Preprint from Logic and Architecture Synthesis, State-of-the-art and novel approaches,
Edited by Gabriele Saucier and Anne Mignotte, Chapman&Hall, 1995 (pp.57-63)

Efficient ROBDD based computation of common decomposition functions of multi-output boolean functions *

Christoph Scholl †, *Paul Molitor* ‡
† *Universität des Saarlandes*
Department of Computer Science, D 66041 Saarbrücken, FRG.
‡ *Martin-Luther Universität Halle*
Department of Computer Science, D 06099 Halle (Saale), FRG.

Abstract

One of the crucial problems multi-level logic synthesis techniques for multi-output boolean functions $f = (f_1, \ldots, f_m) : \{0,1\}^n \rightarrow \{0,1\}^m$ have to deal with is finding sublogic which can be shared by different outputs, i.e., finding boolean functions $\alpha = (\alpha_1, \ldots, \alpha_h) : \{0,1\}^p \rightarrow \{0,1\}^h$ which can be used as common sublogic of good realizations of $f_1, \ldots, f_m$.

In this paper we present an efficient ROBDD based implementation of this COMMON DECOMPOSITION FUNCTIONS PROBLEM (CDF).

Formally, CDF is defined as follows: Given m boolean functions $f_1, \ldots, f_m : \{0,1\}^n \rightarrow \{0,1\}$, and two natural numbers p and h, find h boolean functions $\alpha_1, \ldots, \alpha_h : \{0,1\}^p \rightarrow \{0,1\}$ such that $\forall 1 \leq k \leq m$ there is a decomposition of f_k of the form

$$f_k(x_1, \ldots, x_n) = g^{(k)}(\alpha_1(x_1, \ldots, x_p), \ldots, \alpha_h(x_1, \ldots, x_p),\\ \alpha^{(k)}_{h+1}(x_1, \ldots, x_p), \ldots, \alpha^{(k)}_{r_k}(x_1, \ldots, x_p), x_{p+1}, \ldots, x_n)$$

using a minimal number r_k of single-output boolean decomposition functions.

1 INTRODUCTION

The long term goal for logic synthesis is the automatic transformation from a behavioral description of a boolean function to near-optimal netlists, whether the goal is minimum delay, minimum area, or some combination. Most of the approaches attacking the multi-level logic synthesis problem use gate count as optimization criterion. A survey can be found in Brayton (1990). Alternatively, some recent papers (e.g. Hwang (1992), Lai (1993), Schlichtmann (1993)) propose an approach different from the one addressed above. This approach to multi-level logic synthesis which originates from Ashenhurst (1959), Curtis (1961), and Karp (1963) is based on minimizing communication complexity. The methods

*This work was supported in part by DFG grant SFB 124 and the Graduiertenkolleg of the Universität des Saarlandes

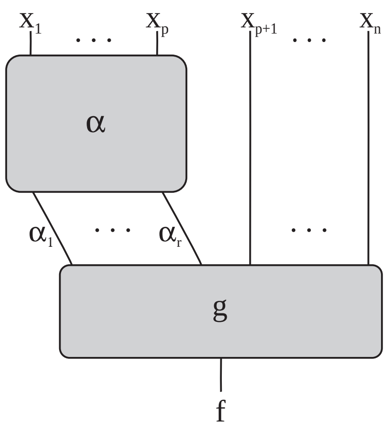

Figure 1 Decomposition of a boolean function $f : \{0,1\}^n \to \{0,1\}$

used to reduce communication complexity employ functional decomposition, i.e., given a boolean function $f : \{0,1\}^n \to \{0,1\}$ they are looking for boolean functions α and g, such that $f(x_1, \ldots, x_n) = g(\alpha(x_1, \ldots, x_p), x_{p+1}, \ldots, x_n)$ holds for all $(x_1, \ldots, x_n) \in \{0,1\}^n$ (see Figure 1). $\alpha = (\alpha_1, \ldots, \alpha_r){:}\{0,1\}^p \to \{0,1\}^r$ is a multi-output boolean function. The goal is to find decompositions where the number r of decomposition functions (i.e. the number of wires between block α and block g) is minimal. (If $r < p$, then the decomposition is called non-trivial.)

A fundamental step in logic synthesis is the identification of common sublogic. Methods to identify common sublogic by (algebraic or boolean) division were developed by Brayton et al. and included in the SIS package (Sentovich (1992)). In this paper we present a method to identify common sublogic for the case that boolean functions are realized by decomposition. This method comes into play when we have to process multi-output boolean functions $f = (f_1, \ldots, f_m) : \{0,1\}^n \to \{0,1\}^m$. Note that even if the original function f has only 1 output ($m = 1$), in most cases we need a generalization to multi-output boolean functions when we apply functional decomposition recursively to α and g.

All f_i are decomposed as single-output functions, but in order to identify common sublogic we make use of our freedom in the choice of the decomposition functions to compute such decomposition functions which can be used in the decomposition of as many f_i as possible. Unlike Lai (1994) we avoid to compute the huge set of all possible decomposition functions for all f_i to choose common decomposition functions of the functions f_i from these sets. We present an algorithm which directly computes a maximum number of common decomposition functions of $f_1, \ldots, f_m$.

In contrast to Molitor/Scholl (1994), which was based on function tables and decomposition charts, we efficiently make use of REDUCED ORDERED BINARY DECISION DIAGRAMS (ROBDD) during the computation of common decomposition functions. ROBDDs (Bryant (1986)) are compact representations for many of the boolean functions encountered in typical applications. In this paper we show that it is possible to carry out all necessary steps based on ROBDDs. This increases the efficiency of the approach in a high degree. In particular, we show that the computation of *common decomposition functions* for the decomposition of several single-output functions can be performed efficiently based on ROBDD's.

Benchmarking results show the new method to be efficient with respect to layout size, signal delay and running time.

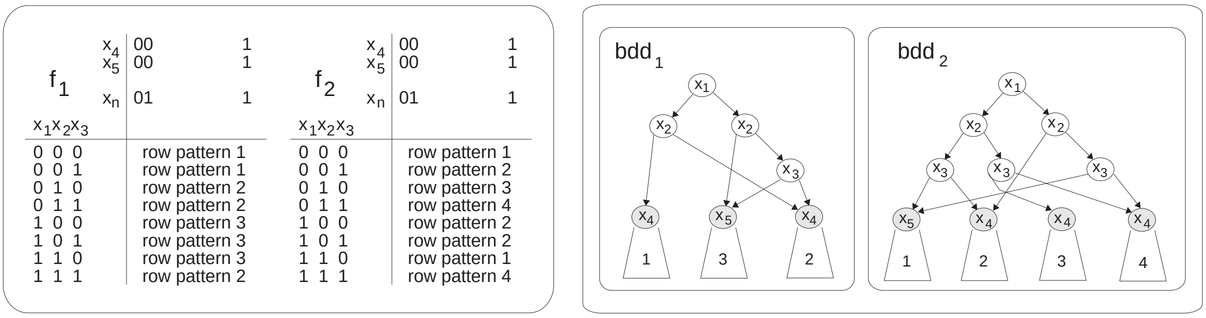

Figure 2 Charts and ROBDDs of the single-output boolean functions f_1 and f_2 which are decomposed with respect to $(\{x_1, x_2, x_3\}, \{x_4, \ldots, x_n\})$. Each chart obviously consists of 8 rows. A row pattern is associated to each row. There are three (four) different row patterns in $M(f_1)$ $(M(f_2))$ denoted by the numbers 1, 2, and 3 (1 to 4). Thus, $r_1 = r_2 = 2$ holds.

2 BASIC DEFINITIONS

Definition 1 *A* decomposition of a multi-output boolean function $f : \{0,1\}^n \to \{0,1\}^m$ *with respect to the input partition* $\{X_1, X_2\}$ $(X_1 = \{x_1, \ldots, x_p\}, X_2 = \{x_{p+1}, \ldots, x_{p+q}\},$ $p+q=n)$ *is a representation of* f *of the form*
$f_k(x_1, \ldots, x_n) = g^{(k)}(\alpha_1^{(k)}(X_1), \ldots, \alpha_{r'_k}^{(k)}(X_1), X_2)$ $(\forall k \in \{1, \ldots, m\})$,
with functions $\alpha_i^{(k)} : \{0,1\}^p \to \{0,1\}$, *and* $g^{(k)} : \{0,1\}^{r'_k+q} \to \{0,1\}$. $\alpha_i^{(k)}$ *are called* decomposition functions of f_k. $g^{(k)}$ *is called* composition function of f_k.

With respect to a given input partition $\{X_1, X_2\}$, a single-output function f_k can be represented as a $2^p \times 2^q$ matrix $M(f_k)$, the *decomposition matrix of* f_k or the *chart of* f_k with respect to $\{X_1, X_2\}$. (For illustration see Figure 2.) Each row and column of $M(f_k)$ is associated with a distinct assignment of values to the inputs in X_1 and X_2, respectively, such that $f_k(X_1, X_2) = M(f_k)[X_1, X_2]$ where $M(f_k)[X_1, X_2]$ represents the element of $M(f_k)$ which lies in the row associated with X_1 and the column associated with X_2.

Note that $(\alpha_1^{(k)}, \ldots, \alpha_{r'_k}^{(k)})$ of definition 1 encodes the rows of chart $M(f_k)$. Of course, the following property has to hold.

Encoding Property: If the row pattern of row $(v_1, \ldots, v_p) \in \{0,1\}^p$ differs from the row pattern of row $(v'_1, \ldots, v'_p) \in \{0,1\}^p$, then $(\alpha_1^{(k)}, \ldots, \alpha_{r'_k}^{(k)})$ has to assign different codes to $(v_1, \ldots, v_p)$ and $(v'_1, \ldots, v'_p)$.

The minimum number of communication wires required between the subcircuit which encodes the rows of $M(f_k)$ and the composition function $g^{(k)}$ is $\lceil \log p_1^{(k)} \rceil$ where $p_1^{(k)}$ is the number of distinct row patterns in $M(f_k)$. r_k will denote value $\lceil \log p_1^{(k)} \rceil$ in the following. In the following we will always consider decompositions with minimal number $r_k = \lceil \log p_1^{(k)} \rceil$ of decomposition functions.

If f_k is given by a ROBDD bdd_k, the minimal number of decomposition functions can be determined in an easy way too: For all $(v_1, \ldots, v_p) \in \{0,1\}^p$ the row pattern belonging to row $(v_1, \ldots, v_p)$ of $M(f_k)$ equals the function table of the cofactor $(f_k)_{x_1^{v_1} \cdot \ldots \cdot x_p^{v_p}}$ (with $x_i^0 = \overline{x_i}$ and $x_i^1 = x_i$). Thus the problem of determining the number $p_1^{(k)}$ of different row patterns of $M(f_k)$ is equivalent to the problem of computing the number of different cofactors $(f_k)_{x_1^{v_1} \cdot \ldots \cdot x_p^{v_p}}$. The ROBDD of the cofactor $(f_k)_{x_1^{v_1} \cdot \ldots \cdot x_p^{v_p}}$ is given by the sub-bdd of bdd_k whose root is reached by starting at the root of bdd_k and then following the path corresponding to $(v_1, \ldots, v_p)$. The roots of these cofactors are called *linking nodes* (the shaded nodes in Figure 2). Since f_k is given by a ROBDD, the number of different linking nodes of bdd_k obviously equals the number of different cofactors. The computational complexity of determining the number of different linking nodes is at most linear in the size of bdd_k since it can be determined by traversing bdd_k in a depth first search manner.

The rows of chart $M(f_k)$ induce a partition of $\{0,1\}^p$ into equivalence classes $K_1^{(k)}, \ldots, K_{p_1^{(k)}}^{(k)}$ such that $v, v' \in \{0,1\}^p$ belong to the same class $K_j^{(k)}$ if and only if the two corresponding row patterns of $M(f_k)$ are identical. We denote the corresponding equivalence relation by $\equiv_k$ and the set of the equivalence classes $\{K_1^{(k)}, \ldots, K_{p_1^{(k)}}^{(k)}\}$ by $\{0,1\}^p/_{\equiv_k}$.

Since every equivalence class $K_j^{(k)}$ is associated to exactly one linking node $n_j^{(k)}$ and vice versa, we are able to compute $K_j^{(k)}$ from the ROBDD bdd_k for f_k. We receive a BDD for the characteristic function of $K_j^{(k)}$ if we replace $n_j^{(k)}$ by the constant 1 and all other linking nodes by constant 0.

3 CDF

To compute decomposition functions (with domain X_1) of a multi-output function f which are used by different single-output functions f_k, we have to consider the following problem which will be denoted by CDF.

Given: Let $f = (f_1, \ldots, f_m) : \{0,1\}^n \to \{0,1\}^m$ be a multi-output boolean function, $A = \{X_1, X_2\}$ with $X_1 = \{x_1, \ldots, x_p\}$ and $X_2 = \{x_{p+1}, \ldots, x_n\}$ be an input partition, and h be a natural number with $h \leq r_k$ $(= \lceil \log p_1^{(k)} \rceil)$ $(\forall k)$.

Find: h single-output boolean functions $\alpha_1, \ldots, \alpha_h$, which can be used as decomposition functions of every single-output function f_k for $k = 1, \ldots, m$ such that there is a decomposition of f_k with minimal number r_k decomposition functions of the form
$f_k(x_1, \ldots, x_n) = g^{(k)}(\alpha_1(X_1), \ldots, \alpha_h(X_1), \alpha_{h+1}^{(k)}(X_1), \ldots, \alpha_{r_k}^{(k)}(X_1), X_2)$.
Of course, such h boolean functions need not to exist. We have proven the problem whether such functions $\alpha_1, \ldots, \alpha_h$ exist to be NP-complete. Nevertheless, we have to solve CDF[†]. An algorithm which is applicable from the practical point of view (as shown by the benchmarking results) is presented in this section.

3.1 Theoretical Background

We start with a theoretical result working towards a solution to CDF. It gives a condition necessary and sufficient that h single-output functions $\alpha_1, \ldots, \alpha_h : \{0,1\}^p \to \{0,1\}$ are common decomposition functions of $f_1, \ldots, f_m$. It is a generalization of a lemma shown by Karp (1963). For this, we need the following notations: Let $\theta^{(k)} : \{0,1\}^p \to \{1, \ldots, p_1^{(k)}\}$ be the function which maps $v \in \{0,1\}^p$ to the index j of the class $K_j^{(k)}$ to which it belongs.

[†]The (maximal) value of parameter h of CDF is determined by logarithmic search. After that we solve CDF for subsets of $\{f_1, \ldots, f_m\}$, but only for such subsets $\{f_{i_1}, \ldots, f_{i_l}\}$ where all pairs f_{i_j} and f_{i_k} have at least one common decomposition function. Note that this question is *not* NP-complete, but can be decided efficiently by dynamic programming. Also note that the algorithms for the computation of common decomposition functions given in this paper can be generalized in a canonical manner for the case that some of the decomposition functions $\alpha_i^{(k)}$ $(i > h)$ are already predetermined. More details of how the CDF algorithm is integrated in the tool can be found in Molitor/Scholl (1994).

Furthermore, for given $\alpha_{1,\ldots,h}$ [‡] and all $a \in \{0,1\}^h$, let $S_a^{(k)}$ be the set $\{\theta^{(k)}(v);\ \alpha_{1,\ldots,h}(v) = a\}$ of those classes which contain a row mapped to a by $\alpha_{1,\ldots,h}$. ($\alpha_{1,\ldots,h}$ is not able to tell these rows apart (see the Encoding Property).) Note that $S_a^{(k)}$ and $S_{a'}^{(k)}$ need not to be disjoint for $a \neq a'$, and that the number $|\,S_a^{(k)}\,|$ of elements of $S_a^{(k)}$ equals the number of *distinct* row patterns of $M(f_k)$ mapped to a by $\alpha_{1,\ldots,h}$.

Lemma 1 *$\alpha_1, \ldots, \alpha_h$ are common decomposition functions of $f_1, \ldots, f_m$ with respect to $\{X_1, X_2\}$ such that there is a decomposition of f_k with minimal number of decomposition functions of the form*
$f_k(x_1, \ldots, x_n) \ = \ g^{(k)}(\alpha_1(X_1), \ldots, \alpha_h(X_1), \alpha_{h+1}^{(k)}(X_1), \ldots, \alpha_{r_k}^{(k)}(X_1), X_2) \quad \forall k \in \{1, \ldots, m\}$
if and only if $\max\{|\,S_a^{(k)}\,|;\ a \in \{0,1\}^h\}$ *is* $\leq 2^{r_k - h}$ $(\forall k)$.

Proof. Since $(\alpha_{1,\ldots,h}, \alpha_{h+1,\ldots,r_k}^{(k)})$ has to assign different values to rows of chart $M(f_k)$ with different row patterns (see the Encoding Property), $\alpha_{h+1,\ldots,r_k}^{(k)}$ has to assign different values to those rows which cannot be told apart by $\alpha_{1,\ldots,h}$. As $\alpha_{h+1,\ldots,r_k}^{(k)}$ can produce at most $2^{r_k - h}$ different values, the statement of the lemma follows. □

3.2 Solution

h common decomposition functions $\alpha_1, \ldots, \alpha_h$ can be computed (on principle) by a (simplified) branch and bound algorithm (see also Molitor/Scholl 1994). It constructs the function table of $\alpha_{1,\ldots,h}$ row by row (assigning function values to all elements of $\{0,1\}^p$). Branches are pruned as soon as the condition of lemma 1 is violated for the initial part of the function table of $\alpha_{1,\ldots,h}$ constructed so far.

In order to speed up the branch and bound algorithm and to receive 'simpler' decompositon functions we restrict our search for common decomposition functions to a subclass of functions, which we will call 'equivalence preserving decomposition functions'[§]:

Definition 2 *A decomposition function $\alpha_i : \{0,1\}^p \to \{0,1\}$ of a boolean function $f_k : \{0,1\}^n \to \{0,1\}$ is said to* preserve equivalences *if $\alpha_i(v) = \alpha_i(v')$ holds for every $v, v' \in \{0,1\}^p$ with $v \equiv_k v'$.*

Common equivalence preserving decomposition functions $\alpha_1, \ldots, \alpha_h$ of $f_1, \ldots, f_m$ have to assign the same value to v and $v' \in \{0,1\}^p$ whenever there is a $k \in \{1, \ldots, m\}$ such that the rows of $M(f_k)$ corresponding to v and v' have identical row patterns. More formally, let
$v \sim v' \ \stackrel{\text{def}}{\Longleftrightarrow}\ (\exists 1 \leq k \leq m)\ v \equiv_k v'$,
then the corresponding equivalence relation partitions the rows, i.e. $\{0,1\}^p$, into equivalence classes $E_1, \ldots, E_l$ such that common equivalence preserving decomposition functions have to assign the same value to each $v \in E_i$. We will denote the set of these equivalence classes by $\{0,1\}^p/_\sim$.

Now we can modify our branch and bound algorithm, such that it makes assignments not to single elements of $\{0,1\}^p$ but to whole classes E_i. Since l mostly is much smaller than 2^p, this approach considerably reduces the running time (see also section 4).

[‡] $\alpha_{1,\ldots,h}$ denotes the tuple $(\alpha_1, \ldots, \alpha_h)$.

[§] In practical applications functions f_k often have some desirable properties like symmetry in some variables or independence of some variables. Equivalence preserving decomposition functions 'preserve such properties'.

Table 1 Experimental results.

Circuit	Running time *mulop*	*mulopII*	ratio	CDF	Layout size *sis*	*mulopII*	ratio	Signal delay *sis*	*mulopII*	ratio
9symml	1.40	1.23	1.14	0.69%	1194336	201400	5.93	27.6	13.6	2.03
C17	0.32	0.15	2.13	0.01%	28800	31744	0.91	4.2	4.2	1.00
cm138a	1.01	0.18	5.61	0.89%	103896	87480	1.19	5.8	6.8	0.85
cm151a	4.16	1.09	3.82	0.13%	95312	177712	0.54	12.6	16.4	0.77
cm152a	2.15	0.50	4.30	0.36%	85536	106704	0.80	10.0	13.2	0.76
cm162a	350.65	3.32	105.62	0.26%	131976	192000	0.69	12.0	13.2	0.91
cm163a	2923.31	2.35	1243.96	0.08%	144008	164416	0.88	13.0	10.4	1.25
cm82a	0.38	0.21	1.81	0.01%	74784	61600	1.21	7.2	7.0	1.03
cm85a	7.46	3.73	2.00	0.27%	165456	180000	0.92	10.2	11.0	0.93
cmb	1836.13	2.52	728.62	0.05%	204792	123496	1.66	9.4	6.8	1.38
decod	26.15	2.56	10.21	1.93%	140448	119496	1.18	6.2	5.0	1.24
f51m	3.14	1.83	1.71	0.28%	561184	251392	2.23	51.0	18.4	2.77
majority	0.44	0.08	5.50	0.01%	42200	39168	1.00	7.8	6.6	1.18
parity	111.06	1.37	81.07	0.00%	99408	96976	1.03	5.0	5.0	1.00
z4m1	0.66	0.76	0.87	0.16%	156288	103896	1.50	16.2	9.8	1.65
$\sum$					3228K	1937K	1.67	198.2	147.4	1.34

As already mentioned in section 2 ROBDDs $bdd_1^{(k)}, \ldots, bdd_{p_1^{(k)}}^{(k)}$ for the characteristic functions of the equivalence classes $K_1^{(k)}, \ldots, K_{p_1^{(k)}}^{(k)}$ with respect to $\equiv_k$ can easily be computed from the ROBDDs of the f_k. To compute the characteristic functions of the equivalence classes with respect to $\sim$, we implicitely construct a graph $G = (V, E)$ where the set V of vertices is given by the ROBDDs $bdd_j^{(k)}$ representing the equivalence classes $K_j^{(k)}$. At the end, there is an undirected edge $\{bdd_{j_1}^{(k_1)}, bdd_{j_2}^{(k_2)}\}$ if and only if $bdd_{j_1}^{(k_1)} \wedge bdd_{j_2}^{(k_2)} \neq \emptyset$, i.e., iff $K_{j_1}^{(k_1)} \cap K_{j_2}^{(k_2)} \neq \emptyset$. Obviously, there is a one-to-one relation between the set of the connected components (in the graph-theoretical sense) of G and the set of the equivalence classes $\{0,1\}^p/_\sim$. For every class E_i, there is a connected component CC_i of G such that the logical-or of the ROBDDs $bdd_j^{(k)}$ (for any fixed k) corresponding to vertices of CC_i results in a representation of E_i and vice versa.

4 EXPERIMENTAL RESULTS

We applied our tool, which uses the CDF algorithm described above as basis, to a number of benchmarks of the 1991 MCNC multi-level logic benchmark set. We will call the ROBDD based implementation of our tool *mulopII*. Our former implementation working on charts will be called *mulop* (Molitor/Scholl 1994).

Columns 1–3 of Table 1 show running times (in CPU seconds, measured on a SPARC-station 10/30 (64 MByte RAM)) of our ROBDD based implementation *mulopII* compared to those of our former version *mulop*. Experiments prove our ROBDD based version to be much more efficient than the former version.

The next column of Table 1 shows the fraction of running time which is used in the computation of common decomposition functions compared to the total running time of the tool *mulopII*. It shows that only a very small fraction of the total running time is used for the computation of common decomposition functions. The running time is dominated by the computation of good input partitions, *not* by the computation of common decomposition functions. This confirms our approach to compute common decomposition functions rather than to encode linking nodes in a straightforward manner.

Columns 5–7 of Table 1 show a comparison between *sis* (Sentovich (1992)) and *mulopII* with respect to layout size¶. For almost two thirds of the benchmark set, our approach dominates (or is as good as) that of *sis* with respect to layout size. Nevertheless, the signal delays of our realizations for more than two thirds of the circuits considered are better (or equal) than those of the realizations synthesized by *sis* (see columns 8–10 of Table 1).

5 CONCLUSION

We have presented a ROBDD based technique for computing common decomposition functions of multi-output boolean functions. This algorithm has been integrated in our multi-level synthesis tool which has been presented in Molitor/Scholl (1994) where more details of how the CDF algorithm is integrated can be found. The benchmarking results show that most of the circuits constructed by our synthesis tool are very efficient. They also prove it to be applicable in terms of running time.

REFERENCES

R.L. Ashenhurst. The decomposition of switching functions. In *Proceedings on an International Symposium on the Theory of Switching* held at Comp. Lab. of Harvard University, pages 74–116, 1959.

R.K. Brayton, G.D. Hachtel, and A. L. Sangiovanni-Vincentelli. Multilevel logic synthesis. *Proceedings of the IEEE*, 78(2):264–300, February 1990.

R.E. Bryant. Graph-based algorithms for boolean function manipulation. *IEEE Trans. on Computers*, C-35(8):677–691, August 1986.

H.A. Curtis. A generalized tree circuit. *J. Assoc. Comput. Mach.*, 8:484–496, 1961.

T. Hwang, R.M. Owens, and M.J. Irwin. Efficient computing communication complexity for multilevel logic synthesis. *IEEE Trans. on CAD*, CAD-11(5):545–554, May 1992.

R.M. Karp. Functional decomposition and switching circuit design. *Journal of Society of Industrial Applied Mathematics*, 11(2):291–335, June 1963.

Y. Lai, M. Pedram, and S. Vrudhula. BDD based decomposition of logic functions with application to FPGA synthesis. In *IEEE/ACM Design Automation Conference DAC93*, pages 642–647, 1993.

Y. Lai, K. Pan, and M. Pedram. FPGA Synthesis using Function Decomposition. In *Proceedings of ICCD94*, pages 30–35, 1994.

P. Molitor, C. Scholl. Communication based multilevel synthesis for multioutput boolean functions. In *Proceedings of the 4th Great Lakes Symposium on VLSI, Notre Dame, Indiana*, March 1994.

U. Schlichtmann. Boolean Matching and Disjoint Decomposition for FPGA Technology Mapping. In *Proceedings of the IFIP Workshop on Logic and Architecture Synthesis*, pages 83–102, 1993.

E. Sentovich et al. SIS: a system for sequential circuit synthesis. Department of EE and CS, UC Berkeley, May 1992.

¶The technology library consists of the 2-input gates from *stdcell2_2.genlib* available in *octtools*. Placement and routing was done by *TimberWolf* integrated in *octtools*.