{X_A|X_B|X_C\}$ for $f(X)$, where X_A, X_B and X_C are the sets of the input variables to decomposition functions f_A, f_B and common to f_A and f_B, respectively.

DEFINITION 4 (DISJOINTNESS). $\epsilon_D = \frac{||X_C||}{||X||}$ *denotes the ratio of the number of common variables to inputs. A value of ϵ_D close to 0 is preferred, as $\epsilon_D = 0$ represents a disjoint bi-decomposition.*

DEFINITION 5 (BALANCEDNESS). $\epsilon_B = \frac{\left|||X_A|| - ||X_B||\right|}{||X||}$ *denotes the absolute size difference between X_A and X_B. $\epsilon_B = 0$ represents a balanced variable partition.*

In practice, disjointness is preferred since a lower value represents a smaller number of shared input variables of the resulting decomposed circuit that typically has smaller area and power footprint. A lower balancedness typically corresponds to smaller delay of the decomposed network.

3. SATISFIABILITY-BASED MODELS

Traditional bi-decomposition algorithms [4, 11, 15, 16] are based on BDDs. This can have significant impact on different aspects of logic synthesis (see [5] for a review). Recent work on bi-decomposition mainly focuses on the performance of models when decomposing large functions [5, 12] as well as the quality of the bi-decompositions [6]. This section briefly reviews Satisfiability-based models in accordance with the precision of targeted quality, namely approximate and exact.

3.1 Approximate Models

The primary objective of using approximate models is to achieve good performance with approximate quality.

SAT-Based and MUS-Based Models. The approximate models are either SAT-based [12] or MUS-based [5]. These models provide a collection of solutions to the OR, AND and XOR bi-decompositions under known and unknown partition of variables. In practice, the partitions of variables are often unknown, and therefore, an automatic derivation of partitions is required.

A distinct feature provided by SAT-based models is that they are capable of automatically deriving variable partitions. The main difference between these SAT-based approaches is in the underlying representation of the constraints for modeling the bi-decomposition, including SAT [12], plain-MUS [5] and group-oriented-MUS [5]. The performance of these models is determined by (1) the modeling of constraints, and (2) how the constraints are solved. For example, the group-oriented MUS model [5] exhibits remarkably good performance when partitioning input variables for bi-decompositions. Consider the explicit groups of clauses for

the modeling of bi-decomposition [5]:

$$\begin{aligned} \mathcal{D} &= \{f(X) \wedge \neg f(X') \wedge \neg f(X'')\} \\ \mathcal{G}_{i_a} &= \{(x_i \equiv x_i')\}, \mathcal{G}_{i_b} = \{(x_i \equiv x_i'')\} \end{aligned} \tag{1}$$

PROPOSITION 1. [5] A completely specified function $f(X)$ can be decomposed into $f_A(X_A, X_C) \vee f_B(X_B, X_C)$ for some functions f_A and f_B if and only if the Boolean formula of the set of clauses $\mathcal{C}$, with

$$\mathcal{C} = \mathcal{D} \cup \mathcal{G}_{\mathcal{A}} \cup \mathcal{G}_{\mathcal{B}} \tag{2}$$

is *unsatisfiable* under a non-trivial partition, where the sub-set $\mathcal{G}_{\mathcal{A}} \subset \{\bigcup_i \mathcal{G}_{i_a}\}$, the sub-set $\mathcal{G}_{\mathcal{B}} \subset \{\bigcup_i \mathcal{G}_{i_b}\}$.

Notice that $\mathcal{D}$ contains clauses for representing the target Boolean function and its two complements. However, $\mathcal{D}$ does not contribute to the size of a group-oriented MUS, and can hence be viewed as *don't care* conditions w.r.t the size of the modeling. This in part motivates the gain of performance and this concept is adapted for the new models proposed in this paper.

The partitioning of variables occupies most of the run time in bi-decomposition [5–7, 12]. Essentially, the derivation of partitions is the process of switching the input variables between the two partitions. Switching of variables can be captured by selecting the groups of the input variables.

3.2 Exact Models

Exact models [6] provide controllable quality of bi-decomposition for various optimizations in logic synthesis. Existing exact solutions are based on QBF [6], and address the problem of computing bi-decompositions with optimum variable partitions. The optimality of achieved variable partitions is measured in terms of the existing metrics, namely disjointness and balancedness. Besides the novel QBF formulation, [6] shows how bi-decompositions can be computed with *optimum* values for target metrics. These QBF-based models guarantee the optimum quality of variable partitions. Nevertheless, exact models incur a performance penalty when compared with approximate models, in particular, with group-oriented MUS-based models [5].

4. NEW MODELS AND EXTENSIONS

Algorithm 1 proposes Quality-Guided-Group-MUS that targets high quality and efficient bi-decomposition.

4.1 OR Bi-Decomposition

Group-oriented MUS model [5] shows remarkably good performance, and it is used as the underlying construction in Algorithm 1. Similar to [5], during MUS search, the group $\mathcal{D}$ of clauses remains unchanged and only the groups $\bigcup_i \mathcal{G}_{i_A}$ and $\bigcup_i \mathcal{G}_{i_B}$ of clauses for variable partitions are considered:

$$\mathcal{F} = \mathcal{D} \bigcup_i \underbrace{\{(\overbrace{(x_i \equiv x_i')}^{\mathcal{G}_{i_A}} \wedge \overbrace{(x_i \equiv x_i'')}^{\mathcal{G}_{i_B}})\}}_{\mathcal{G}_{i_C}} \tag{3}$$

Observe that the subset $\mathcal{C}' = \mathcal{D} \cup \bigcup_i \mathcal{G}'_{i_a} \cup \bigcup_i \mathcal{G}'_{i_b}$ computed from (3) indicates the variable partitions. $\mathcal{G}'_{i_a}$ and $\mathcal{G}'_{i_b}$ are defined as follows: $((\mathcal{G}'_{i_a} \equiv \mathcal{G}_{i_a}), (\mathcal{G}'_{i_b} \equiv \mathcal{G}_{i_b})) = (1,1)$, $(1,0)$, $(0,1)$, and $(0,0)$ denote $x_i \in X_C$, $x_i \in X_B$, $x_i \in X_A$ and x_i can either be in X_A or X_B, respectively.

Algorithm 1: Quality-Guided-Group-MUS ($\mathcal{F}$,X)

```
   Input: F — UNSAT formula of a seed variable partition
   Input: X — input variables of Boolean function f(X)
   Output: M — MUS with approximate good quality
   Data: i ← 2 — do not involve the two seed variables
1  for i ≤ ||X|| do
2      if DIFF(F) ≥ 0 then
3          F' ← INCREASE_X_B_DECREASE_X_C(&F, i)
4          if SAT_SOLVE(F') == UNSAT then
5              F ← F'
6          else
7              F' ← INCREASE_X_A_DECREASE_X_C(&F, i)
8              if SAT_SOLVE(F') == UNSAT then
9                  F ← F'
10             end
11         end
12     else
13         ...
14     end
15     i ← i + 1
16 end
17 M ← F
```

4.1.1 Approximating Optimum Balancedness

The refinement of a quality metric initially starts from an UNSAT formula of a seed variable partition [12], where the set X_A and the set X_B each takes at least one variable. The improvement of balancedness essentially corresponds to reducing the size difference between X_A and X_B:

$$DIFF_{AB} = ||X_A|| - ||X_B|| \tag{4}$$

If (4) results in a non-negative value (line 2), then the size (or cardinality) of set X_B is increased whereas the size of X_C is thereby reduced (line 3). This behaviour is captured by deleting the clauses of one group $\mathcal{G}_{i_B}$ of (3), which produces a formula $\mathcal{F}'$ with reduced size (line 3). The resulting $\mathcal{F}'$ can be UNSAT (line 4) and then be substituted for the original $\mathcal{F}$ (line 5). As a result, the balancedness is improved by the guide of the cost function (4). In contrast, formula $\mathcal{F}'$ can be SAT which implies an unchanged $\mathcal{F}$ at the current step, since $\mathcal{F}$ is only used as a constant reference in function INCREASE_X_B_DECREASE_X_C(...) (line 3).

4.1.2 Approximating Optimum Disjointness

Observe that achieving optimum Disjointness consists of reducing the size of X_C. Such a reduction can either be done by removing the group $\mathcal{G}_{i_A}$ or the group $\mathcal{G}_{i_B}$ from formula (3). As a result, if the previous steps (line 3 to 5) failed to produce a sub-formula $\mathcal{F}'$ with the deletion of group $\mathcal{G}_{i_B}$ (line 3), the deletion of group $\mathcal{G}_{i_A}$ is thus performed (line 7) and, followed by a SAT checking (line 8). This greedy scheme of removing either group $\mathcal{G}_{i_A}$ or group $\mathcal{G}_{i_B}$ (line 2 to 14) insists on an improvement of disjointness by looking for an UNSAT subformula $\mathcal{F}'$ with reduced size. Algorithm 1 also considers the case when $DIFF_{AB}$ has negative value. This situation is similar to previous ones, and can be implemented by replicating lines 2-11 to lines 12-14, but swapping lines 3 and 7. In addition, functionally non-support inputs are identified by preprocessing and are shifted to either X_A or X_B. Thus, simultaneously removing $\mathcal{G}_{i_A}$ and $\mathcal{G}_{i_B}$ is not required.

It it important to point out that this way of greedy MUS search is novel compared to [12, 14] mainly in that the used model (3) is different from the ones with *control variables* [12] and the ones with BDD-based *variable grouping* [14]. In ad-

dition, an implicit cost function (4) is introduced to guide the greedy search. Moreover, the techniques proposed in [5] target the efficient computation of MUSes, but are unable to take quality into account during either *plain* or *group* MUS search.

4.2 AND/XOR Bi-Decomposition

AND bi-decomposition is the dual of OR bi-decomposition and can be converted from the construction of OR models [5, 12, 14]. The proposed models can decompose $\neg f$ into $f_A \vee f_B$. By negating both sides, f is decomposed into $\neg f_A \wedge \neg f_B$ [12]. The XOR bi-decomposition is similar to OR bi-decomposition [5,12] and can be explained with an analogous derivation of the OR model. The derivation of AND/XOR bi-decomposition is omitted due to lack of space.

4.3 Application-Oriented Optimization

4.3.1 Targeting Desired Quality

Practical uses of bi-decomposition [4, 11, 16] require the ability to control the quality metrics of computed variable partitions. As noted earlier, approximate models, including ***LJH*** [12] and ***STEP-M*** [5], address the quality of variable partitions, by enumeration of control variable assignments [12] or unguided searching of MUSes [5], but offer no guarantees that the results respect any quality criterion. Moreover, the enumeration of control variables grows exponentially for searching the *optimum* variable partition. Hence, approximate models are unable to guarantee quality criteria of the computed bi-decompositions. This section proposes the use of cardinality constraints in the QBF-based Models ***STEP-Q*** [6] for optimizing and controlling the quality of variable partitions.

This paper focus on two quality metrics, namely disjointness and balancedness. Different applications require distinct combinations of disjointness and balancedness. The QBF models ***STEP-Q*** allow a wide range of possible practical uses. As shown in [6], cardinality constraints defined over the control variables α_x and β_x serve to constrain the computed solutions (see [6] for the definitions of α_x and β_x). Consider small natural numbers p_D, p_B, q_A, $q_B \in \mathbb{N}$. As a result, a number of possible sets of constraints can be envisioned:

- Completely disjoint and completely balanced:
$$[\sum_{x\in X}(\overline{\alpha_x}\cdot\overline{\beta_x})=0]\wedge[\sum_{x\in X}(\alpha_x\cdot\overline{\beta_x}-\overline{\alpha_x}\cdot\beta_x)=0] \quad (5)$$
- Completely disjoint and approximately balanced:
$$[\sum_{x\in X}(\overline{\alpha_x}\cdot\overline{\beta_x})=0]\wedge[\sum_{x\in X}(\alpha_x\cdot\overline{\beta_x}-\overline{\alpha_x}\cdot\beta_x)=p_B] \quad (6)$$
- Approximately disjoint and completely balanced:
$$[\sum_{x\in X}(\overline{\alpha_x}\cdot\overline{\beta_x})=p_D]\wedge[\sum_{x\in X}(\alpha_x\cdot\overline{\beta_x}-\overline{\alpha_x}\cdot\beta_x)=0] \quad (7)$$
- Approximately disjoint and customized balancedness:
$$[\sum_{x\in X}(\overline{\alpha_x}\cdot\overline{\beta_x})=p_D]\wedge[\sum_{x\in X}(\alpha_x\cdot\overline{\beta_x})=q_A]\wedge[\sum_{x\in X}(\overline{\alpha_x}\cdot\beta_x)=q_B] \quad (8)$$

The values of p_D and p_B range from completely disjoint and balanced case to a fully customized case. Figure 1 illustrates the concepts for case (8). The additional constraints, coupled with the basic QBF model ***STEP-Q*** [6], allow fairly flexible modeling of constraints on variable partitions.

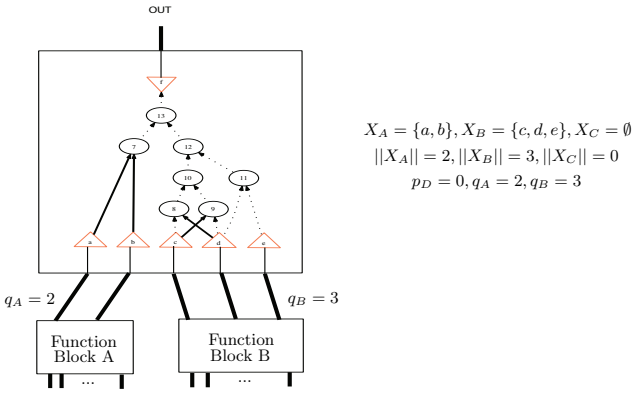

Figure 1: Example AIG (And-Inverter Graph) of the application-oriented optimization technique

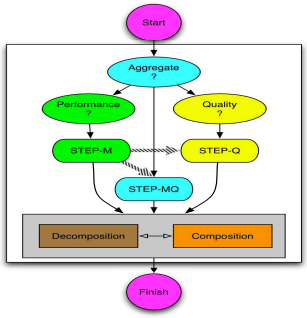

Figure 2: Optimization flow for bi-decomposition

4.3.2 Optimization Flow for Bi-Decomposition

In practice, bi-decomposition is recursively applied in logic synthesis. As stated before, different applications may require different targets for the quality metrics. This applies to both approximate and exact bi-decomposition solutions. In addition, there is often a bound on the allowed runtime. Therefore, it is reasonable to develop different models for tackling different objectives when using bi-decompositions. Figure 2 outlines the flow of bi-decomposing a Boolean function for different targets. Performance sensitive application finds ***STEP-M*** model to compute an approximate solution. Model ***STEP-Q*** directs the quality of bi-decomposition to fulfill the quality requirement of the quality sensitive applications. The new model ***STEP-MQ*** aggregates these two aspects to compute an approximate quality with a good performance. The precedent uses of ***STEP-M***, as shown by the dashed arrows in Figure 2, offer ***STEP-Q*** and ***STEP-MQ*** the preprocessing of upper bounds for QBF searching [6] and candidates for variable partitions, respectively. After obtaining of variable partitions, the original Boolean function is decomposed into decomposition functions, and which are connected by a composition function [16], by using Craig Interpolation [10, 12, 13]. The decomposition functions are required to be *functionally* shared. Moreover, the interactive search of variable partitions should be *incremental*.

5. EXPERIMENTAL RESULTS

The models from Section 4 are implemented in the Boolean function bi-decomposition tool ***STEP*** — ***S**atisfiability-based func**T**ion d**E**com**P**osition*. ***STEP*** is written in C++ (and compiled with G++ 4.4.3), it is implemented as an add-on

Table 1: Performance comparison

Circuit	Circuit Statistics			LJH [12]				STEP									
	#In	#InM	#Out	LJH-P		LJH-Q		STEP-MP [5]		STEP-MG [5]		STEP-QD [6]		STEP-QB [6]		STEP-MQ	
				#Dec	Time (s)	#Dec	Time (s)	#Dec	Time (s)	#Dec	Time (s)	#Dec	Time (s)	#Dec	Time (s)	#Dec	Time (s)
C7552	207	194	108	10	536.48	10	625.13	17	71.79	17	16.56	17	50.72	17	25.64	17	22.48
s15850.1	611	183	684	-	TO	-	TO	294	306.35	294	42.83	294	152.53	294	90.58	296	55.97
s38584.1	1464	147	1730	1065	446.23	1065	1912.06	1055	53.04	1055	23.12	1055	572.78	1055	117.25	1056	28.86
C2670	233	119	140	40	26.17	40	258.68	40	13.58	40	3.86	40	39.89	40	16.83	40	6.97
i10	257	108	224	131	407.22	131	2582.97	150	130.30	150	17.18	150	299.46	150	54.37	153	49.22
s38417	1664	99	1742	1202	5321.27	-	TO	1203	2944.47	1203	2658.25	1203	4718.92	1203	3487.92	1203	2739.31
s9234.1	247	83	250	102	55.79	102	130.43	114	18.32	114	12.23	114	100.10	114	27.50	115	8.39
rot	135	63	107	49	29.25	49	28.53	62	2.03	62	0.81	62	17.88	62	4.42	62	1.43
s5378	199	60	213	107	7.16	107	47.19	111	5.22	111	3.31	111	82.88	111	11.24	111	2.66
s1423	91	59	79	26	57.06	26	53.45	40	6.50	40	1.63	40	22.14	40	5.13	40	1.93
pair	173	53	137	117	12.84	117	84.42	114	13.89	114	10.50	114	202.11	114	33.00	114	6.74
C880	60	45	26	16	6.61	16	64.72	16	2.84	16	2.03	16	6.65	16	7.44	16	2.29
clma	415	42	115	39	1356.93	-	TO	34	1206.85	39	40.90	39	106.27	39	48.01	39	178.80
ITC_b07	49	42	57	14	18.08	14	16.38	18	3.70	18	1.47	18	2.44	18	2.07	18	1.10
ITC_b12	125	37	127	80	18.18	80	17.80	79	2.21	79	0.44	79	13.14	79	1.97	79	1.11
sbc	68	35	84	51	4.43	51	8.80	59	1.00	62	0.57	62	10.28	62	2.80	59	0.81
mm9a	39	31	36	22	10.07	22	103.38	28	8.17	28	4.16	28	28.29	28	10.20	28	4.17
mm9b	38	31	35	20	18.97	20	95.90	26	13.00	26	7.57	26	34.50	26	13.30	26	7.12

to ***ABC*** [1], and it uses ***ABC***'s circuit representation and manipulation. ***MiniSAT*** [9] is used for SAT solving.

This section evaluates the performance and quality of bi-decomposition between SAT-based models. Given a circuit, each Boolean function of Primary Output (PO) is decomposed into simpler sub-functions using the proposed models. Sequential circuits are converted into combinational circuits. In this section, ***LJH-P*** (the fastest performance mode of ***LJH*** model [12]), ***LJH-Q*** (the best quality mode of ***LJH*** model), ***STEP-MP*** (plain-MUS mode of ***STEP*** [5,6]), ***STEP-MG*** (group-oriented-MUS mode of ***STEP***), ***STEP-QD*** (QBF model of ***STEP*** for optimum Disjointness) and ***STEP-QB*** (QBF model of ***STEP*** for optimum Balancedness) are compared with the new model ***STEP-MQ***. The model ***STEP-MQ*** implements Algorithm 1.

The experiments were performed on a Linux server with an Intel Xeon X3470 2.93-GHz processor and 6GB RAM. Experimental results 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 was set to 6000 seconds. Due to space restrictions, only representative experimental results (for the *large* circuits, with **#InM > 30**) are shown. This section mainly presents the experimental results for OR bi-decomposition [1].

5.1 Performance of Models

Performance of models significantly affects the practical uses of bi-decomposition in logic synthesis, partly because bi-decomposition is recursively exploited in a number of internal loops of logic synthesis. This section evaluates the performance of the techniques proposed in this paper. Two performance metrics, CPU time and the number of decomposable functions, were used for assessing performance. Smaller CPU times indicate that decomposing a complete circuit will be faster and a larger number of decomposable functions represents an enhanced decomposability of the tool, indicating the tool is able to decompose more functions in the allowed CPU time, assuming more decomposable functions do exist.

Table 1 presents the performance data for OR bi-decomposition of the *large* circuits. The experimental data is sorted by decreasing number of maximum support variables

[1]AND and XOR bi-decomposition using the ***LJH*** model is unavailable in the *Bi-dec* tool [12].

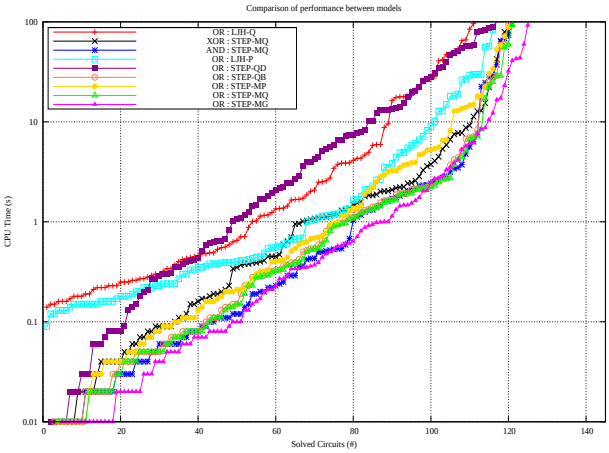

Figure 3: Performance comparison

(**#InM**). Columns **#In**, **#InM**, **#Out**, **#Dec** and **Time(s)** denote the number of primary inputs, maximum number of support variables in POs, PO functions (to be decomposed), decomposed POs and total CPU time, respectively. The results clearly show that ***STEP-MQ*** significantly outperforms ***LJH*** models and the other three ***STEP*** models, while achieving similar decomposibility.

Figure 3 shows the performance data for bi-decomposition of *small* and *medium-size* circuits. The number of solved circuits are plotted in the figure. Moreover, the *total* number of solved circuits out of *all* 145 circuits are sorted and it is showed descendingly by the legends. It is clearly that ***STEP-MQ*** solved more instances than the ***LJH*** models and the majority of ***STEP*** models. ***STEP-MQ*** performs slightly worse than ***STEP-MG*** but it is important to emphasize that ***STEP-MQ*** produces much better quality of bi-decompositions than ***STEP-MG***. In addition, ***STEP-MQ*** produced similar quality to ***LJH-Q***, but with much better performance.

5.2 Quality of Models

The quality of variable partitions is tightly related with the quality of decompositions [5–7,12,13]. This paper evaluates the quality of bi-decompositions using the same quality metrics as in [5,6,12,13], namely disjointness and balancedness.

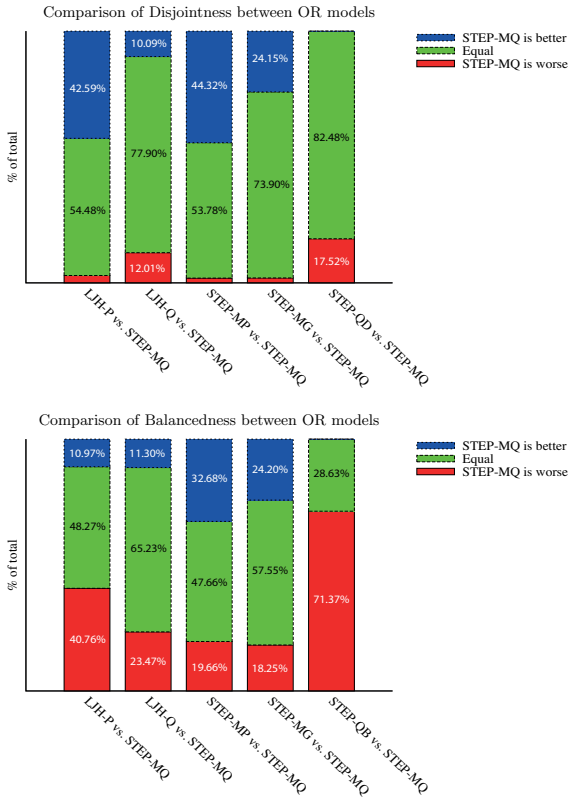

Figure 4: Quality metrics comparison

Following [5, 6, 12], disjointness is preferably considered in part because a smaller disjointness typically corresponds to optimally decomposed circuit during logic circuits [5,12]. Figure 4 summarizes the quality of variable partitions achieved in the decomposed circuits. To guarantee a fair comparison, only the functions that can be decomposed by the both two approaches are calculated. Due to space restrictions, only OR models are shown[1]. ***STEP-MQ*** computes approximate solutions. Unlike the *exact* models [6] ***STEP-{QD,QB}***, the disjointness and balancedness cannot be guaranteed by ***STEP-MQ***. ***STEP-MQ*** produces inferior balancedness compared to the SAT-based models [12], this phenomenon is because low disjointness and low balancedness are sometimes mutually exclusive [5, 12]. As can be observed, ***STEP-MQ*** achieved significantly better disjointness than the major *approximate* models, including ***LJH-P*** and ***STEP-{MP,MG}*** models.

6. CONCLUSION

This paper extends SAT-based models for bi-decomposition, and develops a new model. The relative inefficiency of the *exact* models limits their use on *very large* Boolean functions. The nature of unguided MUS and SAT searching limits the use of existing *approximate* models for *controllable* qualities. This paper develops new approximate group-oriented MUS-based models for bi-decomposition, and extends existing SAT-based models for addressing practical requirements of bi-decomposition. A key feature is that new models allow MUS searching to be guided by an implicit cost function. Comprehensive experiments with the existing and the new bi-decomposition models demonstrate that the new models achieve significant performance gains and, more importantly, also achieve visible improvement in the quality of bi-decompositions. Future work will address a tighter integration between tools, including ***STEP*** [5,6] and ***ABC*** [1], in logic synthesis, targeting area, delay and power reduction.

Acknowledgment

The authors would like to thank Prof. Jie-Hong Roland Jiang for kindly providing the SAT-based Boolean Function bi-decomposition tool *Bi-dec*. This work is partially supported by SFI PI grant BEACON (09/IN.1/I2618).

7. 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] R. Ashenhurst. The decomposition of switching functions. In *Proceedings of an International Symposium on the Theory of Switching*, pages 74–116, 1957.

[3] H. K. Buning and T. Lettman. *Propositional Logic: Deduction and Algorithms*. Cambridge University Press, 1999.

[4] S. Chang, M. Marek-Sadowska, and T. Hwang. Technology mapping for TLU FPGA's based on decomposition of binary decision diagrams. *IEEE Trans. on CAD*, 15(10):1226–1236, 1996.

[5] H. Chen and J. Marques-Silva. Improvements to satisfiability-based boolean function bi-decomposition. In *VLSI-SoC*, pages 142–147, 2011.

[6] H. Chen and J. Marques-Silva. QBF-Based Boolean Function Bi-Decomposition. In *DATE*, 2012.

[7] M. Choudhury and K. Mohanram. Bi-decomposition of large boolean functions using blocking edge graphs. In *ICCAD*, pages 586–591, 2010.

[8] H. Curtis. *A new approach to the design of switching circuits*. Van Nostrand, Princeton, NJ, 1962.

[9] N. Eén and N. Sörensson. An extensible SAT-solver. In *SAT*, pages 502–518, 2003.

[10] J.-H. Jiang, C.-C. Lee, A. Mishchenko, and C.-Y. Huang. To SAT or Not to SAT: Scalable Exploration of Functional Dependency. *IEEE Trans. Comp.*, 59(4):457–467, Apr. 2010.

[11] Y. Lai, M. Pedram, and S. Vrudhula. BDD based decomposition of logic functions with application to FPGA synthesis. In *DAC*, pages 642–647, 1993.

[12] R.-R. Lee, J.-H. Jiang, and W.-L. Hung. Bi-decomposing large boolean functions via interpolation and satisfiability solving. In *DAC*, pages 636–641, 2008.

[13] H.-P. Lin, J.-H. Jiang, and R.-R. Lee. To sat or not to sat: Ashenhurst decomposition in a large scale. In *ICCAD*, pages 32–37, 2008.

[14] A. Mishchenko, B. Steinbach, and M. Perkowski. An algorithm for bi-decomposition of logic functions. In *DAC*, pages 103–108, 2001.

[15] T. Sasao and J. T. Butler. on bi-decomposition of logic functions. In *IWLS*, pages 1–6, 1997.

[16] C. Scholl. *Functional decomposition with application to FPGA synthesis*. Springer Netherlands, 2001.