UNIVERSITY OF CALIFORNIA

Los Angeles

Logic Synthesis of MEM Relay Circuits

A thesis submitted in partial satisfaction

of the requirements for the degree

Master of Science in Electrical Engineering

By

Kevin Dwan

2011

The thesis of Kevin Dwan is approved.

Robert Candler

Lei He

Dejan Marković, Committee Chair

University of California, Los Angeles

2011

TABLE OF CONTENTS

LIST OF FIGURES

List of Tables

ACKNOWLEDGEMENTS

I would like to express my deepest gratitude to my advisor, Professor Dejan Marković, for his constant guidance throughout my graduate experience. I have benefited greatly from his vast technical knowledge and tutelage. His patience and wisdom have not only helped me grow as an engineer, but helped me to develop as a person and a professional. I would also like to thank my colleague Cheng Wang whose knowledge and experience in relay circuit design was instrumental in this work. I am also very thankful to the entire MEM relay research team whose work made this thesis possible. I also wish to thank my thesis committee members, Professor Robert Candler and Professor Lei He for reviewing my thesis.

I am also very thankful to the rest of my research group colleagues. I would especially like to thank Richard Dorrance and Yuta Toriyama for their patience in reading and helping to edit this manuscript. The rest of my colleagues including Henry Chen, Vaibhav Karkare, Sarah Gibson, Rashmi Nanda, Fengbo Ren, Tsung-Han Yu, Fang-Li Yuan, Chia-Hsiang Yang, Vicki Wang, and Qian Wang have also been great friends. The support and assistance of my entire research have been very important in my life.

Most of all, I would like to thank my parents Roger and Margie Dwan. Their unconditional love and support have given me the confidence to face seemingly impossible challenges. Their guidance has shaped me into the person I am today. Thank you.

ABSTRACT OF THE THESIS

Logic Synthesis of MEM Relay Circuits

by

Kevin Dwan

Master of Science in Electrical Engineering

University of California, Los Angeles, 2011

Professor Dejan Marković, Chair

As CMOS scaling begins to reach its fundamental limits, micro-electro-mechanical (MEM) relays provide an attractive option for improvements in energy efficiency due to their low leakage and near ideal I-V characteristics. However, mechanical actuation of MEM relays introduces significantly more delay than traditional CMOS electrical delay. Circuit designers have mitigated this effect in relay based circuits by arranging for all mechanical actuations to happen simultaneously. However, the design of these customized circuit topologies requires significant time and effort. A general method for designing optimized relay circuits with any arbitrary logic function is presented. Optimizations are performed at the truth table and schematic level to reduce the number of devices needed in the design. These circuit techniques are implemented in a synthesis tool to automate the design process. The resulting circuits match custom designs in delay and device count while minimizing the amount of design effort required by the user. The output circuits can easily be ported to commercial place and route tools creating a simple automated MEM relay circuit design flow.

CHAPTER I

Introduction

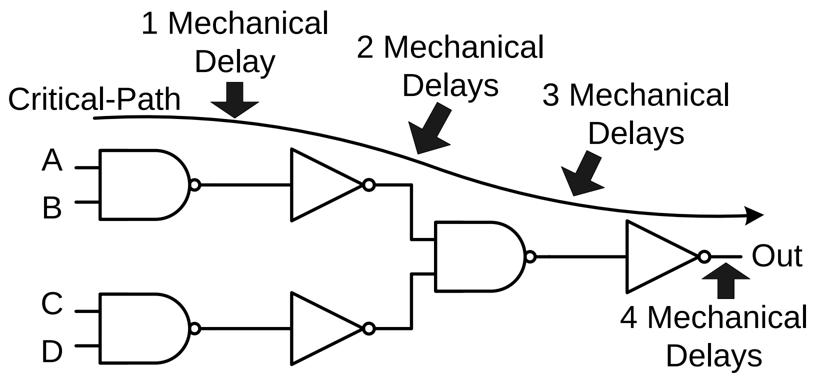
1.1 Overview of MEM Relays

In traditional CMOS circuit design, technology scaling has historically achieved significant gains in performance, cost, and energy efficiency. Recently however, threshold voltages have become fine tuned to the point where they balance leakage and switching currents. Further threshold voltage scaling would therefore increase the energy consumption per operation. Without the flexibility of threshold voltage scaling, continued scaling of the supply voltage causes a decrease in performance, spurring a trend toward parallel circuit designs.

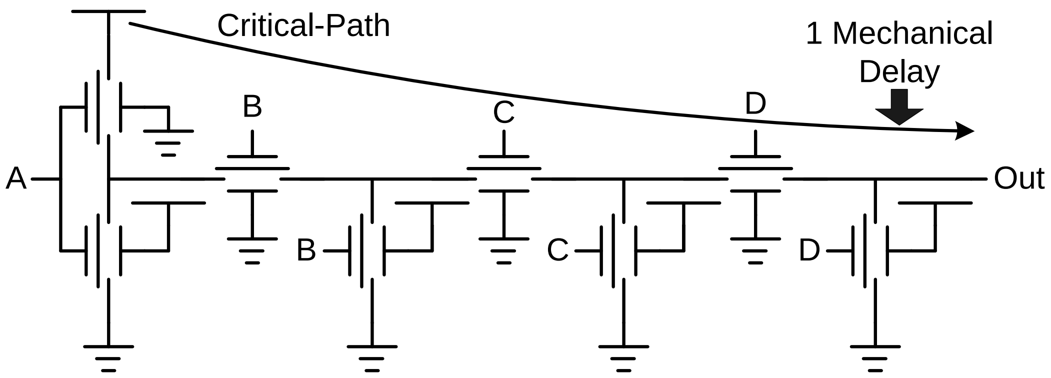
However, this trend will not be effective indefinitely because CMOS transistor energy efficiency is inherently limited by its sub-threshold leakage. In sub-threshold operation, an increase in threshold voltage decreases leakage current by the same amount that it increases delay. Therefore regardless of how slowly a circuit is allowed to run, the energy per operation cannot decrease past a defined minimum level [1]. The solution to improving energy efficiency thus lies in finding a device with more ideal leakage characteristics [2].

Micro-electro-mechanical (MEM) relay devices offer an attractive solution with nearly ideal switching characteristics and much steeper sub-threshold slope than CMOS.

In fact, an on-to-off current ratio of ten orders of magnitude has been demonstrated as well as immeasurably low leakage currents [3]. These devices are mechanical switch circuits that operate through electrostatic actuation and which have similar circuit behavior as traditional CMOS transistors.



(a)



(b)

Figure 1-1: A MEM relay implementation of a 4-input AND function in (a) a design optimized for CMOS and (b) a design optimized for relays. The critical-path delay is marked at various nodes.

Although they have near ideal I-V characteristics, MEM relays suffer from a relatively slow mechanical delay compared to electrical switching delays of traditional CMOS circuits. Specifically, predictions about scaled relays fabricated with a 90nm lithography process show delays to be in the 10's of nanoseconds which contrasts with similarly scaled CMOS transistors with delays ranging from the 100's of picoseconds to nanoseconds [4]. This drawback has been significantly mitigated by employing special design techniques to implement logic as large complex gates in order to minimize critical-path delay [5]. Figure 1-1 shows a comparison between a traditional CMOS based approach in (a) and these design techniques in (b) resulting in a critical-path reduction from four to one mechanical delay. The merits of MEM relays have been further demonstrated in a test chip confirming the merits of this design technique in reducing circuit delay on several important building blocks of VLSI digital systems such as logic, latches, memory, and I/O circuits. Analog and mixed signal devices have also been demonstrated in highly energy efficient DAC and ADC circuits [6]. The MEM relay technology for circuit design applications is still relatively young with devices demonstrated with sizes in the 10's of microns [7]. However, scalable relay models have predicted that at the 90nm technology node, MEM relays can achieve energy-delay characteristics that are nearly an order of magnitude better than CMOS over a wide range of frequencies with only three times the area [4, 5].

1.2 Motivation for Synthesis

Currently, circuit designers use a completely custom approach when designing relay circuits in order to leverage the energy-efficiency benefits without incurring severe penalties in delay or area. This process requires significant time and effort and requires designers to become uniquely familiar with each type of logic circuit in order to overcome the slow switching behavior of MEM relays. MEM relays also come in three different variations, including 4-terminal, 6-terminal, and seesaw type relays, which offer additional flexibility than CMOS counterparts and thus require more innovation to fully utilize. It is clear to see that the necessary design effort for relay circuit designers is substantial.

Thus far this custom strategy has been effective in generating optimized circuits for very small designs as discussed in [4-6], and which have been used to prove the potential for large-scale integration. In order to realize this potential however, much more design effort will be needed to increase the complexity and size of the circuits designed as well as develop relay optimized designs for all necessary components. The synthesis tool presents an alternative to custom design that significantly decreases this design effort while producing circuits that rival and potentially even surpass current customized circuits.

In order to produce circuits that are comparable to custom designed circuits, the synthesis tool focuses on two main design merits. First, the tool aims to create circuits with a single mechanical delay in the critical path between inputs and outputs. This matches the behavior of custom designed circuits. Second, the tool aims to minimize the number of relays needed to implement a design to ultimately minimize the area.

1.3 Thesis Outline

The remainder of the thesis is organized as follows. In Chapter 2, I discuss the fundamental structure and circuit behavior of the MEM relay as well as some previous works. Chapter 3 presents a methodology for designing low delay relay circuits with arbitrary function. Chapter 4 presents an overview of the synthesis tool and the various optimization techniques used by it. Chapter 5 discusses the results of various circuits that have been synthesized by the tool. Finally, Chapter 6 concludes the thesis.

CHAPTER II

Structure and Operation of MEM Relays

2.1 Structure of MEM Relays

The MEM relay device consists of a movable poly-SiGe gate suspended over tungsten drain, source, and body terminals. When the voltage difference between the gate and body terminals, $|V_{gb}|$, is greater than a threshold pull-in voltage, V_{pi}, the electrostatic force between them pulls the gate towards the body. When in this actuated state, a conductive channel located under the gate terminal contacts both the source and drain terminals and causes a short between them. When the $|V_{gb}|$ is lowered below a threshold pull-out voltage, V_{po}, the electrostatic force weakens and four flexure springs return the gate to its original state causing an air gap between the channel and the drain and source terminals. In this state, the drain and source terminals are electrically isolated. Figure 2-1 shows an example of these states as well as a diagram and a scanning electron microscope (SEM) image of this 4-terminal device [4].

As understanding of the device fabrication process improved, additional variations on the original device were developed including a 6-terminal device and a seesaw device. Figure 2-2 shows a diagram and an SEM image of this 6-terminal device. This device differs from the standard 4-terminal variation in that it has an extra pair of source and drain terminals. When the gate is actuated, each pair is shorted by its respective channel. Otherwise all source and drain terminals are electrically isolated. Figure 2-3 shows an example of the seesaw relay device. This device has two sets of drain, source, and body terminals and a single gate terminal. When no voltages are applied, the torsion beams on the poly-SiGe gate keep the gate suspended above both body terminals. For combinational logic, the body terminals are generally biased at opposite voltages, one at V_{DD} and one at *Gnd*. As the gate terminal changes voltages, the electrostatic force will cause it to be pulled towards one of the two body terminals. The source and drain pair corresponding to this body terminal will be shorted while the other pair will be open. This gate structure actuates to each side like a seesaw, hence the name.

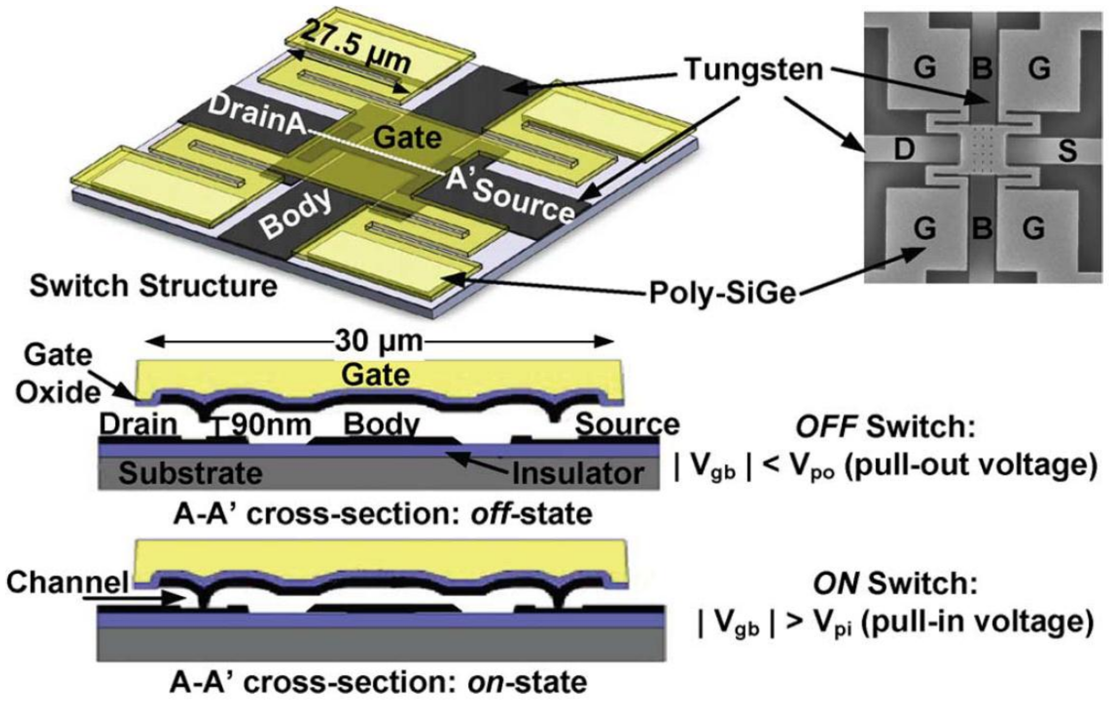
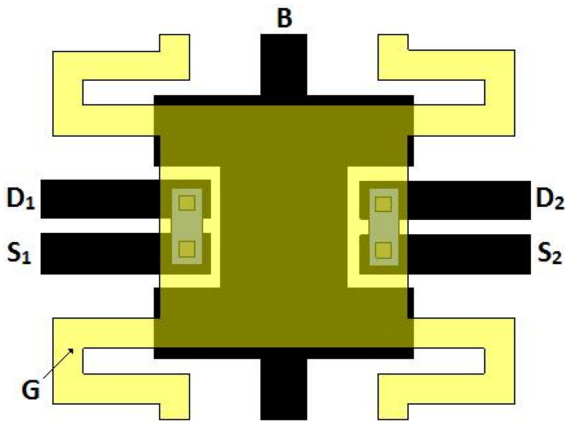


Figure 2-1: Diagram, operating states, and SEM image of standard 4-terminal relay device fabricated in a 1μm lithography process.



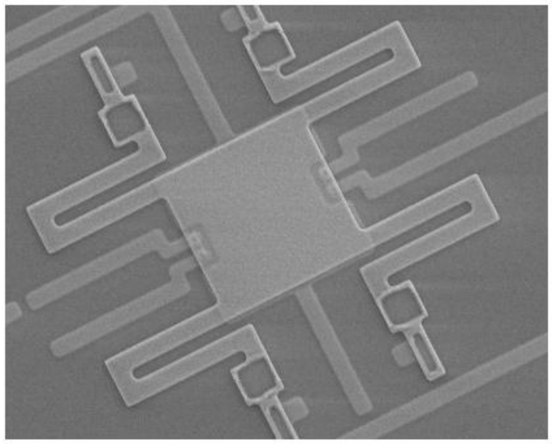
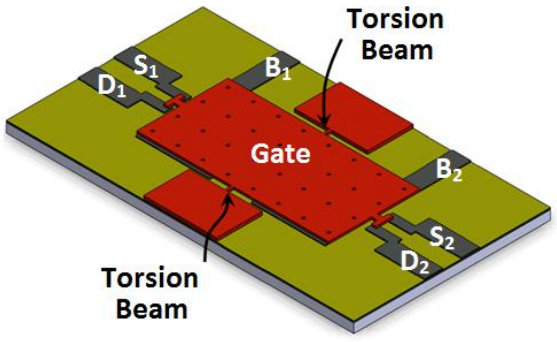


Figure 2-2: Diagram and SEM image of a 6-terminal relay device fabricated in a 0.25μm lithography process.*



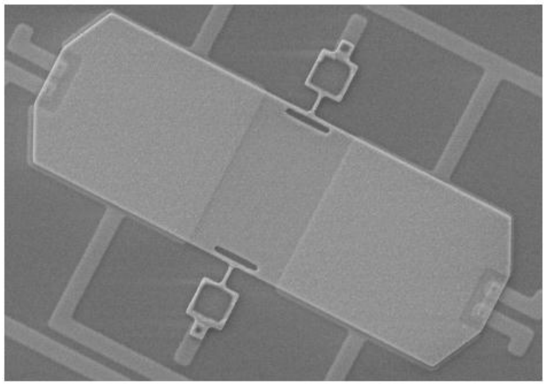


Figure 2-3: Diagram and SEM image of a seesaw relay device fabricated in a 0.25μm lithography process.*

*SEM images courtesy of Rhesa Nathanael and Jaeseok Jeon, UC Berkeley.

2.2 Circuit Behavior of MEM Relays

A summary of the circuit behavior of each of the three variations of MEM relays is shown in Figure 2-4. First let us examine the 4-terminal relay. There are two possible states of for this relay, *On* state and *Off* state. *On* state occurs when the gate and body terminals have opposite voltages and results in a short between the drain and source terminals. *Off* state occurs when the gate and body terminals have the same voltage and results in an open between the source and drain terminals. This behavior is very similar to traditional CMOS devices. In fact, by biasing the body terminal at *Gnd*, the device will be in the *On* state when the gate terminal has a high voltage, just like an NMOS device. Similarly, by biasing the body terminal at V_{DD}, behavior similar to a PMOS device can be achieved. However, unlike CMOS devices, the body terminal is a completely independent terminal and is not limited to biasing at reference voltages. Custom designs have taken advantage of this effect and sometimes have signals connected to both gate and body terminals of a single relay device to create XOR-like functionality.

The 6-terminal relay is fairly similar to the standard 4-terminal relay. As shown in Figure 2-4, the only difference is that there are two pairs of drain and source terminals that are shorted or opened as the relay transitions between *On* and *Off* states. One thing to note is that the circuit diagram shown in the figure does not mirror the actual layout of the device in the location of the source and drain terminals. This symbol is the standard for portraying the 6-terminal relay and allows for simpler circuit diagrams. Like in the 4-terminal variation the body terminal is not limited to be biased at referenced voltages, but when it is the device exhibits similar behavior to its CMOS counterpart. For example, if the body terminal is tied to *Gnd* the device behaves similarly to two NMOS devices with their gates shorted.

The seesaw relay, with its seven independent terminals and two independent body terminals allows varied functionality in circuit design. Custom designers have taken advantage of this flexibility and used clever biasing of the body terminals to create state elements. For combinational logic, the two body terminals are generally biased at opposite voltages. In this situation, the gate will always be actuated towards one side causing two possible states as shown. In this configuration this device behaves similarly to the 6-terminal relay except that the drain-source pairs are shorted alternately instead of simultaneously.

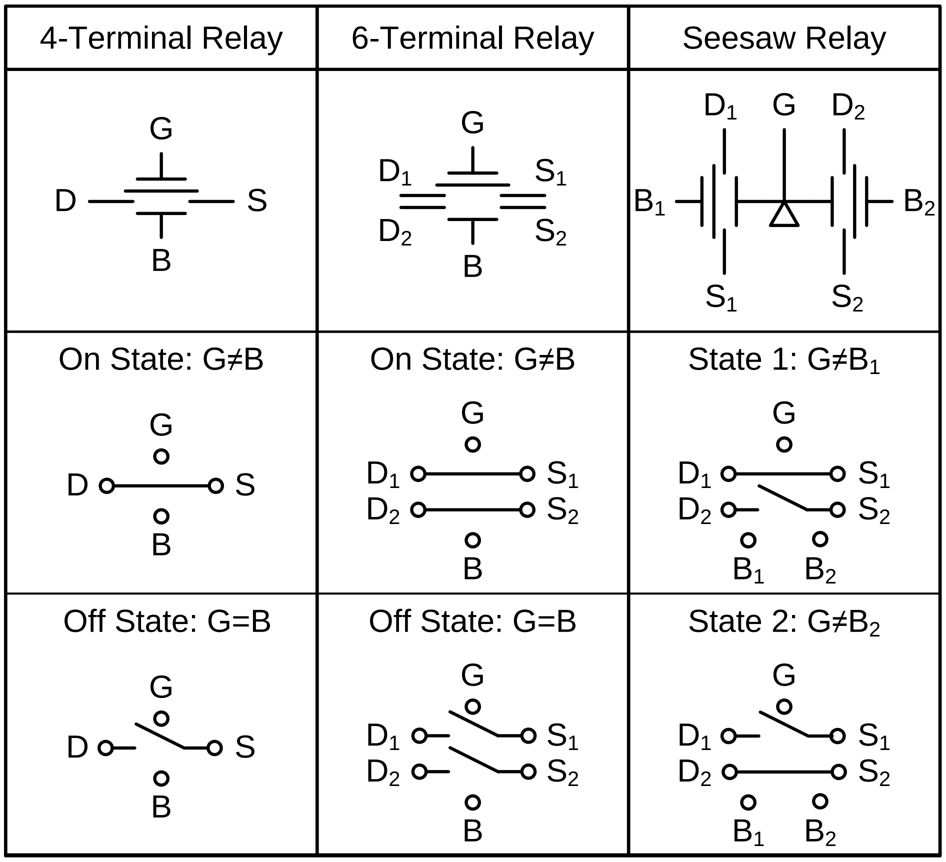


Figure 2-4: Summary of circuit behavior of the three MEM relay types.

With three different devices to choose from, each with many possible circuit configurations, a MEM relay circuit designer is faced with the challenging task of effectively utilizing each device to implement optimal designs without unnecessary extra relays. This requires significant design effort and generally leads to elegant, but unconventional designs. Another concern is circuit delay. At the device level the largest delay occurs when the device must change states. The circuit designer must balance these considerations to achieve a design with low area and delay.

Relay devices make excellent pass gates because of their ideal short and open behavior. Because signals are not degraded by V_{th} at each successive device, large stacks can be built without buffers. As shown in Figure 2-5, when the body terminal of each type of device is connected to V_{DD} or *Gnd,* they can be thought of as ideal pass gates between source and drain pairs with their passing behavior controlled entirely by their gates. The circuit design strategy presented in the subsequent chapters will utilize this pass-gate configuration. The abbreviated symbols shown in figure will be used to represent the full circuit symbol. The convention followed by the abbreviated symbols are that each circle represents a short between the drain and source pair when the gate is at *Gnd*. Each square represents a short when the gate is at V_{DD}.

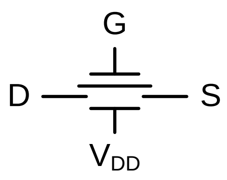
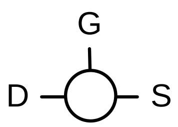
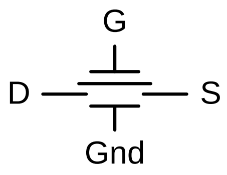
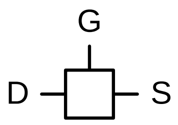
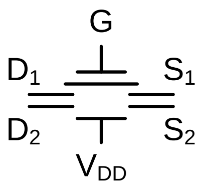
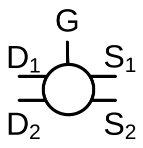
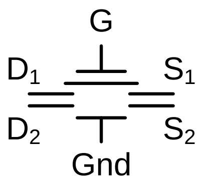
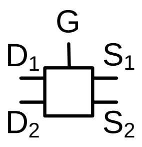
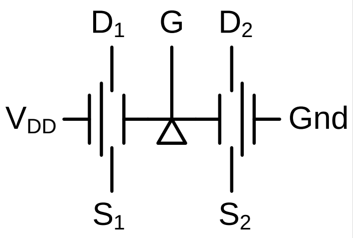
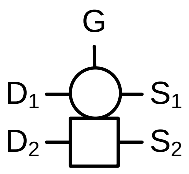
Circuit Symbol	Abbreviated Symbol
	
	
	
	
	

Figure 2-5: Abbreviated symbols for common relay device configurations.

2.3 Current State of MEM Relay Technology

Fabricated MEM relay chips have demonstrated the functionality of a variety of different circuits including multiplier components, flips flops, ADC’s, DAC’s oscillators, RAM modules, adders, and power gating circuits [6]. Although the technology is relatively young, integrated relay circuits have been demonstrated using a 1µm lithography process. According to predictions based on a relay operational model, further scaling down to the 90nm technology node is expected to give MEM relays an order of magnitude better energy-delay characteristics over CMOS at the expense of only three times the area [4, 5].

Previously designed circuits have used a completely custom approach and have thus been limited to relatively small and simple designs. Currently, the largest demonstrated working circuit is a 7:3 compressor which was implemented in 90 relays [8]. Relay devices differ from CMOS devices in that there is a mechanical part that moves. In comparison with CMOS, this mechanical movement makes a single relay device significantly slower. Therefore the custom approach has been necessary in order to mitigate this effect through custom circuit topologies. Even with this slower operation, the predictions regarding energy-delay characteristics at 90nm technology still hold largely because of these relay optimized circuit topologies.

The eventual target of MEM relay devices is to implement large-scale integrated digital systems such as microcontrollers. In order to show an improvement over CMOS, relay technology must first overcome the hurdles of area, performance, and design effort. The proposed synthesis tool is the logical next step towards this eventual goal as it offers a solution to each of these issues. By automating circuit design, less engineering effort will be needed to design larger, more complex circuits. Additionally by using a relay optimized circuit topology the slow actuation of relays can be mitigated. Finally, by optimizing for the number of devices needed to implement a given design, the issue of area can be systematically addressed.

CHAPTER III

Introduction to Relay Circuit Design

3.1 Discussion of Relay Design Topologies

As discussed in Chapter 2, a single MEM relay can behave similarly to an NMOS or PMOS transistor by having its body terminal biased appropriately. In the most basic case, this allows a relay implementation of any CMOS based circuit by a simple mapping of “NMOS” and “PMOS” style relays to their corresponding transistor counterparts as shown in Figure 3-1. The most commonly used CMOS design strategy is the standard-cell approach, where small simple gates such as NAND gates, NOR gates, and NOT gates are cascaded in multiple stages. Figure 3-2 shows one possible standard-cell based implementation of a 4-input AND function. This design can be realized with CMOS transistors as well as relays using a one to one mapping.

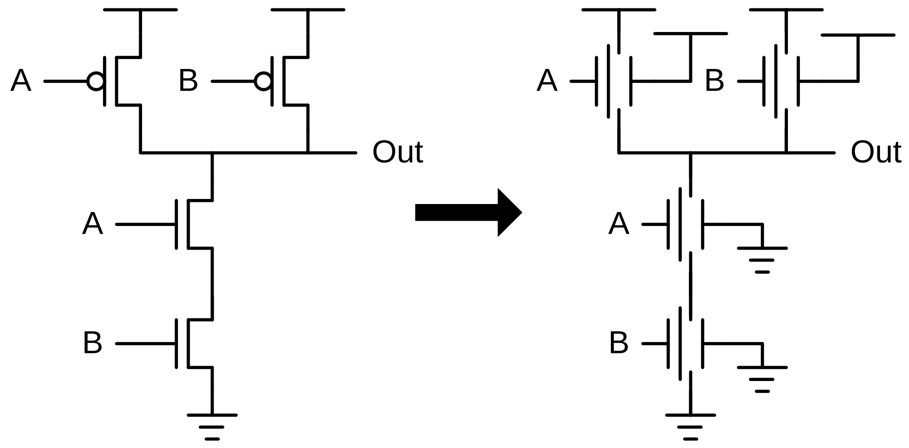


Figure 3-1: Implementation of a NAND gate in CMOS and relays.

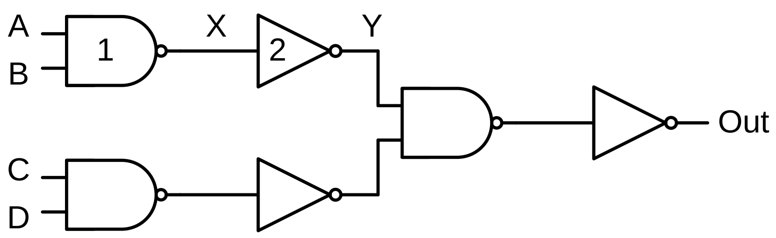


Figure 3-2: A cell based implementation of a 4-input AND function.

The performance of a MEM relay device is dependent on both its mechanical delay and its electrical delay. The mechanical delay refers to the amount of time it takes for the gate to physically actuate when the appropriate signal is applied to it. The electrical delay refers to the time it takes for current to flow through the device and charge or discharge a corresponding node. This differs from CMOS transistors where only the electrical delay is important in determining device performance. As shown in [4] there is a large disparity of over two orders of magnitude between the mechanical delay and electrical delay of MEM relay circuits. Because of this, the overall delay of a relay based circuit is dominated by its total number of mechanical delays and the electrical delay can be ignored when determining circuit performance [9].

A relay based design of a 4-input AND function using the cell based method shown in Figure 3-2 would result in a design with four mechanical delays. Let us take the top branch to be the critical path. Assuming inputs A and B arrive at the same time, all four relays in gate *1* will actuate simultaneously and a resulting signal will propagate to node *X* after one mechanical delay. At this time, both relays in gate *2* will then actuate simultaneously and the signal will propagate to node *Y* after a total of two mechanical delays. Each gate that is connected in series introduces an additional mechanical delay resulting in a total of 4 mechanical delays for the entire circuit.

Figure 3-3 shows an alternate relay based design for the 4-input AND function. Assuming all inputs arrive at the same time, all of the relays in the circuit will actuate simultaneously and the output will be determined after one mechanical delay. Additionally the alternate design requires only eight relays as opposed to 18 in the standard-cell approach. Therefore this design is better optimized for relay circuits. In fact, optimized relay based designs are designed so that all mechanical delays happen simultaneously in order to minimize overall delay. This circuit optimization is necessary to help mitigate the effect of the slow actuation time of MEM relays in comparison to CMOS.

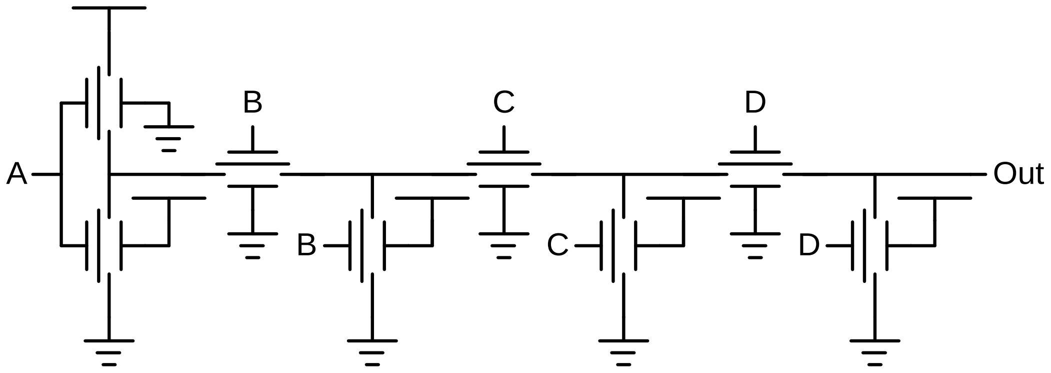
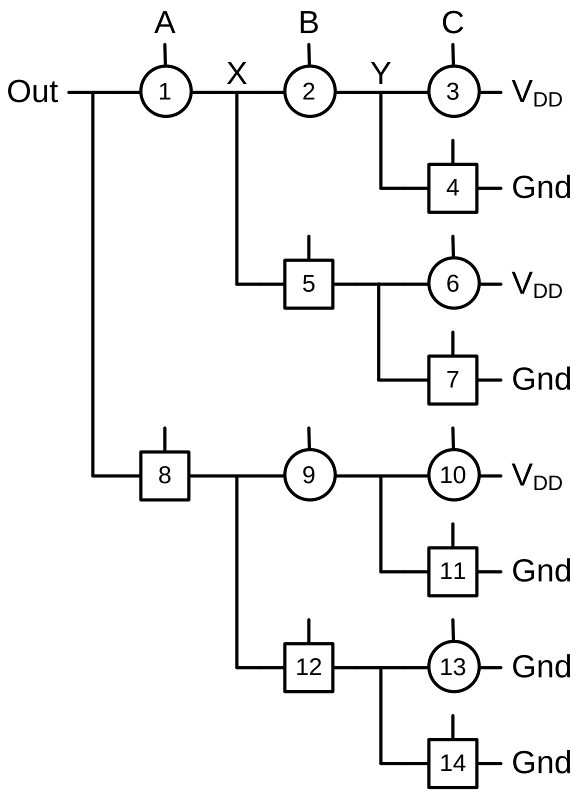


Figure 3-3: An optimized relay based implementation of a 4-input AND function.

3.2 A Binary Tree Method for Designing Relay Circuits

Currently, design of optimized relay circuits has been accomplished via a custom approach, which requires much time and effort from circuit designers working at a very low level of abstraction. Additionally, different circuit functionalities require unique topologies in order to achieve low delays. This requires circuit designers to become thoroughly familiar with every type of circuit and invent unique design strategies to achieve low delays across various designs. This results in widely different styles of relay based circuits, which are often unable to share subcomponents and which require long development time. The solution to this problem is a general method to design relay circuits with low delay that can implement any arbitrary logic function.

Relay circuits with one mechanical delay can be achieved by following one simple design constraint. For a relay circuit used to implement a given logic function, no gate or body terminals can be connected to any nodes other than inputs to the logic function, V_{DD}, or *Gnd*. Therefore intermediate nodes, which are nodes that are not input signals to the circuit, can only be connected to relay source and drain terminals. For the 4-input AND function, the inputs are the signals *A*, *B*, *C*, and *D*. In the cell based design shown in Figure 3-2, the intermediate nodes *X* and *Y* are connected to the gates of several relays resulting in a design with four mechanical delays. However, the alternate implementation shown in Figure 3-3 had all gate terminals connected to *A*, *B*, *C*, or *D* and all body terminals connected to V_{DD} or *Gnd* resulting in one mechanical delay. Another way to think about this constraint is to implement relay circuits as large complex gates rather than the cascade of smaller gates [5].



Out	A	B	C
1	0	0	0
0	0	0	1
1	0	1	0
0	0	1	1
1	1	0	0
0	1	0	1
0	1	1	0
0	1	1	1

Figure 3-4: A relay tree implementation of an arbitrary logic function. Note: each relay in a column under an input all have gate terminals connected to that input. For example, devices *1* and *8* both have their gates connected to input *A*.

An arbitrary logic function can be implemented in a binary tree relay circuit that follows this design constraint as shown in Figure 3-4. This circuit has inputs *A*, *B*, and *C* and output *Out* with the function shown in the truth table. In this example, each V_{DD} and *Gnd* connection shown at the right of the figure corresponds to one row of the truth table. With any combination of input signals, only one of these eight V_{DD} or *Gnd* signals will propagate to the output node. For example, if input *C* is *0*, then V_{DD} will propagate to node *Y* through device *3*. If input *B* is also *0*, then this V_{DD} will continue to propagate to node *X* through device *2*. Finally, if input *A* is also *0*, then the V_{DD} signal will propagate to the output *Out* node as a logic *1*. Therefore if inputs *A*, *B*, and *C* are all *0*, then the output of the gate will be *1*. If however, input *C* is now changed to a logic *1*, device *3* will turn off and the V_{DD} signal will no longer propagate to node *Y*. Instead, device *4* will turn on and the *Gnd* connection that is tied to it will propagate through nodes *Y* and *X* and finally to the output of the gate. Therefore input signals *A*, *B*, and *C* corresponding to inputs *0*, *0*, and *1* respectively would result in an output of *0*.

This example traced through the implementation and operation of the first two rows of the truth table. By connecting V_{DD} and *Gnd* signals to each device in the right column any output function based on these three inputs can be arbitrarily define. By extending the size of the binary tree, any output function based on any number of inputs can be defined. Since the gate terminal of every relay in the tree is tied to the input of the structure, all of the mechanical delays would occur simultaneously as the input signals arrive. This results in a single mechanical delay for the entire binary tree. In this structure, the number of relays through which a signal must propagate before reaching the output is equivalent to the number of inputs to the circuit. Therefore, as the number of inputs increases, the electrical delay of the circuit also increases. However, the stack size at which this electrical delay becomes significant is estimated to be around 200 as shown in [5] and is larger than is needed for any practical circuits. Therefore the binary tree method for designing relay circuits allows for the realization of any arbitrary logic function in a single mechanical delay.

CHAPTER IV

The Synthesis Algorithm

4.1 Synthesis Overview

The proposed synthesis tool aims to decrease the development time of MEM relay circuits and systems by automating the design of low-delay relay circuits able to implement any arbitrary logic function. The binary tree based method described in Chapter 3 provides the basic framework to achieve this, but design area becomes a concern. This is because as the number of inputs *N* to the design increases, the number of devices needed to implement the tree structure increases exponentially as $O(2^N)$. Specifically, it can be characterized by the following equation where *D* is the number of devices and *N* is the number of inputs:

$$D = 2^{N+1} - 2 \tag{4.1}$$

Another concern is that each tree can only generate a single output bit. Most basic building blocks of digital systems, such as adder circuits, require multiple bit outputs and would require multiple trees to implement. It is clear to see that as the number of input and output bits scale up for more complex designs, the number of relays needed to implement the design quickly rises to unmanageable levels.

The synthesis algorithm uses the binary tree method described above in combination with various optimization techniques to minimize the number of relays needed to implement each design. One way to do this is to recognize sections in the truth table where subsets of inputs yield the same result. Using a Karnaugh map based algorithm to simplify the truth table before generating the tree significantly decreases the device count. Another optimization is to recognize when certain sections of the tree generate the same logic and are thus redundant. In this case, a node sharing algorithm is used to find these instances and prune off the redundant sub-trees. This algorithm can be applied to sub-trees from different trees, further decreasing the device count across designs with multiple outputs. Finally, the tree structure of the relay circuits is highly compatible with using 6-terminal and seesaw relays, which can further decrease the device count of the design by approximately 50%. The following sections of this chapter cover these optimizations in further detail.

Currently the synthesis tool can automate the design of these tree based circuits as well as perform these optimization techniques resulting in circuits with device counts within two to three times that of custom designed circuits. It is not as powerful as commercial CMOS synthesis tools and the current version works best with smaller designs. The input to the synthesis flow is structural or behavioral verilog and the output is a relay level netlist in a structural verilog format. This can then be directly ported to conventional place and route tools. Figure 4-1 shows a high-level block diagram of the various steps and optimizations included in the overall synthesis flow.

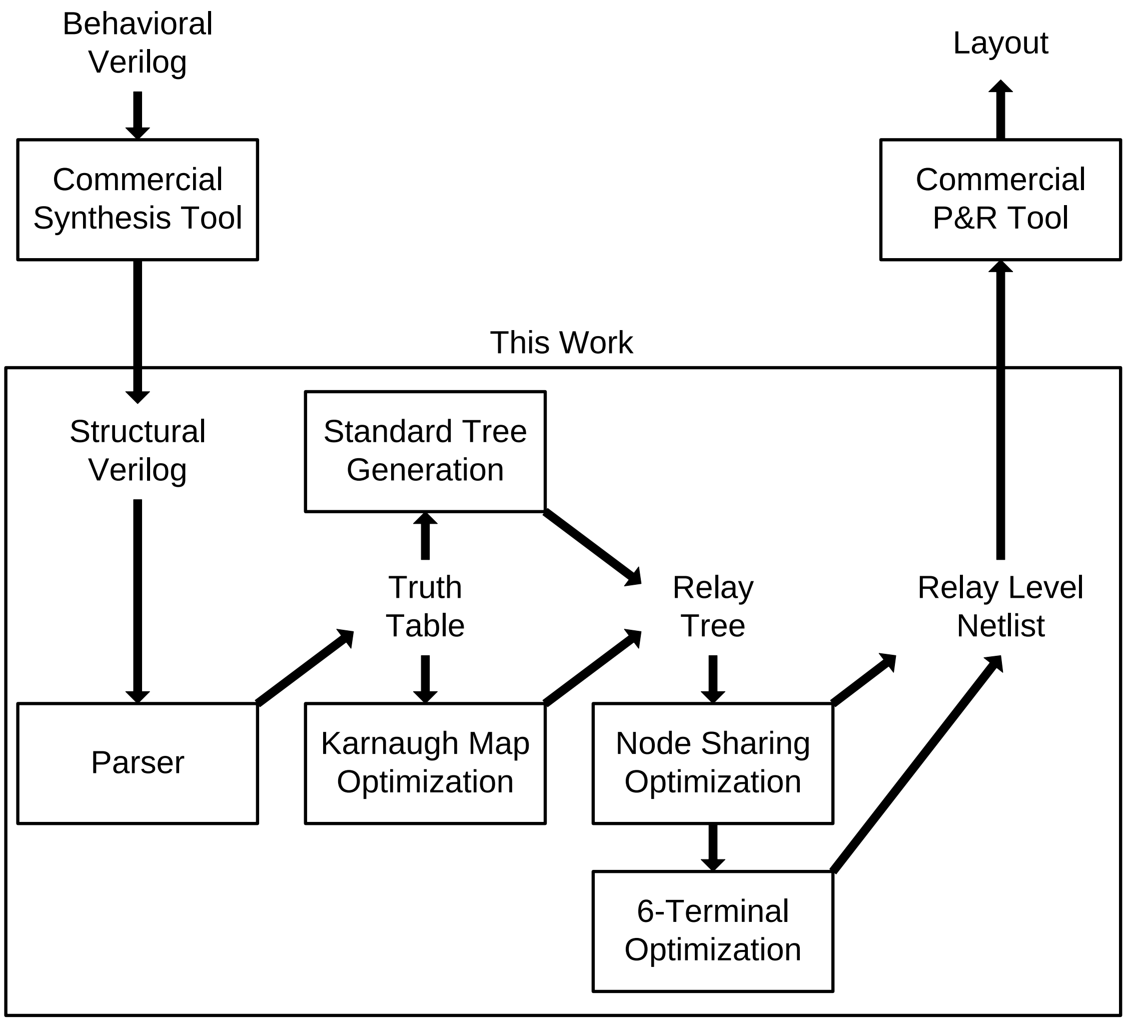


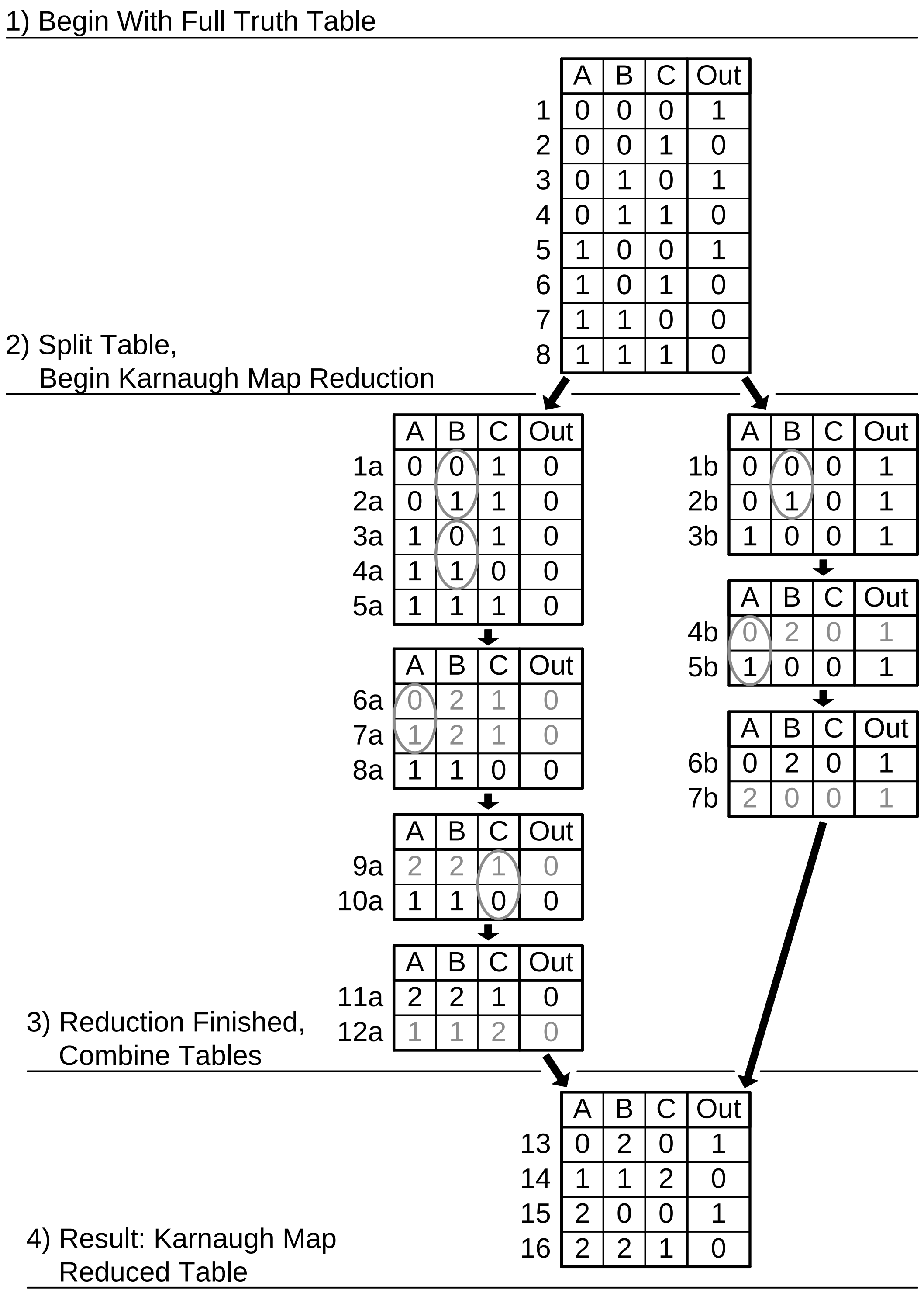
Figure 4-1: Block diagram of overall synthesis flow.

4.2 The Karnaugh Map Optimization

At the truth table portion of the synthesis flow, the tool can run the Karnaugh Map optimization if specified by the user. This optimization will simplify the truth table and use a different algorithm to map this modified truth table to a relay tree implementation.

If the user does not choose to run this optimization, the standard binary tree implementation of the truth table will be used instead.

This algorithm will be illustrated in an example as shown in Figure 4-2. For this example I will again implement the arbitrary logic function that was shown in Chapter 3. When two rows in the truth table differ by only one input, they can be simplified into a single row when the output of the two rows is the same. To implement this, the tool first sorts each row of the truth table into two smaller truth tables by the output resulting from that row. Then it examines the rows in each table looking for the case where two rows differ by only one input value. When a pair of rows is found that satisfies this condition they are combined into one row with a "2" used as a placeholder value replacing the differing input. This "2" means that for a specific input, either a *0* or *1* is acceptable. For example, rows 1 and 3 (or 1b and 2b) indicate that the logic function should output *1* when inputs *A*, *B*, and *C* are *0, 0, 0*, or *0, 1, 0*. Since both rows output *1*, and they only differ by their *B* input, then logically the output function should output *1* when inputs *A* and *C* are both *0*. The resulting optimized row 4b represents this exact function as *0, 2, 0, 1*. When applied fully to the truth table in Figure 4-2, it can be shown that in three iterations the entire eight-row truth table can be represented in four rows. This optimization is very similar to using the Karnaugh map method for simplifying Boolean algebra, except that the operations and output are represented in a truth table format.


1) Begin With Full Truth Table

	A	B	C	Out
1	0	0	0	1
2	0	0	1	0
3	0	1	0	1
4	0	1	1	0
5	1	0	0	1
6	1	0	1	0
7	1	1	0	0
8	1	1	1	0

2) Split Table,
Begin Karnaugh Map Reduction

	A	B	C	Out
1a	0	0	1	0
2a	0	1	1	0
3a	1	0	1	0
4a	1	1	0	0
5a	1	1	1	0

	A	B	C	Out
6a	0	2	1	0
7a	1	2	1	0
8a	1	1	0	0

	A	B	C	Out
9a	2	2	1	0
10a	1	1	0	0

	A	B	C	Out
11a	2	2	1	0
12a	1	1	2	0

	A	B	C	Out
1b	0	0	0	1
2b	0	1	0	1
3b	1	0	0	1

	A	B	C	Out
4b	0	2	0	1
5b	1	0	0	1

	A	B	C	Out
6b	0	2	0	1
7b	2	0	0	1

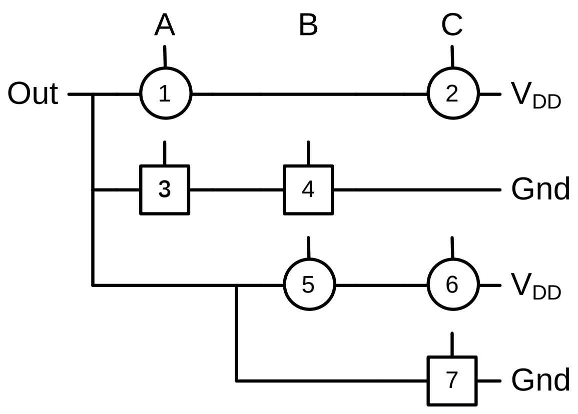
3) Reduction Finished,
Combine Tables

4) Result: Karnaugh Map
Reduced Table

	A	B	C	Out
13	0	2	0	1
14	1	1	2	0
15	2	0	0	1
16	2	2	1	0

Figure 4-2: An example of the Karnaugh map optimization.

Mapping the reduced truth table to a circuit implementation now requires a different algorithm as shown in Figure 4-3. Starting from the leftmost input, *A*, there are three possibilities for inputs in the truth table: *0*, *1*, and *2*. The *0* and *1* cases can be implemented with PMOS and NMOS style relays respectively in a manner similar to the binary tree method described earlier. The *2* case can be implemented with a short. At each node of the tree there are now three possible connections instead of two, but not all three may be needed in every case due to the reduced nature of the truth table. With the *A* input all three options are needed and the *Out* node includes a connection to a PMOS and an NMOS relay as well as a short to other portions of the tree. Moving further right to inputs *B* and *C* the result of the tree constructed in this manner is shown. There are four possible connections to the right of the *C* input that can propagate to the output of the tree. These correspond with the four rows of the reduced truth table. The result is a tree that requires a total of seven relays to implement instead of the original 14 from the binary tree method.



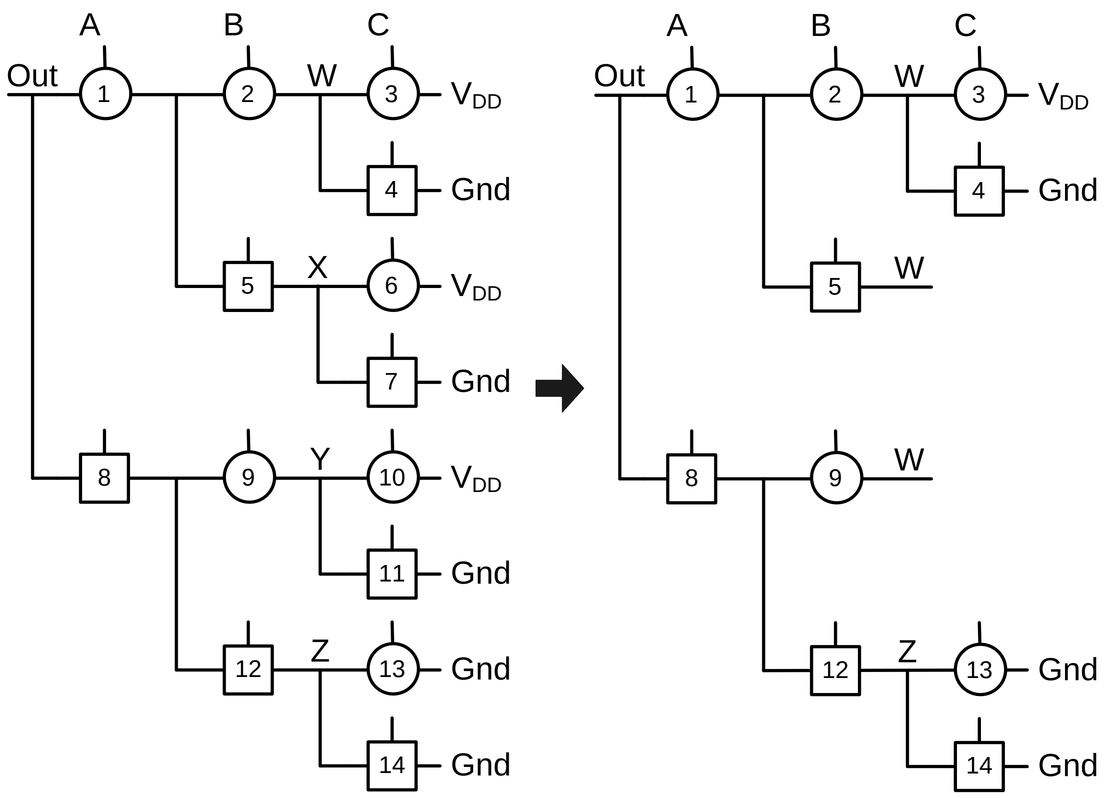
Out	A	B	C
1	0	2	0
0	1	1	2
1	2	0	0
0	2	2	1

Figure 4-3: A relay mapping example of a reduced truth table.

4.3 The Node Sharing Optimization

After the truth table stage, the synthesis flow will convert the design into a relay based tree circuit. The structure of the tree will vary depending on whether or not the Karnaugh map optimization was used. If the design has multiple output bits, a tree circuit will be generated for each. The node sharing algorithm operates on these tree circuits to identify redundant sub-trees and prune them from the design. In other words, it identifies parts of the circuit where the same logic function is being generated by identical relay structures in multiple areas and condenses them into a single structure.

To illustrate this algorithm I will perform the optimization on the resulting binary tree from Chapter 3 as shown in Figure 4-4. It is clear to see that devices *4*, *7*, *11*, and *14* all have very similar behavior. When input *C* is *1*, they all propagate the *Gnd* signal connected to their source terminals to their drain terminals. They appear to have redundant behavior and it is possible to consolidate some of them into a single device. However, before I do this I must take a look at the nodes which their signals are being propagated to: nodes *W*, *X*, *Y*, and *Z*. The logic functions represented by these nodes are shown in the truth tables at the bottom of Figure 4-4. I can see that node *Z* is different because device *13*, which is also connected to node *Z*, has different behavior than devices *3*, *6*, and *10*. Its source is connected to *Gnd* and not V_{DD}. Therefore in performing this algorithm I must check for nodes that have the same functionality rather than devices that pass the same signal. By finding these nodes I am indentifying identical sub-trees. In this case, the device pair of *3* and *4* comprises an equivalent sub-tree as the pair of *6* and *7* as well as the pair of *10* and *11*. These sub-trees can be consolidated into a single sub-tree with output of node *W*. The device pair of *13* and *14* comprises a different sub-tree because of its different behavior and will not be consolidated in the same way.



C	W, X, Y
0	1
1	0

C	Z
0	0
1	0

Figure 4-4: An example of the node sharing optimization.

The condition for making this consolidation at a specific node is that for each device with its drain terminal connected to the node (each device connected to the right of the node), there is a matching device with identical gate, source, and body connections on the node to be consolidated. In the example, node *X* can be consolidated with node *W*, because device *3* matches device *6* and device *4* matches device *7* in this manner. By implementing this simple rule, the synthesis tool is able to recursively consolidate large sub-trees without having to internally store the truth table for each of the internal nodes in the circuit.

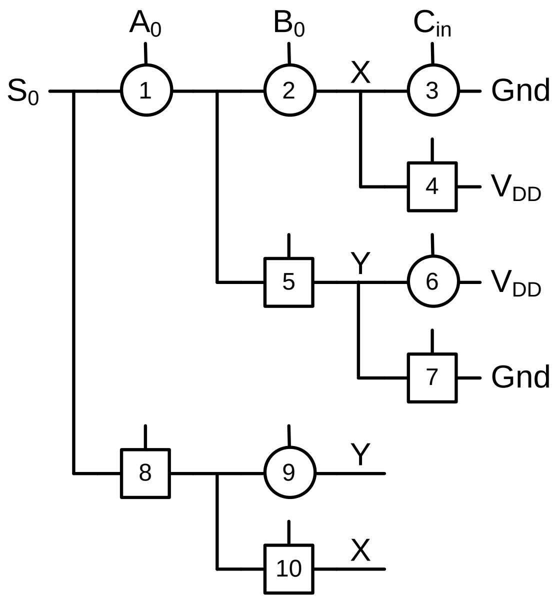


Figure 4-5: A node sharing optimized circuit for the S_0 bit of an adder.

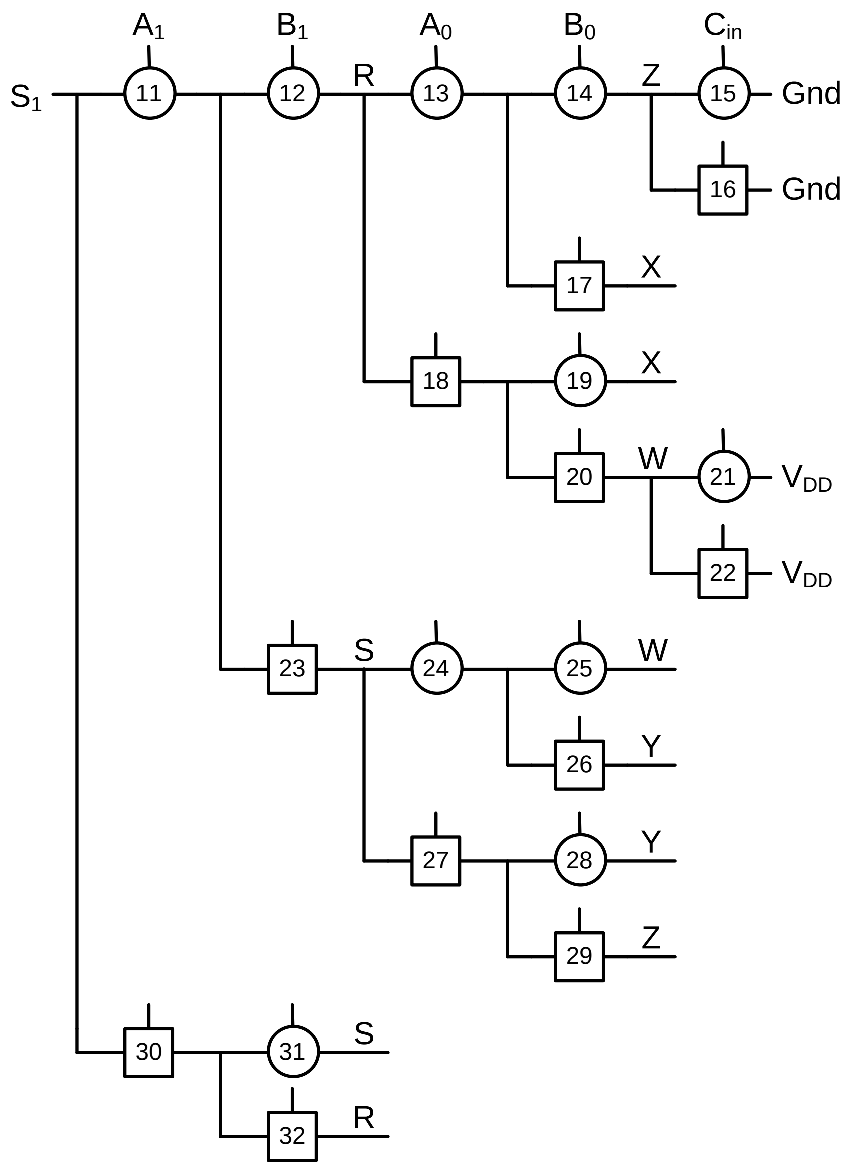


Figure 4-6: A node sharing optimized circuit for the S_1 bit of an adder.

This node sharing optimization is especially useful for decreasing device count in multiple bit designs. For such designs, each output bit is implemented by its own separate tree. These bits often have similar behavior and may share many identical sub-trees. This optimization allows for the sharing of sub-trees across these separate trees.

Since redundant sub-trees must have devices with identical gate, source, and body connections, the synthesis tool orders the inputs to each tree in the same way so that sub-trees will have matching gate connections. Figures 4-5 and 4-6 show the result of this optimization on a 2-bit adder. Notice the order of inputs to the two trees is identical. Since this design is more complex than the one in Figure 4-4, larger sub-trees can be shared such as those connected to nodes *R* and *S*. Sharing sub-trees across multiple trees is also illustrated in this example such as those connected to nodes *X* and *Y*. From Equation 4.1, the device count for the full binary tree implementation of this design can be calculated to be 76 relays. By implementing this optimization on just a simple two output design, a reduction down to 32 relays is shown. Even more drastic device count reductions result from running this optimization on more complex designs which will be discussed in Chapter 5. The sharing of sub-trees across different output bits is crucial in reducing the device count for circuits with multiple output bits.

4.4 The 6-Terminal and Seesaw Optimization

In addition to the source, drain, gate, and body terminals of the standard relay, the 6-terminal variation has an additional source and drain pair as described in Chapter 2. This allows two independent standard relays with identical gate and body connections to be implemented in a single 6-terminal relay. The seesaw variation of the relay can also be used in a similar way to replace the functionality of two standard relays with the same gate connections but opposite body biasing. If specified by the user, the synthesis tool can identify pairs of standard relays to be implemented as 6-terminal or seesaw relays, resulting in a design with all three types of relays.

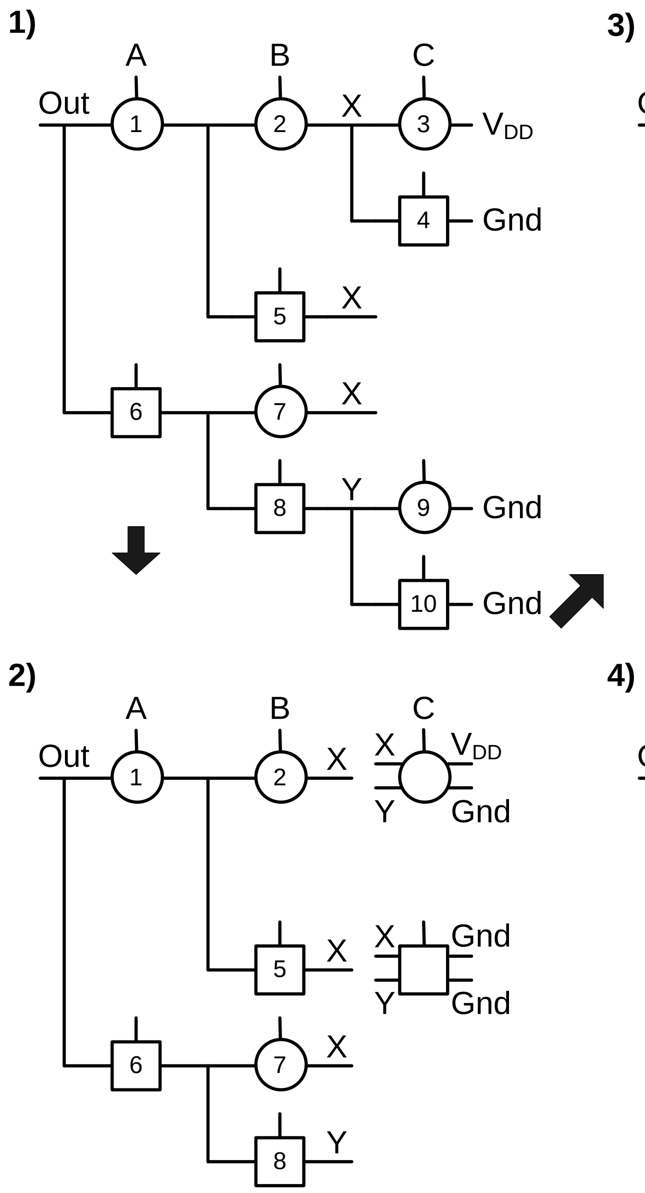


Figure 4-7: An example of the 6-terminal and seesaw optimization.

Figure 4-7 shows this optimization performed on the circuit resulting from the node sharing algorithm in Figure 4-4. Devices *3* and *9* both have their gate terminals connected to the signal *C* and their body terminals connected to V_{DD}. Therefore instead of using two 4-terminal relays they can be consolidated into a single 6-terminal relay with its gate connected to *C* and its body connected to V_{DD}. The source and drain connections of device *3* are mapped to one source and drain pair of the resulting 6-terminal relay while the source and drain connections of device *9* are mapped to other source-drain pair. Similarly, the device pairs of *4* and *10*, *2* and *7*, and *5* and *8* can also be consolidated in this manner as shown. At the third step in the figure, two 4-terminal relays which have the same gate connection but different body connections are left. Because of the differing body connections, they are not compatible with being consolidated into a 6-terminal relay, but are perfect for consolidation into a seesaw relay. Again the source and drain connections of the initial 4-terminal devices are mapped to the source-drain pairs of the resulting seesaw relay. The body connections of the initial 4-terminal devices must also be mapped along with their source-drain pairs. The result is a 50% reduction from 10 4-terminal relays to five total 6-terminal and seesaw relays.

The tree structures generated in the synthesis algorithm are well suited for this type of optimization because there are many relays with their gates tied to a common input. The body terminals of all relays are tied either to V_{DD} or *Gnd*. This means that there are many pairs of relays that have the same gate and body connections and which can be implemented by a single 6-terminal or seesaw relay. In fact when using 6-terminal and seesaw relays in the design, for each input, there will only be a maximum of one standard relay that cannot be replaced by an alternate relay and this occurs when an odd number of standard relays are connected to that input. For complex designs moving from a standard relay design to a 6-terminal and seesaw relay design results in close to a 50% reduction in device count. Since the 6-terminal relays have the exact same device footprint as the standard relays and the seesaws are very similar in area, this also translates into roughly a 50% reduction in area. The worst case scenario with the highest number of relays is characterized by the following equation where $D_{6\text{-}S}$ is the number of relays in the optimized 6-terminal and seesaw design, D_4 is the number of relays in the design before the optimization, and N is the number of inputs:

$$D_{6\text{-}S} = (D_4 + N)/2 \tag{4.2}$$

Let us take the circuit from Figure 4-7 as an example. Before the 6-terminal and seesaw optimization, this circuit was implemented with 10 4-terminal relays. The circuit has three inputs. Therefore substituting 10 for D_4 and three for N in Equation 4.2 yields $D_{6\text{-}S}$ of 6.5, which rounds up to 7. Therefore in the worst case scenario, a 10 relay circuit with three inputs will require 7 relays after the 6-terminal and seesaw optimization. In the example in Figure 4-7 only five relays were necessary. For more complex circuits the number of devices D_4 is much larger than N. Therefore $D_{6\text{-}S}$, the worst case implementation after the optimization, will be approximately half of D_4.

CHAPTER V

Synthesized Circuit Results and Discussion

5.1 Synthesized and Custom Adder Circuits

Whether they are custom designed or synthesized, MEM relay circuits are generally organized in large complex gates where the total delay of the circuit is dominated by a few mechanical delays. Since both synthesized and custom approaches target one mechanical delay circuits, the more interesting metric to examine is the number of relays required to implement a particular design.

First let us examine the difference between custom designed and synthesized adders. A custom designed 32-bit relay adder has been shown using the standard ripple carry style [4]. It is made up of full adder cells that are implemented with 12 relays each and results in one mechanical delay for the entire add operation regardless of adder size. The full adder cell from this design is reproduced in Figure 5-1. Using this style of adder design, the number of relays required to design smaller adders that can be used in comparison with the synthesis tool can be determined. As shown in the figure, complementary inputs are required for the C_{in} and A inputs. Generating these complementary inputs is not trivial as in CMOS with a simple inverter. This is because adding in an inverter stage adds an additional mechanical delay to the circuit. Therefore this type of adder would work best in a custom architecture that can supply these necessary inputs.

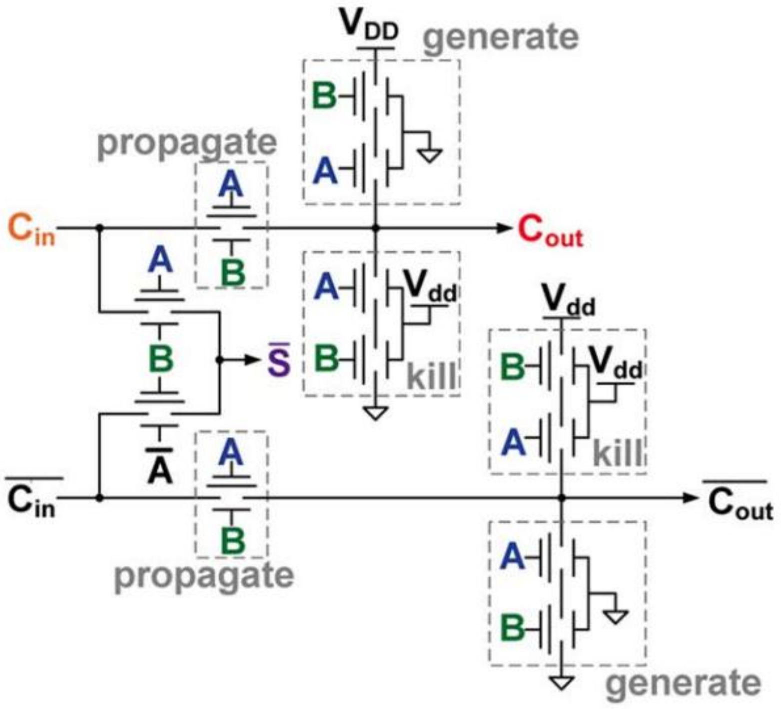


Figure 5-1: A custom relay implementation of a full adder cell for a ripple carry adder.

Like the circuits generated by the custom approach, the synthesized adders also require only one mechanical delay. Additionally these adders do not require complementary inputs and therefore are suitable for inclusion into a more generic architecture. The relay count of these adders is approximately twice that of the custom approach when only using 4-terminal relays. Examples of these adders are shown in Figures 4-5 and 4-6 from the previous chapter. However, after the optimization for 6-terminal and seesaw relays is performed, a 50% reduction in device count is achieved.

With this optimization the device count of the synthesized adders are very close to those of the custom adders.

Table 5-1 shows a summary of the device count of the various adders for varying numbers of input bits. Standard synthesis refers to synthesis with only 4-terminal relays. One thing to note is that the synthesized adders do not require complementary inputs whereas the custom adders do. The number of transistors required for a CMOS ripple carry adder based on the well known mirror adder cell [10] is shown as well for comparison.

TABLE 5-1. DEVICE COUNT OF VARIOUS ADDER CIRCUITS

Circuit Type	**Relay Synthesis**		**Relay Custom***	**CMOS**	**Bits**	
	Standard	***6T and SS***			***Input***	***Output***
2-Bit Adder	46	23	24	56	5	3
3-Bit Adder	72	36	36	84	7	4
4-Bit Adder	98	49	48	112	9	5
5-Bit Adder	124	62	60	140	11	6
6-Bit Adder	150	75	72	168	13	7

* Complementary inputs required for the custom adder.

5.2 Miscellaneous Synthesized Circuits

The 7400 series circuits represent important building blocks essential for building digital systems [11]. As shown in Table 5-2, four different circuits were synthesized including a 4-bit magnitude comparator, 4-bit ALU, 4-bit adder, and 4-bit carry-look-ahead generator. The number of devices required for these circuits was relatively modest with three of the four circuits requiring fewer devices than the 6-bit adder from Table 5-1. The 4-bit ALU, being the most complex of these circuits with the largest number of input and output bits, required the largest number of relays to implement. A three stage implementation is shown in the table as the functionality of this block can naturally be broken down into this implementation. Transistor counts for CMOS implementations of the designs are included as well for comparison purposes.

TABLE 5-2. DEVICE COUNT OF 7400 SERIES CIRCUITS

Circuit		**Relay Synthesis**		**CMOS**	**Bits**	
Name	***Function***	***Standard***	***6T and SS***		***Input***	***Output***
Circuit 74L85	4-Bit Mag. Comp.	130	65	144	11	3
Circuit 74181	4-bit ALU	295*	148*	260	14	8
Circuit 74283	4-Bit Adder	98	49	112	9	5
Circuit 74182	4-Bit PGK	88	44	64	9	5

*Results for a 3-stage implementation.

TABLE 5-3. DEVICE COUNT OF MISCELLANEOUS CIRCUITS

Circuit Type	**Relay Synthesis**		**CMOS**	**Bits**	
	Synthesis	***6T and SS***		***Input***	***Output***
4x4 Multiplier	384	192	384	8	8
7:3 Compressor	98	49	112	7	3
Instruction Decode	394	197	448	16	18

Table 5-3 shows the synthesis results for a few other circuits that are commonly used in digital systems. The instruction decode block is of special interest as it demonstrates another important role of the synthesis tool: synthesizing random logic. In digital systems, in addition to the many common parts that have been widely studied and optimized such as adders or ALU components, there are also irregular “glue-logic” circuits that allow these components to function properly together. When these circuits become a significant portion of the overall system design, optimizing them can be a tedious task for a custom design. The instruction decode circuit is an example of one of these circuits where optimized relay based topologies have not been developed. The synthesis tool provides a low-effort approach to generating optimized circuits that fall into this category.

The synthesis tool is not as powerful as commercialized CMOS style tools and currently can only synthesize small designs due to runtime constraints as will be discussed in Section 5.4. However, these circuits show that the synthesis tool can be used to generate the basic functional blocks needed to build more complex digital systems using very few devices, while meeting mechanical delay constraints.

5.3 Factors Influencing Device Count

The number of relays required to implement a design is dependent on the interplay between the circuit function and the number of input and output bits. As described in the synthesis flow in Chapter 4, each output bit is first individually constructed into its own reduced relay tree. Then common sub-trees between each of the output bits are shared among these separate trees. Therefore each synthesized design can be thought of as a collection of trees circuits, one for each output bit, with shared sub-trees.

In its unreduced form, each tree has a device count that scales with the numbers of inputs as described in Equation 4.1. Therefore one big factor determining the number of devices in the entire design is how many input bits are needed to fully characterize each output bit. Additionally, the tree can be reduced more significantly if it has a simple function which results in many rows in the truth table having the same output. This would result in a simpler reduced structure and would also increase the number of shared sub-trees. Therefore a second big factor in determining the device count is complexity of the function of each output bit. Finally, sub-tree sharing between trees of different output bits is most effective when the output bits have similar functions because this would result in more redundant sub-trees between separate trees. Thus the third main factor in affecting the device count is how similar the functions of each output bit are. In summary the factors influencing the device count of a design are:

1) The number of input bits needed to fully characterize each output bit.
2) The complexity of the function of each output bit.
3) The similarity between functions of each output bit.

Let us examine the 6-bit adder and the 4x4 multiplier to illustrate this point. The multiplier has fewer input bits and only one more output bit than the adder, but requires more than twice as many relays to synthesize. For the five most significant bits of the multiplier, each requires all of the input bits to characterize. This is only true for the two most significant bits of the adder. Also the adder circuit has a relatively simpler function that is similar between each output bit. However, each multiplier output bit is a complex function that must be uniquely generated from the input bits and thus relatively fewer sub-trees can be shared. Therefore the number of input and output bits help to serve as reference point on how many relays are needed for a design, but as illustrated in this example, the function of each circuit is also very important.

5.4 Discussion of Synthesis Run Time

A summary of the run time of each synthesized circuit is shown in Table 5-4. Like the device count, the run time is correlated to the number of input and output bits in the design. Unlike for the device count however, the function of each circuit, whether simple or complex, has relatively little effect. Specifically, the most important factor determining the run time is the number of input bits needed to fully characterize each output bit. This is because the number of input bits determines the size of the truth table and tree for each output bit. This creates the base number of nodes and devices that must be parsed through by the various optimizations algorithms. The number of output bits determines how many of these trees must be constructed. Building these large structures and running optimizations on them contributes significantly to the run time. From this

TABLE 5-4. RUN TIMES OF ALL SYNTHESIZED CIRCUITS

Circuit Type	**Run Time**	**Bits**	
		Input	***Output***
2-Bit Adder	< 5 minutes	5	3
3-Bit Adder	< 5 minutes	7	4
5-Bit Adder	20 minutes	11	6
6-Bit Adder	> 1 hour	13	7
Circuit 74L85	< 5 minutes	11	3
Circuit 74181	> 1 hour	14	8
Circuit 74283	< 5 minutes	9	5
Circuit 74182	< 5 minutes	9	5
4x4 Multiplier	< 5 minutes	8	8
7:3 Compressor	< 5 minutes	7	3
Instruction Decode	> 1 hour	16	18

discussion it is shown that the number of output bits has a linear effect on the run time while the number of input bits has an exponential effect as described in Equation 4.1. A further illustration of this exponential effect is shown in Figure 5-2, which plots the time required to synthesize an adder circuit as a function of the number of bits in the adder. This ignores all complexity factors and focuses on the relationship between input bits, output bits, and synthesis time. The exponential relationship between the input bits and the synthesis time dominates as illustrated.

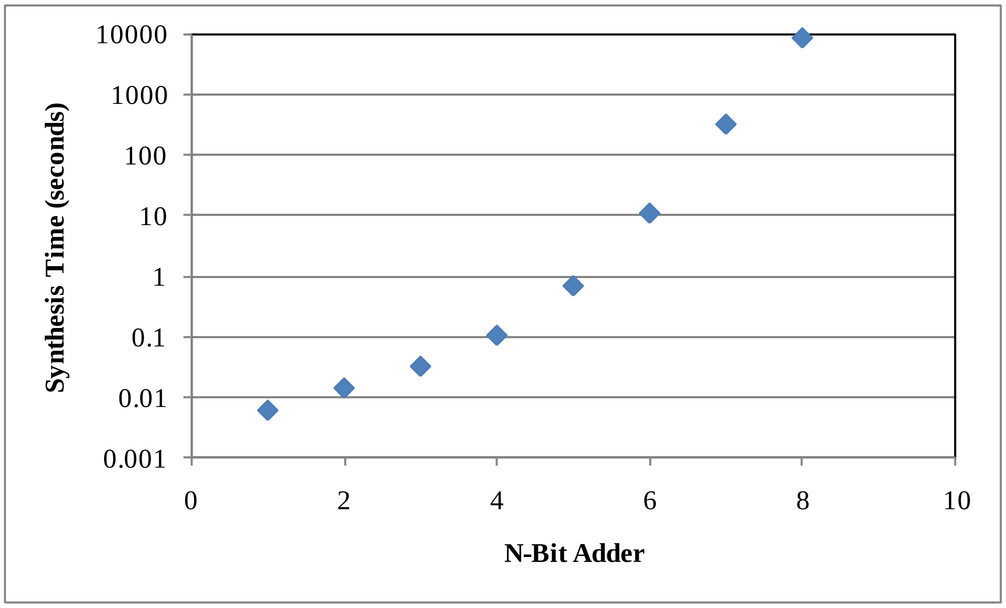


Figure 5-2: Plot of synthesis time as a function of adder size.

Currently run time considerations set the limitations on the size of synthesized circuits although, theoretically, any sized circuits can currently be synthesized. As shown, circuits with fewer than 10 inputs tend to synthesize in less than five minutes. Circuits with up to about 15 inputs can be synthesized in a moderate amount of a time greater than one hour.

5.5 Building Large Circuits

Because of the current limit on the size of synthesized circuits, larger circuits can be hierarchically constructed from smaller sub-blocks. For example, a 16x16 multiplier would require 32 inputs and would thus be too complex for the tool to synthesize. However, compressor blocks, such as the 7:3 compressor from Table 5-4, can be synthesized. Larger compressor blocks can be created as well, such as a 15:4 compressor or logic blocks representing a combination of compressor and adder functions. These can then be manually connected by the circuit designer to create the 16x16 multiplier. When the output of one block feeds into the input of another, a mechanical delay is introduced. Therefore the designer should take care to minimize this cascading behavior and maximize the instances of logic operating in parallel. This process does require some engineering effort, but since large logic blocks can be abstracted away, this effort is significantly less than a fully custom approach.

CHAPTER VI

Conclusion

6.1 Concluding Remarks

As MEM relay technology matures and continues to move toward scaled VLSI designs, the design effort required by custom designers will become increasingly heavy presenting an obstacle towards this end. The synthesis algorithm presents a logical step towards this goal by automating the design process and integrating seamlessly with automated place and route. Through an optimized and systematic approach, the synthesis tool has demonstrated circuits with the same performance as custom designs and comparable area. Additionally it has demonstrated performance and area optimized relay circuits for functions that have not been custom designed but which are important building blocks for VLSI systems. The synthesis tool also provides a method for easily incorporating the less conventional 6-terminal and seesaw relay variations in order to achieve an additional 50% area reduction.

6.2 Summary of Research Contributions

Below is a list of the major research contributions described in this thesis.

- Developed a general method for implementing arbitrary logic functions with MEM relays.

- Developed a systematic truth table simplification process along with a method to map to relay circuits in the Karnaugh map optimization.
- Developed a method for identifying identical sub-trees and sharing them throughout the design in the node sharing optimization.
- Developed a systematic approach to incorporating 6-terminal and seesaw relay variations into an existing 4-terminal relay circuit.
- Created a synthesis tool using these optimizations to automate the design of low delay circuits with low device counts.
- Demonstrated a variety of build block circuits necessary for large-scale integration.
- Discussed factors influencing device count and synthesis time.

6.3 Future Work

Potential future work in this field could involve extending the synthesis functionality to larger circuits. One possibility would be to automate the hierarchical design of these complex circuits using the resulting circuits from this tool as black boxes while minimizing overall delay. Another possibility would be to develop additional algorithms to enhance the current approach so that it can handle more complex circuits with lower device counts.

APPENDIX

Synthesis Tool Details

A working version of the synthesis tool has been implemented in Matlab. For the reader's reference, instructions on downloading and setting up the synthesis tool can be found at the following website:

http://icslwebs.ee.ucla.edu/dejan/researchwiki/images/c/c3/README_RelaySynthesis.txt

REFERENCES

[1] B.H. Calhoun, A. Wang, and A. Chandrakasan, "Modeling and sizing for minimum energy operation in subthreshold circuits," *IEEE J. Solid-State Circuits*, vol. 40, Sept. 2005, 1778-1786.

[2] H. Kam, T. King-Liu, E. Alon, and M. Horowitz, "Circuit-level requirements for MOSFET-replacement-devices," in *Proc. IEEE Int. Electron Device Meeting Tech. Dig.*, Dec. 2008, p. 473.

[3] R. Nathanael, V. Pott, H. Kam, J. Jeon, and T.-J. K. Liu, "4-terminal relay technology for complementary logic," in *Proc. IEEE Int. Electron Device Meeting Tech. Dig.*, Dec. 2009, pp. 223-226.

[4] M. Spencer, F. Chen, C. Wang, R. Nathanael, H. Fariborzi, A. Gupta, H. Kam, V. Pott, J. Jeon, T. King-Liu, D. Markovic, E. Alon, V. Stojanovic, "Demonstration of Integrated Micro-Electro-Mechanical Relay Circuits for VLSI Applications," *IEEE Journal of Solid-State Circuits*, vol. 46, no. 1, Jan. 2011, pp. 308-320.

[5] F. Chen, H. Kam, D. Markovic, T. King-Liu, V. Stojanovic, E. Alon, "Integrated circuit design with NEM relays," in *Proc. IEEE/ACM International Conference on Computer-Aided Design*, Nov. 2008, pp. 750-757.

[6] F. Chen, M. Spencer, R. Nathanael, C. Wang, H. Fariborzi, A. Gupta, H. Kam, V. Pott, J. Jeon, T. King-Liu, D. Markovic, V. Stojanovic, E. Alon, "Demonstration of integrated micro-electro-mechanical switch circuits for VLSI applications," in *Proc. IEEE Int. Solid-State Circuits Conf. Tech. Dig*., Feb. 2010, pp. 150-151.

[7] H. Kam, R. Nathanael, J. Jeon, E. Alon, and T.-J. K. Liu, "Design and reliability of micro-relay technology for zero-standby-power digital logic applications," in *Proc. IEEE Int. Electron Device Meeting Tech. Dig*., Dec. 2009, pp. 809-812.

[8] H. Fariborzi et al., "Design and Demonstration of Micro-Electro-Mechanical Relay Multipliers," unpublished.

[9] H. Kam, R. Nathanael, J. Jeon, E. Alon, and T. King-Liu, "Design and reliability of a micro-relay technology for zero-standby-power digital logic applications," in *Proc. IEEE Int. Electron Device Meeting Tech. Dig*., Dec. 2009, pp. 809-812.

[10] J.M. Rabaey, A. Chandrakasan, and B. Nikolic, Digital Integrated Circuits: A Design Perspective, Second Edition. Prentice Hall: Upper Saddle River, NJ 07458, 2003, p. 567.

[11] 74X-Series Circuits. [Online]. Available: http://www.eecs.umich.edu/~jhayes/iscas/