Chortle-crf: Fast Technology Mapping for Lookup Table-Based FPGAs

Robert Francis, Jonathan Rose, Zvonko Vranesic

Department of Electrical Engineering, University of Toronto, Canada

Abstract

A new technology mapping algorithm for lookup table-based Field Programmable Gate Arrays (FPGA) is presented. The major innovation is a method for choosing gate-level decompositions based on bin packing. This approach is up to 28 times faster than a previous exhaustive approach. The algorithm also exploits reconvergent paths and replication of logic at fanout nodes to reduce the number of lookup tables in the circuit.

The new algorithm is implemented in the Chortle-crf program. In an experimental comparison Chortle-crf requires 14 % fewer lookup tables than Chortle [Fran90] and 10 % fewer lookup tables than mis-pga [Murg90a] to implement a set of benchmark networks.

Chortle-crf can also implement a network as a circuit of Xilinx 3000 series Configurable Logic Blocks (CLBs). To implement the benchmark networks as circuits of CLBs Chortle-crf requires 12 % fewer CLBs than mis-pga and 22 % fewer CLBs than XNFOPT [Xili89]. In these experiments Chortle-crf was an average of 68 times faster than mis-pga and 30 times faster than XNFOPT. [1]

1 Introduction

Field Programmable Gate Arrays (FPGAs) are a recent innovation in Application Specific Integrated Circuits (ASICs) that provide both large scale integration and user-programmability [Hsie88] [Ahre90]. The user-programmability of FPGAs can dramatically reduce ASIC turn-around time and manufacturing costs.

An FPGA consists of an array of programmable logic blocks and a programmable routing network. An important class of FPGAs consists of those that use logic blocks containing lookup tables, such as the first commercial FPGA [Cart86]. Moreover, recent studies in FPGA architectures have suggested that lookup tables are an area-efficient method of implementing combinational functions [Rose90]. A K-input lookup table is a digital memory with K address lines and a one-bit output. This memory contains 2^K bits and is capable of implementing any Boolean function of K input variables.

This paper presents a new algorithm for lookup table technology mapping which is implemented by the Chortle-crf program. Chortle-crf converts a combinational network of ANDs, ORs, and NOTs into a circuit of lookup tables where every lookup table has K or fewer inputs. The goal is to minimize the total number of K-input lookup tables in this circuit. For example, the network in Figure 1a can be implemented by the circuit of three 5-input lookup tables shown in Figure 1b. The dotted boundaries indicate the functions implemented by each lookup table. Note that one of the lookup tables uses only 4 of the available 5 inputs. All examples in the remainder of this paper will assume that K is equal to 5.

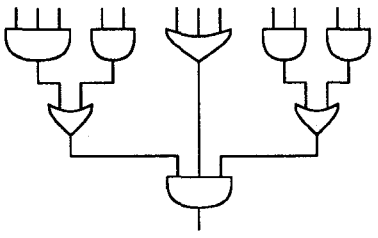
2 Background

Technology mapping produces a circuit that implements a combinational network using a restricted set of circuit elements. Early work in technology mapping, such as SOCRATES [Greg86] and the work by Kahrs [Kahr86], focused on circuits created from standard cell libraries. An important advance in library-based technology mapping was the introduction of dynamic programming by Keutzer [Keut87]. Other library-based technology mappers include misII [Detj87] and McMAP [Lisa87].

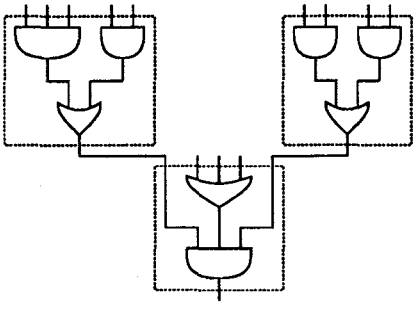
A lookup table of K-inputs can implement 2^{2^K} different Boolean functions of K variables. For values of K greater than 3 the library required to describe a K-input lookup table becomes impractically large and therefore technology mapping algorithms that deal specifically with lookup tables are required [Fran90]. Two previously reported lookup table technology mappers are Chortle [Fran90] and mis-pga [Murg90a].

The Chortle technology mapper presented in [Fran90] uses an exhaustive search to find the optimal gate-level decomposition of every node in a fanout-free tree. However, the partitioning of the original network into

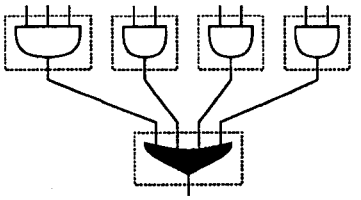
[1] This work was supported by NSERC Operating Grants #URF0043298 and #OGP0005280, a research grant from Bell-Northern Research, and a research grant from the ITRC of Ontario.



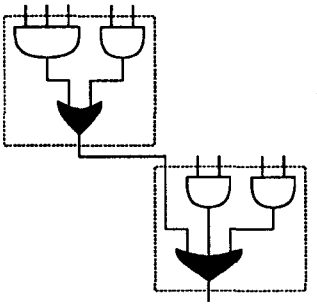
a) combinational network



b) circuit of 5-input lookup tables
Figure 1.



a) without gate decomposition



b) with gate decomposition
Figure 2.

fanout-free trees precludes optimizations that exploit reconvergent paths and replication of logic at fanout nodes.

The mis-pga technology mapper produces a circuit of lookup tables as an intermediate result [Murg90a]. It initially performs a non-optimal decomposition of the combinational network and then focuses on a covering problem to reduce the number of lookup tables in the circuit. The covering problem does allow optimizations that exploit reconvergent paths and replication of logic at fanout nodes.

3 The Chortle-crf Algorithm

A major innovation in Chortle-crf is the application of *bin packing* to choosing gate-level decompositions. Two other important features are the exploitation of reconvergent paths and replication of logic at fanout nodes to reduce the number of lookup tables in the circuit.

The principal technique used by Chortle-crf is dynamic programming. The combinational network is traversed beginning at the primary inputs and proceeding toward the primary outputs. At each node a circuit implementing the cone extending from the node to the primary inputs of the network is constructed. This circuit is referred to as the *Best Circuit* implementing the node.

Chortle-crf has two goals when constructing the Best Circuit. The first is to minimize the number of lookup tables in the circuit and the second is to maximize the number of unused inputs at the output lookup table. These unused inputs are important because they may allow subsequent nodes to be implemented without the addition of extra lookup tables.

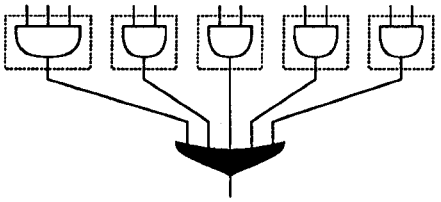
3.1 Bin Packing Approach to Gate Decomposition

The key to constructing the Best Circuit implementing a node is finding the decomposition of the node that reduces the number of lookup tables in the final circuit. For example, five lookup tables are required to implement the tree shown in Figure 2a. In Figure 2b, the single OR node of Figure 2a has been decomposed into two OR nodes, which allows the tree to be implemented with just two lookup tables.

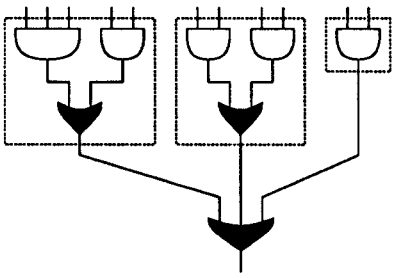
The construction of the Best Circuit for a node depends upon the Best Circuits that implement the node's immediate fanin nodes. The order of the network traversal ensures that these immediate fanin circuits have been previously constructed. The output lookup tables of the fanin Best Circuits will be referred to as the fanin lookup tables. Figure 3a shows an OR node and its five fanin lookup tables.

The goal of finding the best decomposition is attained by constructing a tree of lookup tables that implements both the functions of the fanin lookup tables and a decomposition of the node. This tree must contain the minimum number of lookup tables and the output (root) lookup table must have the maximum number of unused inputs possible without increasing the number of lookup tables in the tree.

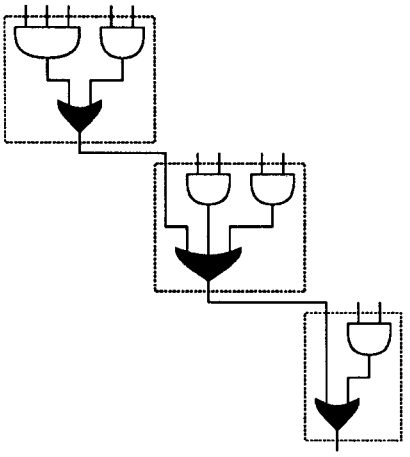
The tree of lookup tables is constructed in two steps. First, a two-level decomposition is constructed and then this decomposition is converted into a multi-level decomposition. Figures 3b and 3c illustrate the two-level and multi-level decompositions constructed from the



a) fanin lookup tables



b) two-level decomposition



c) multi-level decomposition

Figure 3.

fanin lookup tables of Figure 3a.

3.1.1 Two-Level Decomposition

The two-level decomposition consists of a single *first-level node* and several *second-level nodes.* In Figure 3b the 3-input OR node is the first-level node and its three inputs are the second-level nodes. Each second-level node implements the operation of the node being decomposed over a subset of one, some, or all of the fanin lookup tables. In Figure 3b there are three second-level nodes each of which is implemented by a lookup table. The first-level node is not yet implemented by any lookup tables, however, it will be implemented when the two-level decomposition is converted into a multi-level decomposition.

The two-level decomposition is constructed using a bin packing algorithm. In general, the goal of bin packing is to find the minimum number of bins into which a set of boxes can be packed [Gare79]. In this case, the bins are the second-level lookup tables and the boxes are the fanin lookup tables. The capacity of each bin is K, and the size of each box (fanin lookup table) is its number of used inputs. In Figure 3a the boxes have sizes 3, 2, 2, 2, and 2. In Figure 3b the final contents of the packed bins are 5, 4, and 2. The bin packing algorithm used is First Fit Decreasing as outlined in Figure 4 [Gare79].

```
FirstFitDecreasing
   {
   start with an empty bin list

   while there are unpacked boxes
      {
      if the largest unpacked box will not fit
      within any bin in the bin list
         {
         create an empty bin and
         add it to the end of the bin list
         }

      pack the largest unpacked box into the
      first bin it will fit within
      }
   }
```

Figure 4: Pseudo code for First Fit Decreasing

3.1.2 Multi-Level Decomposition

The decomposition tree is completed by implementing the first-level node with a tree of lookup tables. The inputs to the leaf lookup tables of this first-level tree are the outputs of the second-level lookup tables of the two-level decomposition. Any second-level lookup table with unused inputs can be used to implement a portion of the first-level tree, thereby reducing the total number of lookup tables in the decomposition tree. Figure 3c illustrates the multi-level decomposition constructed from the two-level decomposition of Figure 3b.

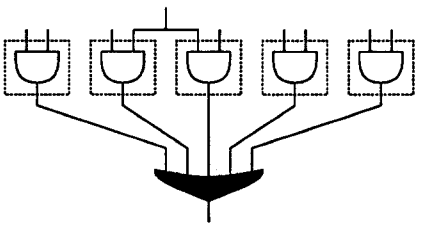
The detailed procedure for converting the two-level decomposition into a multi-level decomposition is outlined in Figure 5.

The final multi-level decomposition can be shown to be optimal if the network is a fanout-free tree and the value of K is less than or equal to 5 [Fran91]. For networks partitioned into fanout-free trees the bin packing approach is up to 28 times faster than the previous exhaustive search approach [Fran90], yet it produces circuits with the same number of lookup tables. This improvement in speed makes it practical to consider optimizations exploiting reconvergent paths and replication of logic at fanout nodes, as discussed in the following sections.

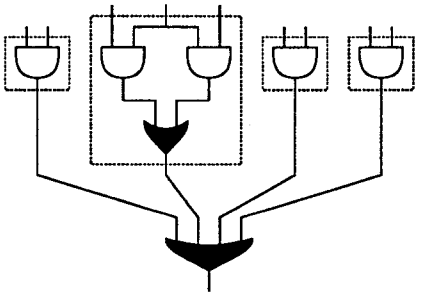
```
MultiLevel
   {
   while there is more than one unconnected bin
      {
      if there are no free inputs among the
      remaining unconnected bins
         {
         create an empty bin and
         add it to the end of the bin list
         }

      connect the most filled unconnected bin to
      the next unconnected bin with a free input
      }
   }
```

Figure 5: Pseudo code for multi-level conversion



a) fanin lookup tables with shared input



b) realized reconvergent paths

Figure 6.

3.2 Exploiting Reconvergent Paths

It is possible to exploit local reconvergent paths to find a better circuit implementing a node. The following discussion uses the terminology of the previous section, where the fanin lookup tables are referred to as boxes and the second-level lookup tables are referred to as bins.

If two boxes share the same input, then there exists a pair of reconvergent paths. If the total number of distinct inputs to these two boxes is less than or equal to K, then it is possible to pack the two boxes into one bin. When these two boxes are packed into the same bin, the volume occupied is the total number of *distinct* inputs, which is less than the sum of the boxes' individual sizes. Figure 6a shows a pair of boxes that share an input and Figure 6b shows the pair of reconvergent paths realized within a bin.

By merging the two boxes and realizing the pair of reconvergent paths within a single lookup table, a smaller portion of the bin is occupied. This may lead to a superior bin packing, which in turn may lead to a superior Best Circuit.

However, two boxes can only be merged if they are packed into the same bin. The two boxes can be forced into the same bin by merging them before the bins are packed. Forcing these two boxes into one bin may interfere with the bin packing algorithm and actually result in an inferior packing. To find the Best Circuit, both the packing with the forced merge and the packing without the forced merge need to be considered.

A further complication is that more than one pair of reconvergent paths may terminate at the node. To find the Best Circuit, Chortle-crf begins by finding all pairs of local reconvergent paths. For every possible combination of these pairs, including none, a circuit is constructed by first merging the respective boxes of the chosen pairs and then proceeding with the bin packing. The circuit with the fewest lookup tables (and the greatest number of unused inputs at the output lookup table) is retained as the Best Circuit. This realization of reconvergent paths is a greedy local optimization that is considered at every node as the network is traversed.

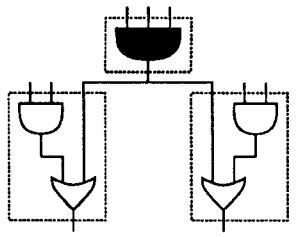
In our experiments with the MCNC benchmark networks the largest number of reconvergent pairs at any one node has been found to be six pairs. The bin packing approach is fast enough to make the search of all possible combinations of these pairs practical.

3.3 Replication of Logic at Fanout Nodes

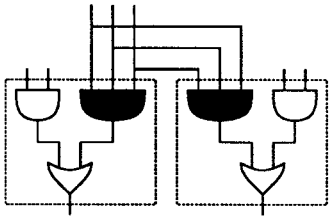
The previous version of Chortle partitions the combinational network into a set of fanout-free trees [Fran90]. This forces every fanout node to be explicitly implemented as the output of a lookup table, and allows these nodes to be treated as primary inputs to the rest of the network.

It is possible to implement the fanout nodes implicitly inside lookup tables, which requires the replication of some logic at a fanout node. This replication may decrease the total number of lookup tables in the circuit implementing the network. For example, in Figure 7a, three lookup tables are required to implement the network when the fanout node is explicitly implemented. In Figure 7b, the AND gate implementing the fanout node is replicated and only two lookup tables are required to implement the network.

When the dynamic programming traversal of the network encounters a fanout node the Best Circuit implementing the fanout node is constructed. At this point



a) no replicated logic



b) with replicated logic

Figure 7.

two options are considered. The fanout node can be either explicitly implemented, or implicitly implemented. If the fanout node is explicitly implemented it is treated as a primary input to the rest of the network. If it is implicitly implemented, a replica of the function of the *output lookup table* is made for each fanout edge. This replica replaces the fanout node as the source of the edge.

Every path starting with an edge from a fanout node will eventually reach another fanout node or a primary output of the network. These subsequent fanout nodes and primary outputs will be referred to as the *visible* nodes.

To determine if the replication is worthwhile Chortle-crf solves a series of subproblems. For each visible node the Best Circuit implementing the visible node is constructed twice; once with the replication and once without the replication. Each subproblem is itself solved using Chortle-crf with the assumption that any remaining fanout nodes encountered in these subproblems are explicitly implemented and can therefore be treated like primary inputs. The bin packing approach is fast enough to make solving these subproblems practical.

After the subproblems have been solved the total number of lookup tables required to implement the visible nodes both with and without the replication are known. If the total number of lookup tables is reduced by the replication, then the replication is retained. The replication of logic is considered at every fanout node as it is encountered by the dynamic programming traversal of the network.

Network	Chortle-crf				mis-pga
	-c lookups	-cr lookups	-cf lookups	-crf lookups	lookups
z4ml	9	9	9	6	8
misex1	20	20	19	19	11
vg2	24	24	23	21	30
5xp1	34	31	34	27	31
count	47	45	40	31	31
9symml	63	59	62	55	56
9sym	69	65	67	59	72
apex7	72	71	71	64	64
rd84	76	76	74	73	40
e64	95	95	80	80	82
C880	115	110	112	86	103
apex2	123	123	121	120	80
alu2	131	121	127	116	129
duke2	138	136	126	120	128
C499	166	164	158	74	66
rot	219	207	208	189	200
apex6	232	219	230	212	243
alu4	238	219	227	195	235
apex4	603	600	579	558	765
des	1073	1060	1050	952	1016
total	3547	3454	3417	3057	3390

Table 1: Results for K = 5

4 Results

To evaluate Chortle-crf a series of experiments were performed on networks from the MCNC logic synthesis benchmark suite. Four experiments were performed on each network:

-c using only the constructive bin packing approach

-cr using the reconvergent optimization

-cf using the replication optimization

-crf using both reconvergent and replication

The first step in the experimental procedure was technology independent logic optimization using the misII logic optimizer with the standard script [Bray86]. Chortle-crf was then used to implement the networks as circuits of 5-input lookup tables. Note that Chortle-crf is capable of implementing networks as circuits of K-input lookup tables for values of K from 2 to 10.

Table 1 records the number of 5-input lookup tables required to implement the networks in each of the four experiments. The reconvergent optimization reduced the total number of lookup tables required to implement the networks by 2.7 % , and the replication optimization reduced the total number of lookup tables by 3.7 %. Combining both optimizations reduced the total number of lookup tables by 14 %.

The reduction achieved when using both optimizations together often exceeds the sum of the individual reductions. This occurs when reconvergent paths that cross fanout nodes are found and realized within a single

Network	Chortle-crf			
	-c CLBs	-cr CLBs	-cf CLBs	-crf CLBs
z4ml	5	5	7	3
misex1	14	14	14	14
vg2	20	19	21	18
5xp1	23	20	23	20
count	32	31	32	27
9symml	50	42	50	41
9sym	52	44	56	42
apex7	48	45	49	42
rd84	52	52	53	53
e64	48	48	54	54
C880	75	70	94	69
apex2	94	90	97	93
alu2	94	86	98	83
duke2	88	87	91	89
C499	84	84	96	50
rot	134	129	144	131
apex6	169	161	169	161
alu4	165	144	174	138
apex4	457	451	463	448
des	714	695	797	743
total	2418	2317	2582	2319

Table 2: CLB Results

Network	Chortle-crf		mis-pga		XNFOPT	
	CLBs	sec. [1]	CLBs	sec. [2]	CLBs	sec. [1]
z4ml	3	0.8	7	-	6	296.5
misex1	14	0.7	10	-	12	298.2
vg2	18	0.6	21	25.6	20	299.7
5xp1	20	3.2	23	45.5	19	301.1
count	27	2.0	28	-	32	301.9
9symml	41	59.1	43	-	56	901.2
9sym	42	62.9	59	-	52	305.1
apex7	42	2.9	50	117.3	51	304.6
rd84	53	15.4	32	65.1	38	303.2
e64	54	1.9	61	-	65	901.5
C880	69	12.6	82	-	101	1809.4
apex2	93	34.9	70	-	102	909.7
alu2	83	56.3	102	-	91	907.8
duke2	89	9.1	105	357.1	99	903.6
C499	50	15.9	50	137.5	121	1847.0
rot	131	14.0	153	844.8	166	1811.4
apex6	161	25.3	191	1376.8	198	1822.6
alu4	138	178.1	189	-	232	1849.4
subtotal	1128		1276			
apex4	448	323.1	-	-	528	1931.5
des	743	291.9	-	-	988	15831.1
total	2319				2977	

[1] execution times on a Sun 3/60
[2] execution times on a VAX 8800

Table 3: CLB Results

lookup table. A dramatic example is the network C499, where using both optimizations reduces the number of lookup tables by 55 %.

As an intermediate result the mis-pga technology mapper produces a circuit of 5-input lookup tables [Murg90a]. The sixth column of Table 1 records the number of 5-input lookup tables in the circuits produced by mis-pga [Murg90b]. In total, Chortle-crf required 10 % fewer lookup tables than mis-pga to implement the benchmark networks.

4.1 Xilinx CLBs

The Xilinx 3000 series of FPGAs uses lookup tables to implement combinational logic [Hsie88]. These devices contain an array of Configurable Logic Blocks (CLBs). Each CLB can implement one 5-input lookup table or two 4-input lookup tables as long as the total number of distinct inputs to the CLB is less than or equal to 5.

A circuit of CLBs can be derived from each circuit of 5-input lookup tables by using one CLB to implement each lookup table. The number of CLBs can be reduced by finding pairs of lookup tables that fit inside a single CLB. Finding the maximum number of such pairs can be restated as a Maximum Cardinality Matching problem [Murg90a] [Gibb85]. Table 2 records the number of CLBs in the circuits derived from the previous Chortle-crf experiments.

Note that using only the replication optimization can increase the number of CLBs in the derived circuit, even when the optimization reduces the number of lookup tables. The replication of logic at a fanout node may increase the number of inputs used at some lookup tables thereby precluding some pairings of lookup tables into CLBs and reducing the maximum number of pairs that can be found. If the reduction in the number of pairs exceeds the reduction in the number of lookup tables then the replication will result in a net increase in the number of CLBs.

Two other logic synthesis systems capable of implementing networks as circuits of CLBs are mis-pga [Murg90a] and the Xilinx proprietary design system [Xili89]. Chortle-crf can be compared to these systems on the basis of the number of CLBs in the final circuits and execution time. Table 3 records the number of CLBs required to implement the benchmark networks using Chortle-crf, mis-pga and Xilinx software. In total, Chortle-crf required 12 % fewer CLBs than mis-pga and 22 % fewer CLBs than XNFOPT to implement the benchmark networks.

The table also records the execution times for Chortle-crf on a Sun 3/60 and mis-pga on a VAX 8800 [Murg90a]. In the Xilinx design system technology mapping is performed by the two programs XNFOPT and XNFMAP [Xili89]. Note that XNFOPT will run indefinitely and in these experiments limits were placed on its execution time. The seventh column of Table 3 records the total execution time of the two programs on a Sun 3/60. It should be noted that by conservative

estimate a VAX 8800 is twice as fast as a Sun 3/60. Taking into account the relative speed of the Sun 3/60 and the VAX 8800, Chortle-crf is an average of 68 times faster than mis-pga and 30 times faster than XNFOPT.

5 Conclusions

The bin packing approach to gate decomposition described in this paper is up to 28 times faster than a previous exhaustive search approach. The improved speed of gate decomposition makes it practical to consider local optimizations that exploit both reconvergent paths and replication of logic at fanout nodes.

Using both of these optimizations, Chortle-crf required 14 % fewer 5-input lookup tables than Chortle [Fran90] and 10 % fewer lookup tables than mis-pga [Murg90a] to implement a set of benchmark networks.

Chortle-crf is also capable of implementing networks as circuits of Xilinx 3000 series CLBs. To implement the benchmark networks as circuits of CLBs, Chortle-crf required 12 % fewer CLBs than mis-pga and 22 % fewer CLBs than XNFOPT. On average, Chortle-crf was 68 times faster than mis-pga and 30 times faster than XNFOPT.

6 Future Work

Currently, the optimizations exploiting reconvergent fanout and replication of logic are evaluated locally. There are, however, global interactions among these optimizations. The search for reconvergent paths should be extended to include those paths not found by the local search. As well, realizing a pair of reconvergent paths within a single lookup table may depend upon the replication of logic at multiple fanout nodes.

There are cases where the optimizations requiring replication of logic at different fanout nodes may be mutually exclusive. A computationally tractable method of determining which set of replications at fanout nodes will result in the minimum number of lookup tables for the entire network is needed.

References

[Ahre90] M. Ahrens, et al., "An FPGA Family Optimized for High Densities and Reduced Routing Delay," Proc. 1990 CICC, May 1990, pp. 31.5.1-31.5.4.

[Bray86] R. Brayton, et al., "Multiple-Level Logic Optimization System," Proc. ICCAD, Nov. 1986, pp. 356-359.

[Cart86] W. Carter et al., "A user Programmable reconfigurable gate array," Proc. CICC, May 1986, pp 233-235.

[Detj87] E.Detjens et. al, "Technology Mapping in MIS", Proc. ICCAD 87, Nov 1987, pp. 116-119.

[Fran90] R. J. Francis, J. Rose, K. Chung, "Chortle: A Technology Mapping Program for Lookup Table-Based Field Programmable Gate Arrays," Proc. 27th DAC, June 1990, pp. 613-619.

[Fran91] R. J. Francis, "Technology Mapping for Lookup Table-Based FPGAs," Ph.D. Thesis in preparation, University of Toronto, Department of Electrical Engineering.

[Gare79] M. R. Garey, D. S. Johnson, "Computers and Intractability, A Guide to the Theory of NP-Completeness," W. H. Freeman and Co., 1979, pp. 124-129.

[Gibb85] A. Gibbons, "Algorithmic Graph Theory," Cambridge University Press, 1985, pp. 125-133.

[Greg86] D. Gregory, et al., "Socrates: a system for automatically synthesizing and optimizing combinational logic," Proc. 23rd DAC, June 1986, pp. 79-85.

[Hsie88] H. Hsieh, et al., "A 9000-Gate User-Programmable Gate Array," Proc. 1988 CICC, May 1988, pp. 15.3.1 - 15.3.7.

[Kahr86] M. Kahrs, "Matching a parts library in a silicon compiler," IEEE ICCAD, 1986, pp. 169-172.

[Keut87] K. Keutzer, "DAGON: Technology Binding and Local Optimization by DAG Matching," Proc. 24th DAC, June 1987, pp. 341-347.

[Lisa87] R. Lisanke, F. Brglez, G. Kedem, "McMAP: A Fast Technology Mapping Procedure for Multi-Level Logic Synthesis," Proc. ICCD, Oct. 1988, pp. 252-256.

[Murg90a] R. Murgai, et al., "Logic Synthesis for Programmable Gate Arrays," Proc. 27th DAC, June 1990, pp. 620-625.

[Murg90b] R. Murgai, private correspondence.

[Rose90] J. Rose, R. J. Francis, D. Lewis, P. Chow, "Architectures of Field-Programmable Gate Arrays: The effect of Logic Block Functionality of Area Efficiency," IEEE Journal of Solid-State Circuits, Vol. 25, No. 5, Oct. 1990, pp. 1217-1225.

[Xili89] XACT LCA Development System, Vol. II, Xilinx Inc., 1989.