Copyright © 2002, by the author(s).
All rights reserved.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission.

DATA HANDLING CIRCUITRY FOR MASKLESS LITHOGRAPHY SYSTEMS

by

Ben Wild

Memorandum No. UCB/ERL M02/7

15 January 2002

DATA HANDLING CIRCUITRY FOR MASKLESS LITHOGRAPHY SYSTEMS

by

Ben Wild

Memorandum No. UCB/ERL M02/7

15 January 2002

ELECTRONICS RESEARCH LABORATORY

College of Engineering
University of California, Berkeley
94720

Data Handling Circuitry for Maskless Lithography Systems

By Ben Wild

Research Project

Submitted to the Department of Electrical Engineering and Computer Sciences, University of California at Berkeley, in partial satisfaction of the requirements for the degree of **Master of Science, Plan II.**

Approval for the Report and Comprehensive Examination:

Committee:

Professor Borivoje Nikolić
Research Advisor

(Date)

****

Professor William G. Oldham

Second Reader

(Date)

Table Of Contents

ABSTRACT 4

Chapter 1: Introduction 5

1.1 Future Challenges for Conventional Lithography 5
1.2 The Maskless Alternative 5
1.3 Problem Statement 6
1.4 Related Work 7
1.5 Thesis Organization 7
Chapter 2 – The Maskless Lithography System 9

2.1 Maskless Writing 9
2.2 Writing Scenarios 11
2.3 System Constraints 12
2.3.1 Redundancy 12
2.3.2 Optics 13
2.3.3 Grayscaling 13
2.3.4 Energy Source 14
2.4 Data Path 15
2.4.1 Targeted System 15
Chapter 3 Data Path Design for Maskless Lithography 17

3.1 System Architecture 17
3.2 Data Format 18
3.2.1 Binary Format 19
3.2.2 Thermometer Code Format 21
3.2.3 Binary Storage of Data 24
3.2.4 Thermometer Code Storage 24
3.3 Chip Architecture – Overview 25
3.3.1 I/O and Demultiplexing 25
3.3.2 Framing and Error Control 26
3.4 Decompression Datapath 27
3.5 Scaling Issues 29
3.5.1 Area 29
3.5.2 Power 29
3.5.3 Speed 30

3.5.4 Compression Needs 30
3.6 System Specifications 30
Chapter 4 Design of decompressor data path and mirror interface 32

4.1 Circuit Design Approach 32
4.2 SRAM based mirror interface circuitry 33
4.3 Decompressor Architecture 36
4.3.1 Huffman Decoder 37
4.3.2 Lempel-Ziv Decompression 38
4.3.2.1 Runlength Decoder 38
4.3.2.2 Systolic Architecture 39
4.4 Design of Decompression Circuitry 40
4.5 Alternative Memory Interface - Shift Register Based 42
4.6 Alternate Interface Circuitry : DRAM based 43
Chapter 5 – Prototype Implementation and Results 45

5.1 Chip Description 45
5.2 Estimated Performance of Test Chip 50
5.3 Estimated Performance of Future Design 51
5.3.1 Throughput 51
5.3.2 Area 51
5.3.3 Power 53
5.3.4 Further Optimizations for Power and Area Savings 54
Chapter 6 Conclusion 55

6.1 Key Accomplishments 55
6.2 Future Work 56
Acknowledgements 57

Appendix A: Simulation Setup and Results 58

A.1 Simulink Library 58
A.1.1 Matlab Verification 60
A.1.2 VHDL verification 60
A.2 Module Compiler Blocks 61
A.3 FIFO Interface Circuitry 62
Appendix B : Flip Flop Analysis 64

B.1 Trade-off Between Different Dynamic Storage Elements 64
B.2 Custom Designed SMEM Array 65
B.1.2 Pulse Triggered Latch Based Design Considerations 66
References 67

ABSTRACT

This work describes high throughput data handling for maskless lithography. Current optical lithography systems are able to transfer one layer of data from the mask to the entire wafer in about 60 seconds. To achieve this same figure with a maskless technique in the 50nm generation, we are required to transfer approximately 58Tb/s. We examine a system that uses compression and on-chip decompression to transfer the data to the writers. The system uses the Lempel-Ziv algorithm and the Huffman coding algorithm to achieve average compression ratios of 50 for layout data. The on-chip decompression consists of 1024 highly parallel independent decompression paths to maximize the throughput to the writers. We also present an SRAM interface to the writers targeted at a micro-mirror based system. A scaled down version of the chip is presented to demonstrate functionality. The chip consists of 8 parallel decompression paths, where each path consists of a Huffman decoder and a Lempel-Ziv decompressor. Error detection is accomplished using a CRC byte. From this scaled down version, we extrapolate performance figures for a full-scaled version of the chip.

CHAPTER 1: INTRODUCTION

1.1 FUTURE CHALLENGES FOR CONVENTIONAL LITHOGRAPHY

As feature sizes continue decreasing in accordance with Moore's Law, increasing complications are being seen in conventional lithography systems. As feature sizes decrease, the wavelength of the light used in the system also decreases. The expenses involved in creating masks at these feature sizes are close to $700,000 for a mask set in 0.13μm technology [1]. Any defect to the mask, or small design change would require the fabrication of a completely new mask set. The increasing costs of masks are shown in Figure 1. A lithography system capable of writing wafers without the need for a mask would be an attractive solution.

1.2 THE MASKLESS ALTERNATIVE

Currently, research is being conducted in a number of areas in maskless lithography. Two main areas of focus have been on mirror-based and electron beam (e-beam) direct write lithography. In a mirror-based system, an array of mirrors takes the place of a conventional mask, while in e-beam lithography an array of electron beam sources takes the place of the mask. In mirror-based maskless lithography each mirror is configured to project the mask data onto the wafer using a light source. The e-beam system, on the

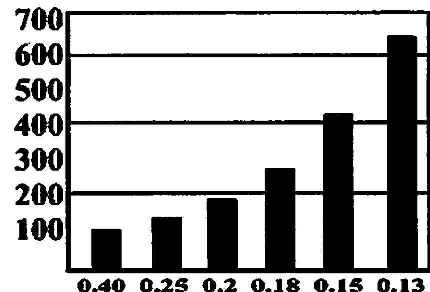


Figure 1 Mask Costs over generations, courtesy of Avanti Coorporation [1].

other hand, directly writes to the wafer using a directed beam of electrons. Both of these schemes require the storage of the wafer data on a storage device such as an array of hard disks and then transferring the data on the fly to the writers. By doing this, an electronic mask is created that never needs to be replaced, instead, the data on the hard disk would be replaced for different designs. One of the key challenges in a maskless lithography system is how to represent the wafer data, as well as transferring the data to the writers at a high enough rate to be competitive with conventional lithography systems.

1.3 Problem Statement

Conventional lithography machines are capable of transferring one layer of data from the mask to the wafer in around 60 seconds. To be competitive with these systems, a maskless system, using 25nm pixels for the 50nm generation must transfer on the order of Terabits/second of data to the writers. This requirement can be relaxed for certain applications such as low-volume ASIC manufacturing. In this work we will be focusing on the architecture of a system capable of meeting these throughputs. The system must interface to the writers, and the design is ideally constrained to being implemented on one chip to avoid mechanical alignment issues. To handle the large amounts of data, on-chip decompression must be used to alleviate the I/O limitations. Lempel-Ziv based decompressor architectures will be analyzed. The circuitry that directly interfaces to the writers is an important part of the system. We will examine interface circuitry for a micro-mirror based system where each mirror is 1 micron on a side. We also constrain the circuitry that interfaces to each mirror to require less than a square micron in area. This is to allow the mirror interface circuitry to be placed directly underneath the mirrors, avoiding routing long wires to the mirrors that would make it impossible to route. The system will be designed in 70nm technology and targeted for use in the 50nm generation.

1.4 RELATED WORK

Recently, a renewed interest in high-speed pattern generators has emerged. A new micro-mirror based pattern generator proposed by Micronics, uses a 2048x512 mirror array. The mirrors are 16μ on a side, coupled with condensing optics that reduces the image by 160 times. The system can write 100nm pixels. The mirrors are built on top of an address electrode grid to allow for the modulation of the mirrors. The mirrors are modulated with an analog voltage to deflect varying amounts of light to achieve pixel grayscaling. The many challenges of that design can be found in [3]. The system presently has throughput suitable for mask making, but is proposed for maskless lithography in the future [4]. Other companies such as Etec, have designed e-beam pattern generators for mask writing applications. The recently developed Etec MEBES Exara™, is mask writer for the 100nm generation. It uses a high-energy 50KV, thermal field emission gun as the energy source. The source is capable of writing at a rate of 320 Mpixels/second. More details of the Etec system can be found in [5]. Both of the above systems require several hours to write one mask.

1.5 THESIS ORGANIZATION

In this research project, we will be focusing our attention on the design of an integrated circuit that will interface to the writers. The chip should be able to transfer data to the writers at a high enough rate to be competitive with conventional lithography systems. Our goal is to design the data path for a system, targeted for the 50nm node, which will be able to write one layer, of a 300mm diameter wafer in 60 seconds. In Chapter 2 we will discuss the main components of a maskless lithography system. In Chapter 3 we will explore possible solutions for building a maskless lithography system capable of writing one layer of a wafer in 60 seconds. In Chapter 4

we will go over the details of the design of the main components of the mirror interface integrated circuit. A scaled down version of this chip has been designed, this design will be presented in Chapter 5, where we also present the implementation issues involved, along with projections for future designs. An explanation of the chip design flow and details of the simulations can be found in Appendix A.

CHAPTER 2 – THE MASKLESS LITHOGRAPHY SYSTEM

There are several ways of building maskless lithography systems. This chapter discusses two types of systems that can be attractive. These are micro-mirror based and e-beam based. Writing requirements for different applications are discussed and the specifications for the data path design are derived.

2.1 Maskless Writing

In a direct-write lithography system the component that exposes the wafer with the energy source is called the writer. To get a high throughput, many of these writers must operate in parallel. As discussed in Chapter 1, we can use mirrors coupled with an energy source or e-beams as our writers. Our research will focus on a mirror based direct write lithography system. Many of the components used in the data handling will be identical for both systems. The differences in the two designs will be in the interface circuitry to the writers. Figure 2 shows a typical e-beam based lithography system, and Figure 3 shows a typical mirror based system.

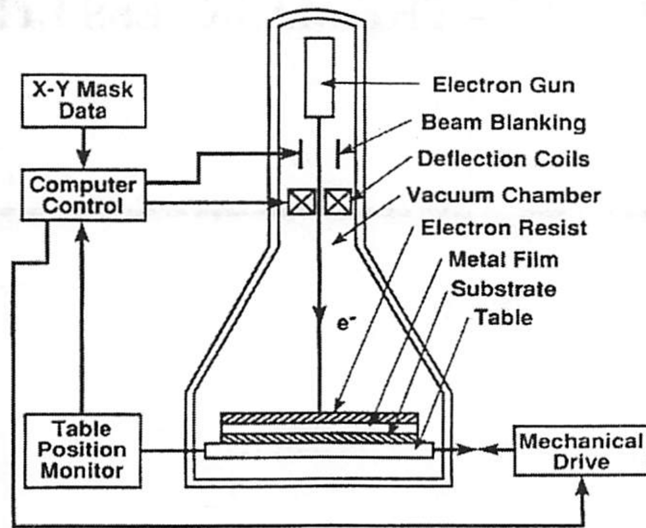


Figure 2 E-beam based lithography system [5].

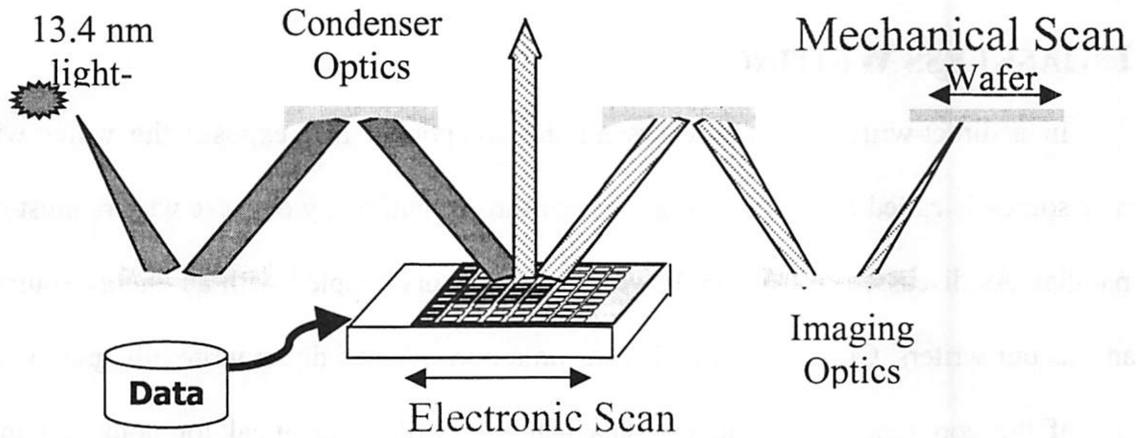


Figure 3 Micromirror based lithography system [6].

In Figure 2 an electron source is directed at the wafer. The electron beam develops the electron resist on the wafer. The pattern is written by mechanically scanning the wafer. The mask data is input to a computer, which is used to control the wafer position as well as deflecting the e-beam when no pixel needs to be written. In Figure 3 an EUV source is directed to an array of micro-mirrors. The mask pattern is loaded into the array, directly modulating the mirrors to reflect the mask pattern onto the wafer. The wafer stage is then mechanically scanned in the x-direction as new wafer data is shifted into the array. Condensing optics are used to reduce the image by the necessary factor.

2.2 WRITING SCENARIOS

Current conventional lithography systems are capable of writing one layer of a 300mm wafer in 60 seconds or less. Maskless systems should be competitive with these figures to be attractive for high volume manufacturing. We target this technique for the sub 50nm generation, which according to the ITRS will make the deployment of the system possible by 2008. This system can be used for several purposes. It can be used as a rapid prototyping tool, eliminating the long delay of getting masks back from the mask writing company. It can also be used for low volume ASIC manufacturers, where an expensive mask would dominate the non-recurring cost of manufacturing the chip. For these companies, a very high throughput may not be necessary, rather the savings in mask costs would allow for a lower throughput system to still be more economically viable. It has been predicted that low-volume ASICs will disappear due to mask costs. Ideally, the system should also be used for high volume ASICs, where throughput would be critical to be economically viable.

To calculate the data rate required for a 50nm system, we assume that we use 25nm pixels and use 31 bits/pixel to allow for grayscale. A 300 mm wafer then represents

$\pi * (150mm)^2 * \left[\frac{10^6 nm}{1mm}\right]^2 * \frac{1\,pixel}{25nm * 25nm} * 31 bits/pixel \approx 3472$ Tb of data. Thus, to expose one layer of an entire wafer in one minute requires a throughput of $\frac{3472\ Tb}{60\ s} \approx 58\ Tb/s$. Satisfying this throughput would allow writing 1 layer/minute and complete the replacement of mask-based lithography.

2.3 System Constraints

In the design of the data path for a maskless lithography system, we must consider several specifications of the system that will affect our design. These are redundancy, grayscaling, energy source technology used, and mirror technology. These are explained below.

2.3.1 Redundancy

To address the possibility of writer failure (micro-mirror or e-beam) it is important to be able to recover from these failures. By flashing a pixel on the wafer multiple times with different writers we can smooth out or eliminate the effects of writer failure. In our scheme we expose a pixel on the wafer with up to 31 different mirrors. We can expose the wafer with additional mirrors for an added level of redundancy. The level of redundancy needed depends on exposure requirements and the mirror failure rate. Calibration can be used to undo the effect of mirror failure. This can be done for example by switching all mirrors to the on position, exposing the array with a light source, then using a CCD to find out which mirrors are defective. Then we can find out which data pixels on the wafer are affected by these defective mirrors. Once this is computed, the data can be modified to undo the effect of the mirror failure.

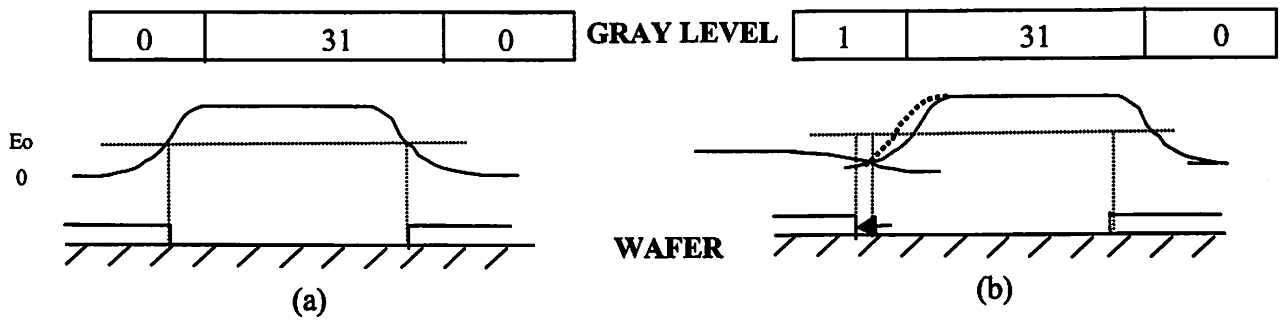


Figure 4 Edge Placement using Grayscaling

2.3.2 Optics

For a maskless lithography system using a micro-mirror based approach, the image data is projected onto the wafer by modulation of a mirror array. Prototype mirrors fabricated today are approximately 1μm on a side and are made of polysilicon, and will eventually be coated with EUV multi-layers [5]. To write 50nm features the image is passed through optics that reduce the image by a certain factor. In the system described in [3] the optics reduce the image by a factor of 160. We will design our system assuming that 1 square micron mirrors are used along with condensing optics with a reduction of 40, allowing for 25nm pixels to be written. The design of optics for EUV systems has many challenges that are outside of the scope of this thesis.

2.3.3 Grayscaling

The goal of grayscaling is to get finer edge placement resolution. The lithographic response of the photoresist represents the thickness of the photoresist removed for a given energy [8]. For simplicity, we assume that the response is rectangular, that is, none of the photoresist is developed if the source energy is under a certain threshold E_0. Once the energy on the wafer exceeds that threshold, then all of the photoresist is developed. The intensity profile of the source

energy projected on the wafer by a micro-mirror has a continuous, smoothly varying intensity as shown in Figure 4. In Figure 4, we have 3 pixels that are next to each other. We assume that the gray level of the pixel takes on a value between 0-31 to get sub-nanometer edge placement [2]. In Figure 4(a) only the middle pixel is exposed to the full gray level, 31. In Figure 4(b) the same pixel is exposed to a level of 31, and the adjacent pixel is exposed to a level of 1. The sum of the energies allows us to get finer edge placement as shown in the figure. We will also assume for the rest of this thesis that the mirrors can only modulate to two positions, the "on" position, where all of the light is reflected to the corresponding pixel on the wafer, and the "off" position, where no light is let through to the wafer. This is due to our area limitations imposed by the 1-micron mirrors; a simple circuit that outputs only two voltage levels is required.

2.3.4 Energy Source

All known EUV sources are pulsed, with pulse length in the 10-100ns range and frequency in the low KHz regime. We will assume that a 10KHz source is used. To calculate the required frequency for a single-chip solution, we will assume that the size of the mirror array grid on the chip is ~ 16,000x16,000. Using 5-bit gray, we are required to flash each pixel up to 31 times. We compute the required source frequency as follows:

$$Number\ of\ Pixels\ on\ 300mm\ Wafer = \pi * (150mm)^2 * \frac{(10^6\,nm)^2}{1mm} * \frac{1\ Pixel}{(25nm)^2} \approx 110\text{T pixels}$$

$$Flash\ Rate = \frac{110T\ Pixels}{60\ Seconds} * \frac{1 Flash}{16{,}000 * 16000\ Pixels} * 31\ Exposures / Pixel \approx 220\ KHz$$

The two possible scenarios that we are faced with are:

(a) A 220KHz source will be available to us. (Extremely unlikely)

(b) A ~10KHz $= f_s$ source will be the maximum source frequency available.

Case (b) is a subset of the data path design for case (a) with $\frac{220KHz}{f_s}$ chips.

2.4 DATA PATH

In any maskless lithography system the mask data must be transferred to the writers in an efficient manner. The data must be arranged in order to allow for grayscaling and redundancy. As will be shown, due to I/O bandwidth constraints, data compression must be incorporated to allow for a one-chip solution. The compression technique must be lossless, since we cannot afford to write incorrect features onto the wafer. The design of circuitry for this purpose is a complex issue, which will be discussed in the rest of this thesis. Some of the main issues to be addressed are:

- Throughput
- Incorporation of grayscaling and redundancy
- Framing of data and error detection
- Power, area, compression ratio and speed trade-offs

2.4.1 Targeted System

We will design the data path of the maskless lithography system targeted for use in general-purpose integrated circuit manufacturing. The system will be designed for use in the 50nm generation. The system will incorporate the following features:

- 1 layer/minute throughput for 300mm wafer.
- Single chip design in 70nm technology, with dimensions of 2cm X 2.5cm.
- 5-bit gray for sub-nanometer edge placement.
- Redundancy, by exposing a pixel on the wafer with up to 31 different mirrors.

- 16,384x16,368 mirror array. Each mirror can only be modulated to two positions, "on" where all of the light is passed to the wafer, and "off" where none of the light is passed to the wafer.
- Mirror interface circuitry that sits directly underneath the mirrors.
- Error detection, framing.

CHAPTER 3

DATA PATH DESIGN FOR MASKLESS LITHOGRAPHY

The system and chip architecture of our proposed design is presented below. We explain several possible data formats that can be used to achieve grayscaling. Once choosing the optimal format, we define the constraints of our system. The detailed specifications of our implementation is also presented.

3.1 SYSTEM ARCHITECTURE

In order to attain a high throughput to the writers, we propose a massively parallel architecture to tackle this problem. The block diagram of the system is shown in Figure 5.

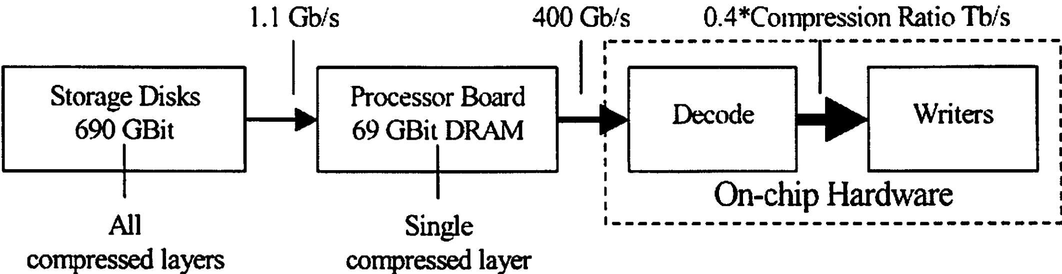


Figure 5 Block Diagram of System [2].

In the above model, the GDS layout file, which contains information about the geometries of the layout, is rasterized to 31 bits per pixel. The need for using 31 bits per pixel for grayscale is discussed in the next section. The data is then compressed and stored on a hard disk. Each layer of the wafer is then transferred to a processor board, capable of transferring the information at a high throughput to the chip. We anticipate from trends in high throughput network processor chips, that the maximum throughput into a chip will be around 400Gb/s [19]. This implies that to get a 58 Tb/s on-chip throughput, we are required to compression the data by a factor of $\frac{58\,Tb/s}{400Gb/s} \approx 145$. We can fit approximately $\frac{\pi*(150mm)^2}{20mm*10mm} \approx 350$, 20mm X 10mm chips on a 300mm wafer. In terms of uncompressed pixels, a chip layout then represents about $\frac{110T\ pixels}{350\ chips}*\frac{31\ bits}{1\ pixel} \approx 10$ Tb of information. Assuming we can compress the data by a factor of 145, the entire chip layout becomes $\frac{10\,Tb}{145} \approx 69\,Gb$ in size, and can be stored on multiple Rambus DRAM chips on the processor board. These RDRAM chips must output 400 Gb/s, which could be accomplished, for example, with 32 RDRAM chips each 16-bits wide operating at 800 MHz

3.2 DATA FORMAT

The goal of the chip is to transfer data to the array of mirrors, where the mirror array will be integrated on the chip. One requirement of the design is that the intensities of the reflected source energy must be grayscaled if necessary. The additional requirement is that it must somehow incorporate redundancy. We also assume that the mirrors can only modulate to two positions, "on" and "off" as explained in Chapter 2. Furthermore, we will design the system to

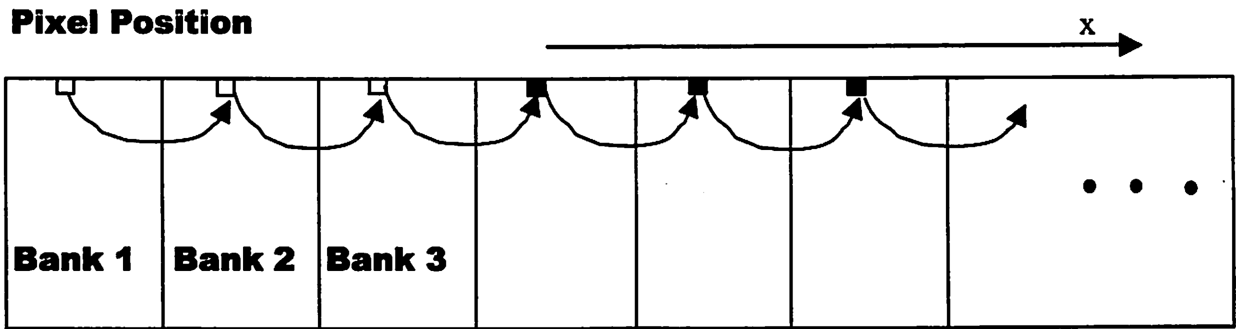


Figure 6 Pixel Propagation for Grayscaling and Redundancy

have 32 levels of gray. Once the data has been decompressed, the question of distributing the data to the mirror array arises. As shown below, the data format will determine the architecture for the circuitry that will transport the data from the decompressor to the mirror array. Two data formats that we will analyze to accomplish the grayscaling are binary and thermometer representation. For convenience, we will assume that a 16,384 x 16,368 mirror array is used. To make the discussion in the following sections more clear, we will define a bank of the mirror array as 528 columns of the array. The number of columns in one bank is obtained by dividing the number of mirrors in a row by the number of non-zero exposure levels. In the case of 5-bit gray, the number of non-zero exposure levels is 31. We can then enumerate the banks from 1 to 31, where bank 1 is the bank closest to the decompression circuitry.

3.2.1 Binary Format

In this format, 5 bits of information are transmitted per pixel, representing the 31 different, non-zero exposure amounts that the pixel can have. We also transmit an additional bit, which modulates the mirrors based on the 5-bit value. If we did not need to incorporate redundancy into this design, then we could have a 5-bit down counter and a comparator sit underneath each mirror. The 5-bit value would be loaded into the counter, and the comparator

output would connect to the mirror electrode. After each flash, the counter would be decremented, and once the counter hit zero, the comparator would output a '0' to the electrode. If for example a pixel had a 5-bit value equal to "00011", then the sequence of control signals that the mirror requires is "111000...0". These sequences of bits are the thermometer code value of the binary number. Instead of having a counter and comparator circuit sitting underneath the mirror, we could instead load this thermometer sequence of bits directly. The disadvantage is the data expansion of 31/5. Thermometer coding is discussed in more detail in the next section.

To allow for redundancy, the wafer stage moves along the positive x-direction to a new position, such that all of the pixels on the wafer get exposed by different mirrors. This procedure is repeated for 31 times, so that each pixel on the wafer can be exposed up to 31 times to get the grayscale effect. Redundancy is obtained by exposing the same pixel multiple times by different mirrors. As the wafer stage moves in the x-direction, the pixels propagate to the new mirror position at the same rate. In this case, the mirror array can be thought of as 31 banks of mirrors, the pixel propagates to each of these banks and inform the corresponding pixel whether it should be "on" or "off", this is illustrated in Figure 6. Figure 6 illustrates the case of a level 3/31 gray pixel propagating through the array. The small squares represent the mirrors that the pixel propagates to; a dark mirror implies that the mirror is in the "off" position. Note that in this case, there must be circuitry underneath the mirrors to rotate the mirror, reflecting the light source away from the wafer. In this case, the writer interface circuitry must propagate the 5 bits of data for each pixel. As the pixel propagates along the banks, a comparator circuit must be implemented in order to turn off the pixel. In the case shown in the figure, a comparator between the 3rd and 4th bank would turn off the pixel.

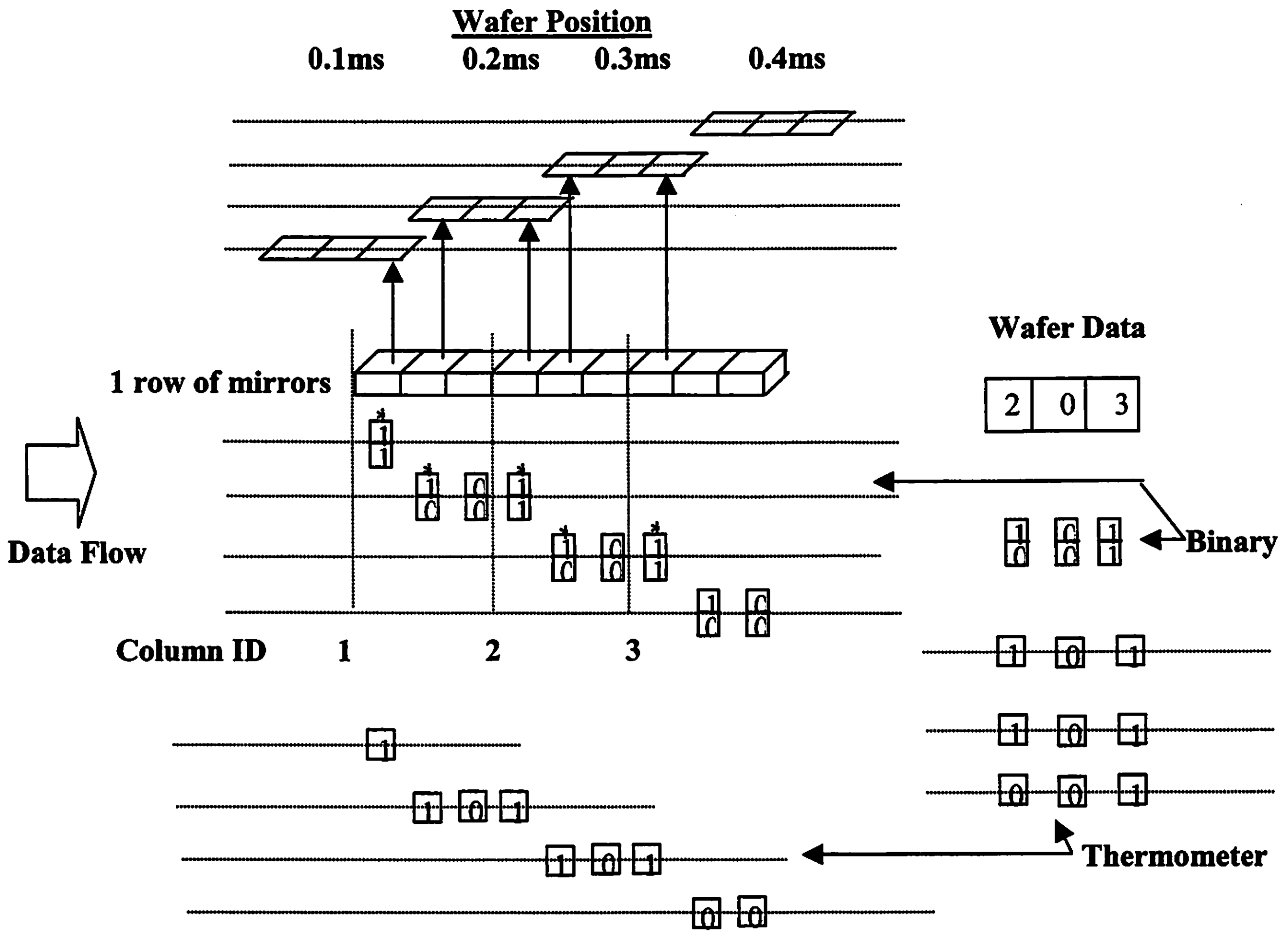


Figure 7 Thermometer vs. Binary Data Format.

3.2.2 Thermometer Code Format

Instead of the data transmitted to the chip in 5 bit pixel groups, we can rearrange the data into thermometer code format, where the 5 bit gray value is expanded into the 31 bit value with the number of 1s corresponding to the gray level of the pixel. In this format an image is broken up into 31 frames. Therefore we load one new frame after every flash, rather than propagating the pixel along the scan line. The advantage of this scheme is that it requires only 1 storage element per pixel instead of 6 for the binary, shift register scheme, since we do not need circuitry to propagate the pixel and comparators to control the turning off of the pixel.

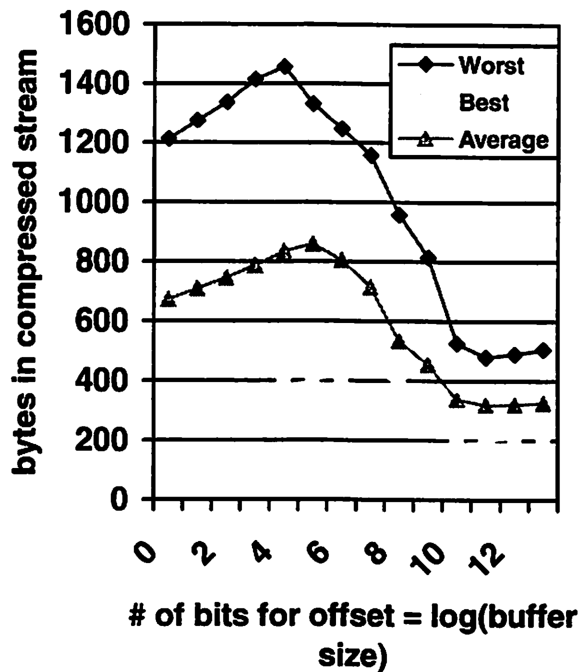


Figure 8 Compression Stream Size Versus Buffer Length.

We can also use a memory array as the writer interface circuitry, which is more power efficient than a shift register. This is because in a shift register architecture, to get a pixel from the decompressor to a mirror that is *n* mirrors down the row requires *n* clock cycles, assuming a synchronous shift register. With a memory we can directly load the memory cell underneath that mirror in one cycle. Since we required 6 storage elements per mirror in the binary case, we see that this data format should save us close to 6 times in area. The disadvantage is that instead of using 5 bits per pixel, we must use 31 bits. The data expansion implies that the throughput to the mirrors with this format will be 58.3 Tb/s as was shown in section 2.2. A Binary format would have required $58\ \text{Tb/s} * 5/31 \approx 9.4\text{Tb/s}$. To achieve the same data rate to the mirrors, the writer interface circuitry must operate at 31 times the previous rate. The reason for this is that we are reloading the whole mirror array after every flash, where in the binary scheme we shifted in 1/31 of the frame after every flash. Figure 7 illustrates the difference between the two schemes. The figure shows one row of mirrors with 9 elements. Three pixels of the wafer are shown as they

move along the x-direction. In the binary case, the pixels (2,0,3) are transmitted in parallel, 2 bits at a time, along with a '1' bit which is the control bit that interfaces to the mirror. The column ID of the rows is compared to the 2-bit pixel value and the control bit is flipped to a '0' if the column ID matches the pixel value. In the thermometer code format, the gray values (2,0,3) are expanded into 3 frames: (1,0,1), (1,0,1), (0,0,1). These frames are then simply loaded into the mirror interface circuitry, one frame at a time. The frames are also shifted in time to allow for redundancy as shown in the figure. In Figure 7, we see another problem with using thermometer code format. In binary format, two pixels that are next to each on the wafer image will be input sequentially. In thermometer format, the data is rearranged so that two pixels that were adjacent to each other on the image will no longer be adjacent to each other when decompressed. This is because we are refreshing the whole mirror array after every flash, implying that the pixel takes several frames to completely expose. Since layout data can be strongly correlated, by spacing the data further apart we are reducing the correlation of the data and thus will not be able to get an increase in the compression ratio proportional to the increase in the data throughput. To combat the information expansion, it is possible to increase the complexity of the compression algorithm used. In the case of the Lempel-Ziv algorithm, which is discussed in section 4.3.2, increasing the size of the history lookup buffer allows us to attain a larger compression ratio. This result is shown in Figure 8, where the size of the compressed stream is plotted versus buffer length. This is for a frame size of 16,000 bytes, using thermometer data format. The figure shows that a history buffer size of $2^{10} = 1024$ achieves a compression ratio of approximately 30-80 with an average of 50. The results are obtained by averaging the compression ratio of the active layer of a typical layout. 5 random regions of the layout were used to get the results [17].

3.2.3 Binary Storage of Data

In the binary scheme, 1/31 of an image is transferred in after every flash one bank at a time. As explained earlier, the pixels are transferred through the rows of mirrors 6 bits at a time. 1 row is used to transfer the control bit for the pixel, and the other 5 rows are used to determine when to turn off the control bit as it propagates down the row of mirrors. 31 compare circuits are needed to decide when to turn off the control bit. We assume that the compare circuits are placed before each bank of mirrors, and that 00000 implies keeping the pixel "on" for all the rows, while 11111 implies turning "off" the pixel immediately before the 1st column. The required hardware per pixel is then approximately 6 storage elements. For a row with 16,000 mirrors we then need 16,000*6 = 96,000 storage elements.

3.2.4 Thermometer Code Storage

In this scheme we only need 1 storage element per pixel since the pixel data is simply loaded into the array. We need 16,000 storage elements for a 16,000-mirror row. For our implementation we will be using this data format. This decision is based on the projection that in 70nm technology we will be able to fit around 10-15 transistors in a square micron [10], preventing us from using a binary format, where we would be required to use 6 storage elements which translates to about 36 transistors, assuming 6 transistors/element. We can also scale a 6T SRAM cell down to approximately 0.5μm^2 in 70nm, making an SRAM memory interface attractive.

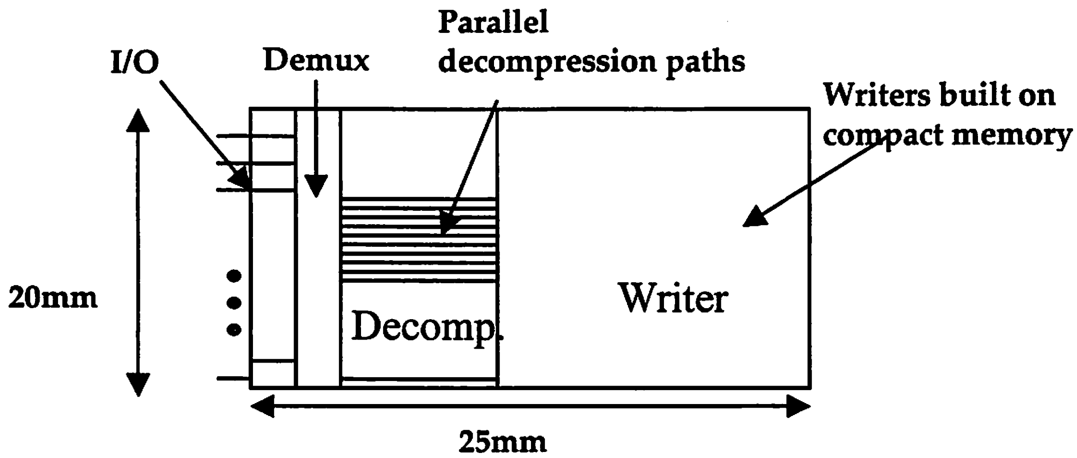


Figure 9 Mirror interface chip.

3.3 CHIP ARCHITECTURE – OVERVIEW

The proposed mirror-interface chip is shown in more detail in Figure 9. The figure shows the data entering on the left side of the chip using high speed I/O. The data is then demultiplexed to a lower data rate, and sent to multiple parallel decompressor paths. Once decompressed, the data is loaded into the mirror interface circuitry. For our implementation, we will be using a thermometer code data format, allowing for one storage element underneath each mirror. We will be using a memory array for the mirror interface. The mirrors sit directly on top of a memory array, where each mirror sits on top of a memory cell. To implement the writer-interface memory such that it is not directly underneath the mirrors would mean routing approximately 16,000x16,000 = 256 Million wires to the mirrors which would be impossible.

3.3.1 I/O and Demultiplexing

A trade-off exists between the number of I/O pads used, and the speed of the I/O. We envision using 128, 3.125 Gb/s high speed input lines to get the necessary throughput. The pads would be placed as close as possible to the input of the decompressor paths to avoid crosstalk

among the lines. Since the decompressor paths will be operating at a fraction of the I/O speed, the data must be demultiplexed by the ratio of the I/O input throughput to the input data rate of the decompressor paths. The number of parallel decompressor paths depends on the maximum output throughput of each decompressor. The number of paths is obtained by: $number\ of\ paths = \frac{Required\ throughput\ to\ mirrors}{Output\ Throughput\ /\ path}$. The output data from the decompressor paths must then be de-multiplexed to the rows of the mirror interface memory. This factor is then simply the number of rows in the mirror array divided by the number of parallel decompression paths. In our implementation we use 1024 decompressor paths, so that the demultiplexing factor is 16,384/1024 = 16.

3.3.2 Framing and Error Control

In a maskless lithography data handling architecture, it is important to detect any errors that may occur during the data transfer. Furthermore, at high data rates, the synchronization of data becomes a challenging task. To allow for synchronization and error detection, we frame the data into the following format shown in Figure 10, where n is the number of mirrors in a row.

8 Start Bytes	n Data Bytes	1 CRC Byte

Figure 10: Frame Format.

When the system detects the start bytes it resets the writer interface address counter, synchronizing the system. The start bytes are chosen to be a random 8-byte sequence. With an output throughput to the mirrors of 58Tb/s, this scheme reduces the probability that some

random 8 byte data sequence exactly matching the start bytes to once in several years [18]. At the end of the frame, we append a 1-byte CRC-8 code. The generator polynomial used is: $g(x) = x^8+x^2+x^1+1$. This choice allows us to detect several common types of errors. A more detailed explanation of CRC codes, generator polynomials and their error detection capabilities can be found in [16]. The start bytes are appended to the start of the frame. The overhead of framing a data packet is then 8 start bytes +1 CRC byte = 9 bytes. For a 16,328 byte frame this is an overhead of 9/16,328 = 0.05%.

To assure synchronization between rows, we simply check when all of the rows are loaded and error free before signaling the source to flash. This implies that we must wait for the row with the worst compression ratio to be ready before flashing. The impact on our overall throughput then depends on the variance of the compression ratio attained among rows. Since we assume that the row data will be strongly correlated, the overall throughput should not be affected too strongly by this.

3.4 DECOMPRESSION DATAPATH

A variety of possible compression algorithms exist that may be suitable for this application [2,9]. We will explore the Lempel-Ziv algorithm due to the simplicity of implementation and relatively good compression ratios achieved for mask data [2]. Compression algorithms can usually be separated into two separate steps: 1) a representational coder which models the data effectively and 2) an entropy coder which takes the symbols used by the model and generates a concise stream of bits to represent them. In decompression, the steps are reversed to recover the original data.

One proposed decompression system uses the Lempel-Ziv algorithm as the representational coder, and a Huffman coder for the entropy coding [9]. The system attains

relatively good compression figures for images with a lot of redundancy, examples of which are layouts. The system is also relatively simple to implement in hardware. A more detailed view of the proposed system is shown in Figure 11.

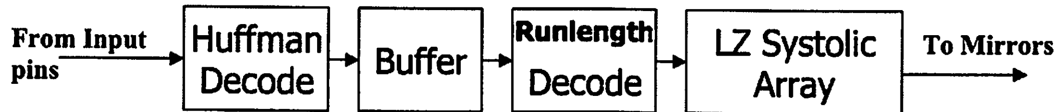


Figure 11 Block Diagram of Decompression System.

The main components of this system are the Huffman decoder, buffer, Runlength decoder and Lempel-Ziv systolic array. These components are described in greater detail in Chapter 4. The Huffman decoder and Lempel-Ziv array operate at different, time-varying rates. This is the reason for the buffer between the components as shown in the figure. It would be natural to use a first in first out (FIFO) buffer for this element. The incorporation of this element into the design is discussed in Appendix A. After decompression, the data must be fed to the writer array; this can be done with simple shift registers or a memory array and is discussed in more detail in Chapter 4. Using a parallel number of chips of the type being designed here can increase the throughput of the system. There is a requirement, beyond the scope of this work, of aligning the chips to adequate accuracy. Ideally, we would like the chip to be reproducible over generations. To ensure this, four major areas will be focused on during the design. These include area, power, speed and compression requirements. The following sections explain the issues facing us due to device scaling from generation to generation.

3.5 SCALING ISSUES

3.5.1 Area

Every technology generation the minimum feature size deceases by a factor of approximately $\sqrt{2}$ implying a decrease in area by a factor of 2. This doubles the throughput requirements each generation. We can solve this problem by using one of two approaches. Assuming that the mirrors also scale through generations, then we could double the number of mirrors in our array. The second approach would be to double the source frequency which may be unreasonable due to technology limitations. If it were possible however, then we could double the clock frequency of the circuitry. A third approach would be to use multiple chips as was previously mentioned.

3.5.2 Power

The general formula for the power dissipation in digital circuitry is given by

$$P \approx \alpha * C_{eff} * Vdd * V_{swing} * f \qquad (1)$$

Where C_{eff} is the total capacitance at all switching nodes, *Vdd* is the power supply voltage, V_{swing} is the average voltage swing of the switching nodes, f is the average frequency of the circuitry, and α is the activity factor of the switching nodes. Given that we want to maintain a 60 second/layer writing time through generations, it implies that the throughput of data must double every generation. Assuming full scaling through each generation we see that we will have capacitance scaling up by a factor of $\sqrt{2}$ through each generation since gate area is halved (neglecting wiring capacitance), oxide thickness is decreased by $\sqrt{2}$ and we have twice as many

gates. Since voltage is also scaled by $\sqrt{2}$, and power depends on the square of the voltage, we see from (2) that the average power will decrease by a factor of $\sqrt{2}$ each generation.

3.5.3 Speed

As shown in the previous section, we will be faced with the doubling of throughput required due to device scaling. Assuming that we will also be able to double the number of mirrors every generation due to device scaling, we do not need to increase clock frequencies to allow for doubling of throughput. This is because we can fit twice as many parallel decompression paths in the same area every generation. At the same clock frequency, this translates to twice the throughput of the previous generation.

3.5.4 Compression Needs

The need for compression comes from the observation that the data throughput to a chip, defined as number of pins multiplied by data-rate per pin has been trailing behind the throughput required each generation [10]. By compressing the data by the ratio of required throughput to available throughput, we overcome the problem of getting enough data onto the chip. The use of high-speed I/O technologies will allow us to stay within I/O pin count limits. Several compression techniques have been proposed in [2] with varying complexities and compression ratios to combat the problem.

3.6 System Specifications

A thermometer code format will be used for the data representation due to area constraints on the mirror interface circuitry, allowing us to use one storage element underneath each mirror. Due to area constraints on the chip, we initially chose to implement a mirror array of

approximately 16,000x16,000. The number of banks in the mirror array was shown to be a factor of the levels of gray, for convenience we will use 16,368. For simplicity we will use 16,384 rows, which is chosen to be a power of 2 for simpler de-multiplexing from the decompression paths. This gives us a 16,384 x 16,368 mirror array. We showed that for a thermometer code representation we are required to have a throughput of 58 Tb/s to the mirror array. We also assumed that we will be able to incorporate 128, 3.125 Gb/s input pins on a single chip to give us 400Gb/s input throughput to the chip. We also assume that we can obtain an average compression ratio of 60 to give us an average throughput of 0.4Tb/s*60 = 24Tb/s to the mirrors. Each decompressor path outputs 8 bits/cycle and the input data rate is 1/8 the rate of the I/O pin. Our chip specification are then:

- 128, 3.125 Gb/s input pins
- 1024 parallel decompression paths
- 16,384 x 16368 Mirror Interface Memory
- 16,384/1024 = 16 mirror rows loaded per decompressor
- 24Tb/s output throughput to allow for 1 layer in 2.5 minutes write time.

CHAPTER 4

DESIGN OF DECOMPRESSOR DATA PATH AND MIRROR INTERFACE

The design of a decompressor data path which consists of a Huffman decoder, Runlength decoder and Lempel-Ziv systolic array is presented. An SRAM based interface to the mirrors is proposed and discussed. A DRAM array is an alternative memory-based solution that is briefly mentioned. An alternative to the memory-based approach is a shift register-based approach; this is evaluated towards the end of the chapter.

4.1 CIRCUIT DESIGN APPROACH

All of the design trade-offs described above and quickly time-varying parameters in this early stage of development calls for a flexible design approach. Firstly, we believe that no matter what implementation is taken, whether it is a single chip or a multiple chip solution, the writer interface circuitry must be minimized in area in order to allow for minimal sized mirrors to fit on top with very little spacing between them. Secondly, it was shown in [9] that power consumption will be a big issue in any single chip implementation. So we need to optimize the circuitry for area and power. This must be done while keeping the data-path delay small enough to meet the necessary throughput requirements. Initially, it was believed that each mirror could

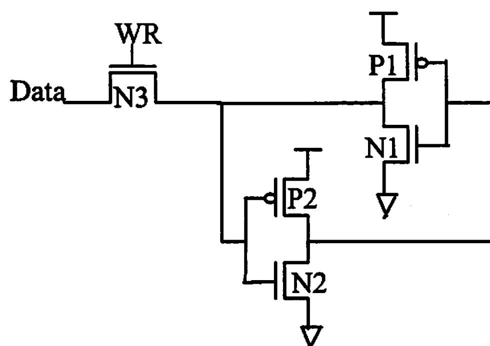


Figure 12 5T write-only SRAM cell

be placed directly on top of the decompression circuitry. Such an implementation would avoid the need for any mirror interface circuitry. This solution is clearly not feasible due to our area constraints. This implies that a memory interface to the mirrors is necessary. The mirror interface circuitry must receive data from the decompressor and transfer it to the mirrors.

4.2 SRAM BASED MIRROR INTERFACE CIRCUITRY

It is worthwhile to analyze a write-only SRAM based architecture due to good noise immunity and a standard layout that can be used to minimize the area. A write-only SRAM eliminates the need for sense amplifiers and output multiplexers; furthermore we can use 5 transistors per cell to reduce the area. The basic memory cell is shown in Figure 12. Since we have a threshold drop after N3, P2 must be sized to have approximately half the drive strength of N2 to allow the data bit to more easily toggle the state [13]. To test that the design works, we could use a CCD camera as the data is loaded into the array. We can also use standard mask inspection technology to inspect a wafer after it is written; this technique is used by Micronics [3]. The required rate of loading data is approximately $\frac{58\,Tb/s}{1\,bit/row} * \frac{1}{16{,}384\,rows} \approx 3.6\,GHz$ for thermometer format. To attempt to load an SRAM at this rate is unrealistic, rather we can

segment the memory into 8 banks and load the memory in parallel to attain a more reasonable 3.6GHz/8 = 450MHz load rate, as shown in the next section.

4.2.1 Layout Considerations for SRAM array

The layout of the SRAM array is a challenging task. The core must be designed to allow for the mirrors to be placed as close as possible to each other to maximize the aperture ratio. The aperture ratio is the ratio of mirror area to total area of the mirror array including mirror gaps. This implies that the address decoder must be placed around the outside of the memory array. Figure 13 shows one row of the memory array. In the figure, the CC blocks represent the cross-coupled inverter pairs of the memory cell. From left to right, we first see data arriving from a decompressor path. We assume that the path is 8 bits wide and is clocked at 8 times the load rate of the memory. To interface to the slower memory, we use a 1:8 serial to parallel converter for each bit output of the decompressor. The 8-bit output from the converter is then routed in parallel to each of the 8 banks of the memory. Using a 16,384 x 16,368 memory array, we have that each bank is of size 16,384 x 2046. This implies that each bit line is connected to 2046 NMOS drains. To minimize the capacitance and resistance of the bit lines we should use one of the thicker, top metal layers. The worst-case delay will be witnessed by the bit line that has to be routed to the furthest bank, bank 8. For this case, to estimate the RC delay due to the bit line using 0.25 μ process parameters, we assume that level 5 metal is used for the bit line. We also assume that the bit line is 2 times the minimum feature size in width. From the 0.25um process parameters, neglecting fringe effects, we find the capacitance and sheet resistance to be $8aF/\mu m^2$ and $30m\Omega$ respectively. The worst-case drain capacitance for a minimum sized NMOS pass gate is found to be approximately 0.2fF.

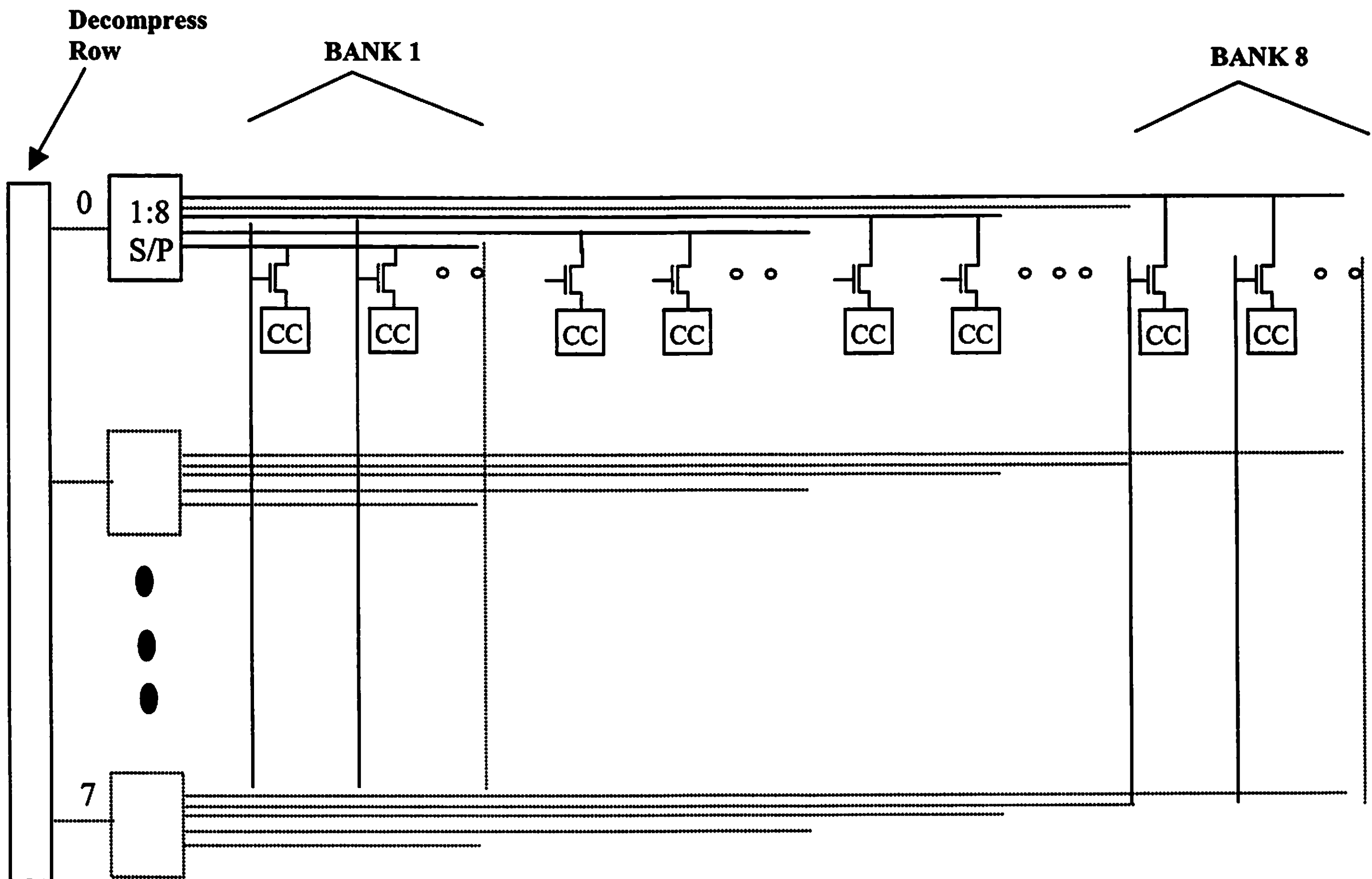
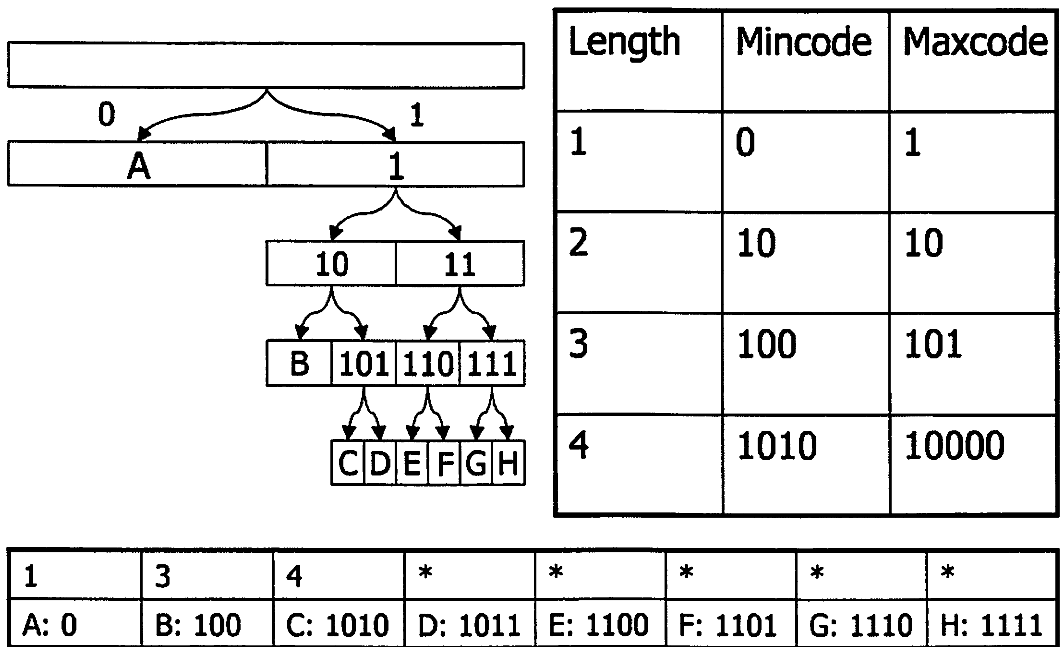


Figure 13 Parallel Load SRAM architecture

For a 16,368 x 0.5 micron bit line we then have R = 980Ω, C = $65fF(wire) + 410fF(drains) = 475fF$. The time constant for this line can be calculated as: $\tau = RC = 980\Omega * 475fF$ $\tau \approx$ 0.47nS. Assuming that the sheet resistance of the metal stays constant through generations, and full scaling otherwise, we estimate the drain capacitance in 70nm to be $\left[\frac{70}{250}\right] * 0.2F \approx 56\, aF$. The resistance for a metal line with a width of 140nm is found to be $\frac{16{,}368}{0.14} * 30m\Omega / square \approx$ 3500 Ω. The capacitance of the bit line is then calculated as $16{,}368\mu m * 0.14\mu m * 8aF / \mu m^2 * \frac{250}{70}$ ≈ 65fF. The total capacitance is then C =

$65fF(wire) + 115fF(drains) = 180fF$. The time constant in 70nm is then given by τ = $3500\Omega * 180fF \approx$ 0.63 nS. Since this time constant is much less than the required load time of 2.2nS, we can design drive circuitry to meet the load time requirements. To achieve the required load time of the memory, two approaches can be taken. The first approach is to place buffers in between the serial to parallel output, and the bit line input of the memory. The buffers should be tapered accordingly, as shown in [14]. A second possibility is to place intermediate buffers along the bit lines to decrease the write times.



Length	Mincode	Maxcode
1	0	1
2	10	10
3	100	101
4	1010	10000

1	3	4	*	*	*	*	*
A: 0	B: 100	C: 1010	D: 1011	E: 1100	F: 1101	G: 1110	H: 1111

Figure 14 Canonical Huffman Table

4.3 Decompressor Architecture

Compression is achieved using a Lempel-Ziv matching algorithm, followed by a Huffman entropy coder. When the data is decompressed, these two steps are reversed to recover the original data. This compression technique was shown to attain a compression ratio of around

80 for layout data with a lot of repetition such as memory [2]. The architecture for these two blocks are discussed below.

4.3.1 Huffman Decoder

In Huffman coding, we try to take advantage of the fact that certain symbols in a stream might appear more often then others. So we assign variable length codes to different symbols based on the usage. In the example in Figure 14, the symbol 'A' is the most frequently used, and is assigned the shortest code '0'. A less frequently used symbol is 'B', and is assigned a longer code "100". In this case, rare symbols such as 'C' and 'D' are assigned longer codes "1010" and "1011" respectively. The input sequence "ABAAACAADAAA" would be written as "0 100 0 0 0 1010 0 0 1011 0 0 0". If we had done a simple encoding using 3 bits per symbol for each of the characters A-H, our output would be 36 bits long. Instead using the Huffman code reduced the output to 20 bits. Theoretically speaking, Huffman codes are allowed to be infinite in length; the less frequently a symbol appears, the longer the code that is assigned to it. Moreover, from an information theoretic point of view, assuming that the input symbols are independent and identically distributed, Huffman codes come very close to entropy, the minimum average number of bits per symbol. From a practical point of view, however, the length of a Huffman code must be limited to some value MAX, and input symbols must be decorrelated using a representation coding such as the Lempel-Ziv algorithm.

```
Input data:      ABCDEFGHIJKLABCDEFGHIMNOPQRSTUVWXYZ
Compressed data:  ABCDEFGHIJKL<12,9>MNOPORSTUVWXYZ
```

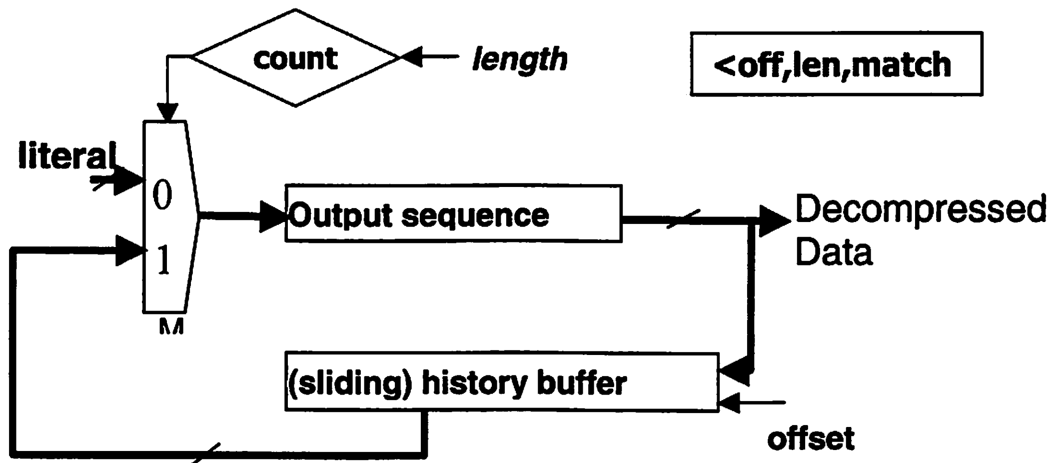


Figure 15 Lempel-Ziv Decompression.

4.3.2 Lempel-Ziv Decompression

The goal of the Lempel-Ziv algorithm is to take advantage of the redundancy in the data in the most efficient way possible. After compression, the Lempel-Ziv algorithm outputs one of two data types. The first is a literal, which is a word that could not be compressed using the algorithm. The second is an offset/length pair, where the offset is the information about how far in the history buffer to look, and the length is how many bytes to copy from the history buffer. A typical sequence of data that is input into the LZ compressor is shown in Figure 15. The VLSI implementation of this algorithm is divided into two parts: runlength decoder and history buffer lookup. In the next two sections, we expand upon on each of these modules in detail.

4.3.2.1 Runlength Decoder

The main function of the runlength decoder is two-fold. The first is to decode the input stream into either <literal> , <length, offset> pairs, and the second is to output the offset byte as many times as the length byte indicates. The repetition unit contains a counter and several

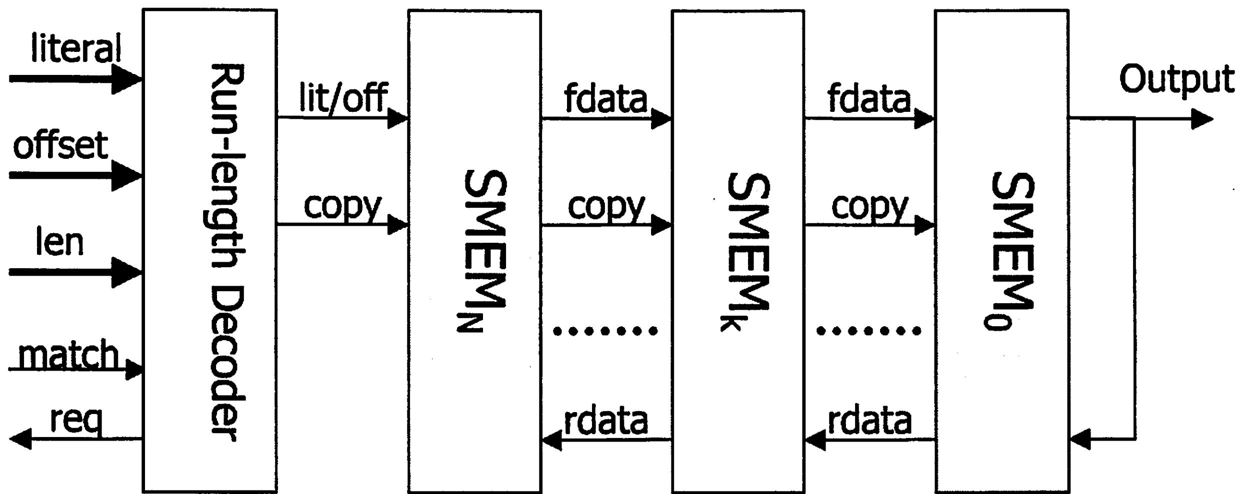
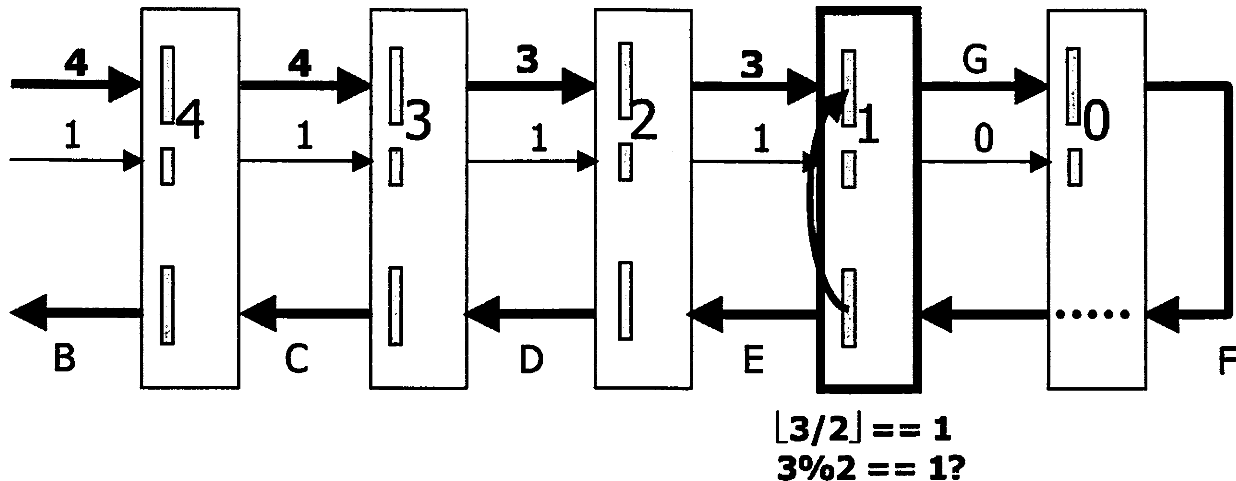


Figure 16 Lempel-Ziv Systolic Array.

registers. The decoder looks at the first bit of the stream to decide between the two possible outputs. If the bit is a '0', then this indicates that the symbol is a literal. If the bit is a '1', then the decoder initializes the counter to the length symbol, and outputs the offset byte, decrementing the counter after each cycle.

4.3.2.2 Systolic Architecture

The systolic architecture consists of a series of decoding cells. The decoding is done locally, avoiding any long wires to the history lookup buffer observed in the non-systolic implementation. This increases the robustness to noise and maximizes the operating speed of the decoder. Decoding is fairly simple; cell n compares the input offset value to the stored offset value of the cell which are 2_n, 2_{n-1}. The output decoded value is copied from the reverse buffer R_n or R_{n-1}, where R_{n-1} comes from the preceding decoding cell, $n-1$. The decompressor receives either a literal or an offset signals. A flag is passed along the cells to differentiate between the two. In the systolic array implementation, data pass bi-directionally from one processing element



Compressed data stream: ABCDEFG<3,2><4,2>ABC
Output so far: ABCDE**F**

Desired output: ABCDEFG**EFFG**ABC

Figure 17 Systolic LZ Array Decoding.

to neighboring ones in a regular, synchronous pattern as shown in Figure 16. Each SMEM cell consists of a forward and a reverse buffer. Data is fed through the forward buffers from left to right, and then fed back around the reverse buffers from right to left. To match the history buffer output, forward moving data is compared to the cell number, until it reaches the correct cell where it can fetch the literal from the reverse buffer. An example of decoding in the systolic architecture is shown in Figure 17.

4.4 Design of Decompression Circuitry

To attain the necessary throughput, we must implement a number of compression paths operating in parallel. The constraints are that it must fit in the 2cm x 2.5cm chip, along with the 1.6cm x 1.6 cm mirror array. This implies that we have approximately 2cm x 0.9cm of area to implement the decompressors. Given this constraint, we would like the output throughput of the

decompressors to be as close as possible to the required 58 Tb/s. The power consumption must also be kept to a reasonable level. The basic building blocks of the design were designed and simulated in Module Compiler, a data-path synthesis tool, using 0.18μm technology. The design used the standard cell library. Table 1 below summarizes the performance. The three components of the design that used the most power and area were the Huffman Decoder, Runlength Decoder and Systolic Array. From the table, we see that the SMEM array consumes the majority of the power and area. Figure 18 below shows the circuit diagram of an SMEM cell.

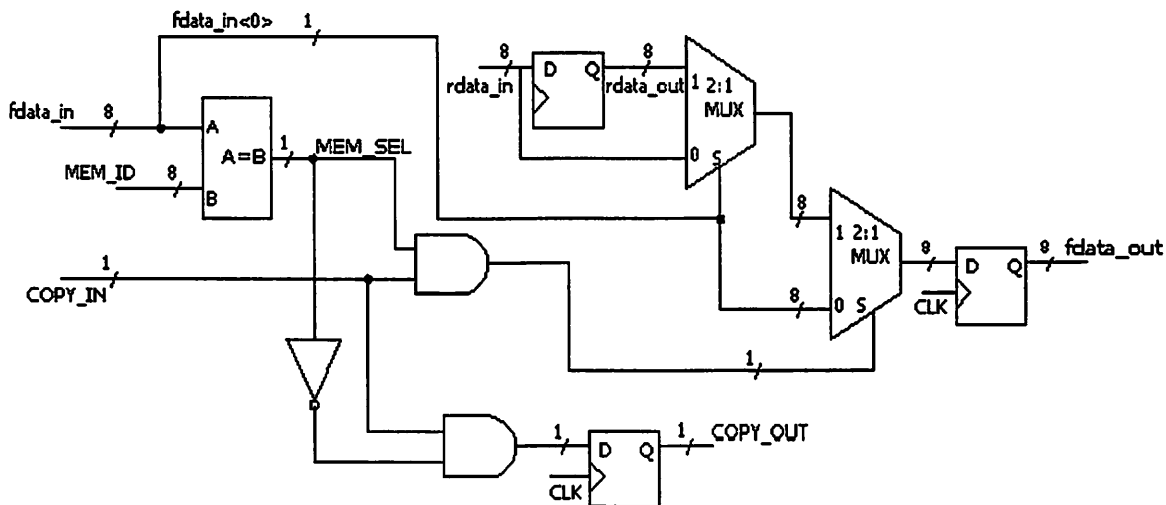


Figure 18: SMEM Cell Block Diagram.

In Figure 18 we see that the SMEM cell is composed of 17 flip-flops, two 8-bit multiplexers, an 8-bit comparator and 3 gates. In this design, the main consumers of area and power are the flip-flops. A custom design of this SMEM array can save approximately 50% in power and operate at twice the speed as shown in Appendix B.

Table 1 Performance of Decompressor (Standard cell based design)

	Max Speed	Power	Area
Huffman decoder	700 MHz	3.15mW @ 500MHz	6,982 μm^2
Runlength decoder	625 MHz	1.2 mW @ 500MHz	3,307 μm^2
128 element Systolic LZ Array	1000 MHz	59.2mW @ 500MHz	262,000 μm^2

4.5 Alternative Memory Interface - Shift Register Based

As discussed in Chapter 3, a shift register array coupled with comparator circuitry is an attractive solution when using a binary data format since we do not have a data expansion as seen in the thermometer format. The disadvantage of such an implementation is that it requires 6 storage elements per mirror to allow the pixel to propagate along the row. We can fit approximately 2 storage elements below each mirror, making this a viable solution only for mirrors that are larger than 2 microns on a side. The operating speed was obtained by simulating the shift registers at several clock frequencies. We can estimate the power consumption of a 16000 X 16000 mirror array, with binary data format, and parallel shifting. The required clock frequency into the shift array is approximately $\frac{9.4\ Tbits/s}{16{,}384 * 5bits/pixel} \approx 100\text{MHz}$. We simulated 4-element, 6-bit wide shift registers using NMOS gate dynamic flip-flops, which gave us a power of 176uW@1GHz. From this we can estimate the power consumption of a 16,384 x 16,368 shift register array operating at 100MHz as: $\frac{16{,}384 * 16{,}384 * 176\mu W @1GHz}{4 * 10} \approx 1130$ W. Scaling down to the 70nm generation and assuming an activity factor of 0.2 for the switching nodes, we

get an estimate of $1130W * \left[\frac{70}{180}\right]^2 * 0.2 \approx 34$ W. We can reduce the above power figure by shifting data in parallel, which would lower the requiring a clock frequency to achieve the same throughput. Using a shift register for the thermometer data format, we would be required to use a 1-bit wide shift register underneath each row of mirrors. As was shown previously, the load rate would be 31 times faster, so that we could estimate the power consumption for this configuration as 34W*1/6*31 =176W.

4.6 ALTERNATE INTERFACE CIRCUITRY : DRAM BASED

A DRAM (dynamic random access memory) architecture could be an attractive option for the mirror interface circuitry. The main concern of this approach is the soft error rate of the memory. A DRAM architecture built on standard CMOS could be attractive for this application, since at a high flash rate we would not need a very high capacitance per cell. It would also be very expensive to have a DRAM+Logic+MEMs process. The lower limit of the capacitance needed per DRAM cell to avoid soft errors is approximately 20fF [13]. We will assume that the capacitor is formed between the N^+ active layer and the poly layer. In 0.25µ process, we have 6fF/µm^2 [12]. Assuming constant voltage scaling, the oxide thickness decreases by a factor of $\sqrt{2}$ each generation, increasing capacitance by $\sqrt{2}$ each generation. We can estimate that we will have approximately $\frac{250\ nm}{70nm} * 6042af / \mu m^2$ = 22fF/µm^2 in the 70nm generation [12]. We would also need to fit a transistor and contacts in the 1µm^2 area. This places us below the soft error rate threshold. Further investigation would be required to analyze the soft error rate obtained by using a fraction of this capacitance. A sketch of a 1-T DRAM cell implemented in standard CMOS is shown in Figure 19.

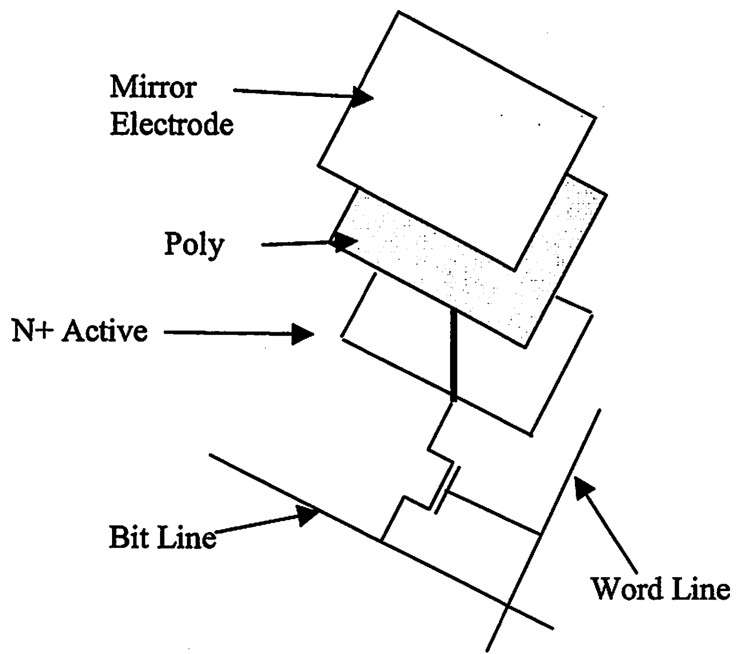


Figure 19 1T DRAM Cell

CHAPTER 5 – PROTOTYPE IMPLEMENTATION AND RESULTS

This chapter discusses the implementation of a scaled down prototype designed in 0.18μ technology, with the goal of demonstrating the concepts described in this thesis. From this we also try to extrapolate the performance of a full-scale chip implemented in the 70nm generation.

5.1 CHIP DESCRIPTION

The architecture used for the design of the prototype chip is a decompressor consisting of a Huffman Decoder, Runlength Decoder and a 128-element Lempel-Ziv systolic array. The mirror interface circuit is an SRAM memory, which is fed data by the decompressor. At lower complexity, we have enough input pins to avoid the need for high-speed I/O pads. Specifically the chip is designed to implement 8 decompress paths; the following is a description of the chip:

- 100 Mb/s I/O pads
- Huffman Entropy decoder
- LZ decompressor using 128 element SMEM array
- Grayscaling with thermometer code data format
- 256 byte FIFO for interfacing LZ and Huffman
- 8 start bytes, 1024 data bytes, and 1 CRC byte per frame
- 64x1024 SRAM memory array for mirror interface
- Single clock operation

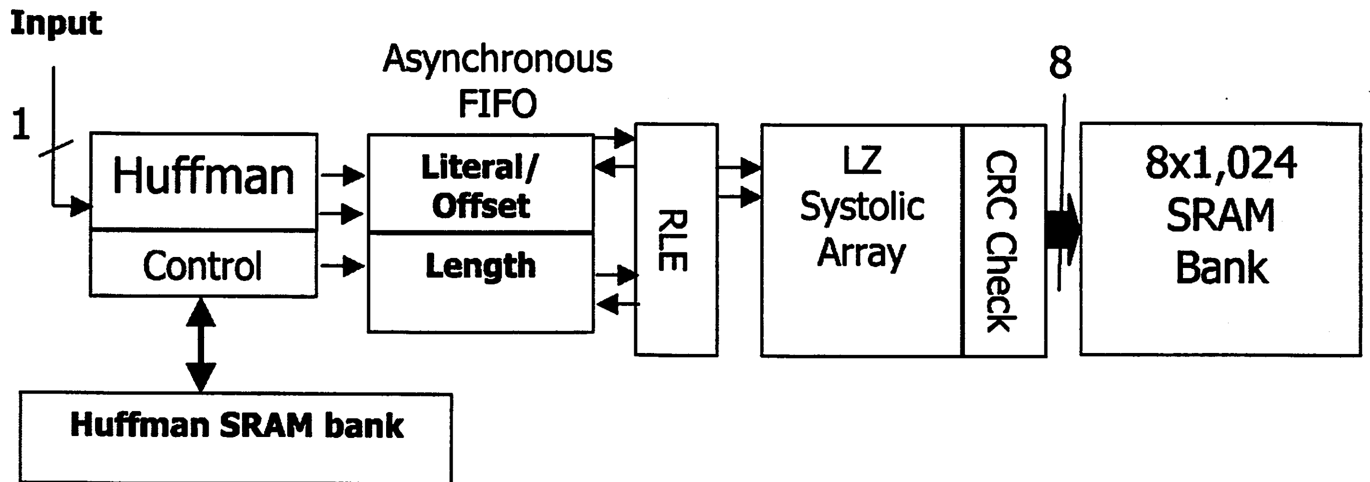


Figure 20 Block Diagram of One Decompress Row

One decompression row and memory row of the system is shown in Figure 20. Since we have a 1-bit data stream flowing into the Huffman, a controller block is needed to decide what type of data is coming in, whether it is a literal, offset or length byte. Based on this, the controller controls the multiplexing of the correct MAX, MIN and OFFSET tables to the Huffman. A FIFO is necessary to interface between the Huffman and the LZ systolic array to smooth out data rate fluctuations between the components. The Huffman decoder and the Lempel-Ziv array have different, time- varying compression ratios. Since the Huffman input is 1 bit wide, and the output is 8 bits wide, while the LZ has both input and output 8 bits wide, we have that the clock ratio of the LZ to the Huffman given by: $\frac{\textit{LZ compression ratio} * \textit{Huffman compression ratio}}{8}$, this is derived in [9]. A FIFO level detector was also built to detect when the FIFO is within 10% of overflowing or if it is empty. In the event that the FIFO is empty, the Lempel-Ziv array will stall until there is data available. In the event that the FIFO is close to overflowing, an overflow pin is asserted, informing the processor board to stop sending data. The data from the decompressor is

sent to the mirror interface circuitry, which is an SRAM array. We use a thermometer data format to achieve 31 non-zero levels of gray. Error detection is accomplished using a 1 byte CRC that is appended to the end of each frame. An 8-byte start byte is appended to the beginning of each frame. The start bytes were arbitrarily chosen to be "maskless". Since the main purpose of the test chip is to demonstrate the concept, we use a single clock domain for the whole chip. The chip I/O is described in Table 2. The decompression circuitry is implemented using standard cells in 0.18μ technology. The blocks were designed using Synopsys Module Compiler.

Table 2 Chip I/O

I/O	**Width**	**Description**
Data (I)	8	Input
Reset (I)	1	Reset
Load_Data (I)	16	Data for Huffman memories
Load? (I)	1	Enable Huffman mem. Load
Read_Mirrors (I)	3	Read mirror array data
Mirror_Reset (I)	1	Reset address decoder
WE (I)	4	Select Huffman memory
Mirror_Data (O)	8	Data from mirror memory
Clock (I)	1	Global clock
Overflow (O)	1	Overflow imminent

The memories were obtained by a memory generator, provided by ST Microelectronics. The memory generator allows us to specify the size of the memory desired. The floorplan and layout are shown in Figure 21. Simulink to Silicon SSHAFT flow. SSHAFT stands for Simulink to Silicon Hierarchial Automated Flow Tools (SSHAFT) [15]. The idea behind SSHAFT is to allow an algorithms designer to develop a prototype chip in a matter of days without any prior knowledge about backend chip assembly. Simulink is an event-driven simulator that is part of the Matlab software package. A high level Simulink model is developed for the system and used to drive the design. The Simulink model should functionally match the chip exactly. A data path description must be written to specify how the blocks of the model are generated. The blocks can be designed with Module compiler, or synthesized directly from VHDL. All of the blocks for the prototype chip were designed with Module Compiler. Module Compiler is a data-path synthesis tool, where the architecture of the design is specified in a syntax similar to the C language. The design consists completely of standard cells and SRAM memories for the Huffman memories and the mirror memory. To test the design, a sample data stream was used. A program was written in Matlab to take in the data stream, add the start bytes and CRC byte and then compress the stream. This compressed data is used to test the design at all levels. We test the design at the Simulink level, then the VHDL level, and then the raw netlist level. Several of these simulations are presented in Appendix A. The decompressor path widths in the layout are determined by the height of the mirror-interface memory as shown in Figure 21.

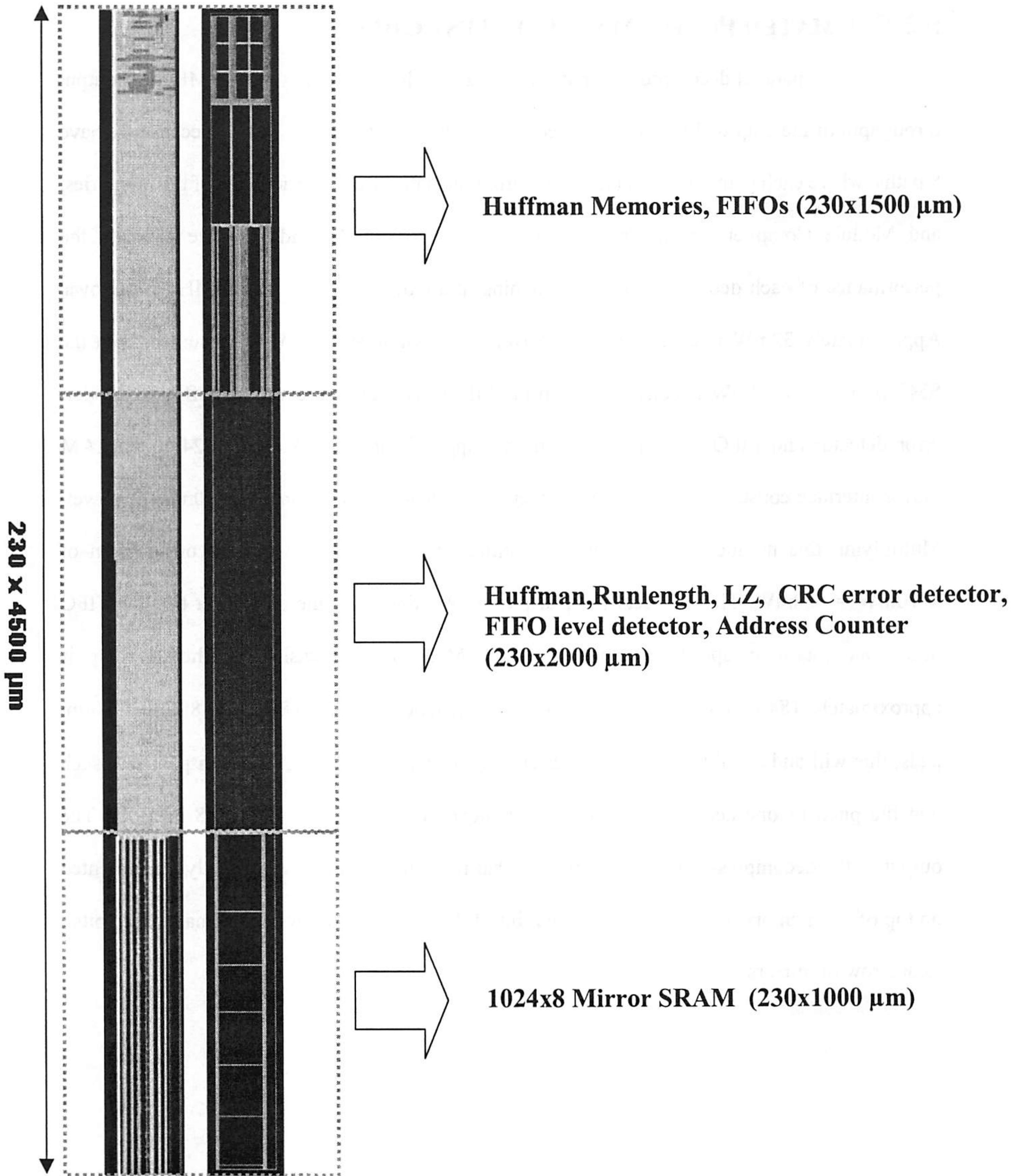


Figure 21 Decompressor Row Floorplan and Layout

5.2 ESTIMATED PERFORMANCE OF TEST CHIP

With 8 parallel decompression paths running at a clock frequency of 100 MHz, the output throughput of the chip will be approximately 8*8*100Mb/s = 6.4 Gbps. This is because we have 8 paths, where each path outputs 8 bits/cycle. From the data sheet specifications of the memories, and Module Compiler simulations, which can be found in Appendix A, we estimate the performance of each decompression path running at a clock frequency of 100MHz as follows. Approximately 32mW is consumed by the Huffman memories, 12mW is consumed from the SMEM array and 13mW is consumed from the FIFOs. The address counter, Huffman decoder, error detector and FIFO level detector consume approximately 2mW. The 1024 X 8 SRAM mirror interface consumes 11mW. All together we get that one path consumes 70mW of power. Multiplying this number by 8, we get an estimate for the test chip power consumption of 8*70mW = 560mW. This neglects I/O pad power. We find from the data sheet that the FIFO limits our maximum speed of operation to 236 MHz. The dimensions of the test chip is approximately 1840μm x 4500μm. This does not include I/O pads. Using 44, 80μm X 80μm pads, this will add an additional 3% to our area numbers. The layout of the data path was such that the pitch of one decompressor row was matched to the pitch of a 1024x8 memory. The output of the decompressor is 8 bits, implying that if the mirror array was directly implemented on top of the memory array, then each output bit of the decompressor would be matched in pitch to one row of mirrors.

5.3 Estimated Performance of Future Design

In the following sections, we will assume full scaling through generations.

5.3.1 Throughput

The chip to be designed in the 70nm generation will have 128, 3.125 Gb/s I/O pads. The data from these pins will be de-multiplexed at a ratio of ~ 1:8 to the Huffman. We Assume that we can attain a compression ratio of approximately 50-60 with thermometer data format with a 1024 byte history lookup as was shown in Figure 8. This implies that the LZ array will operate at approximately $\frac{400\ Gb/s * 60}{1024\ paths} * \frac{1\ bit\ input\ per\ cycle}{8\ bits\ output\ per\ cycle} \approx 3GHz$. The data from the LZ array will then be de-multiplexed at a ratio of 1:16 to the rows of the mirror array, composed of 8, 16,384 x 2046 bit SRAM banks operating at approximately 200 MHz each. A similar configuration was discussed in section 4.2.1. The configuration achieves an output throughput of $128 * 3.125\ Gbps * 60 \approx 24\ Tb/s$. This throughput is approximately 40% of our 58 Tb/s objective.

5.3.2 Area

From Figure 21 we see that area occupied by the standard cells is approximately 460,000μm^2. Since the SMEM array occupies approximately 88% of this area as shown in A.2, the area of the 128 element SMEM can be estimated as $460{,}000\mu m^2 * .88 \approx 400{,}000\mu m^2$. The layout of the standard cells was done with 80% density, in a custom placed layout we would be able to double the density, to get an area estimate of 200,000μm^2. In the design, a 512 element SMEM would be used to give us our required history lookup size of 1024 bytes.

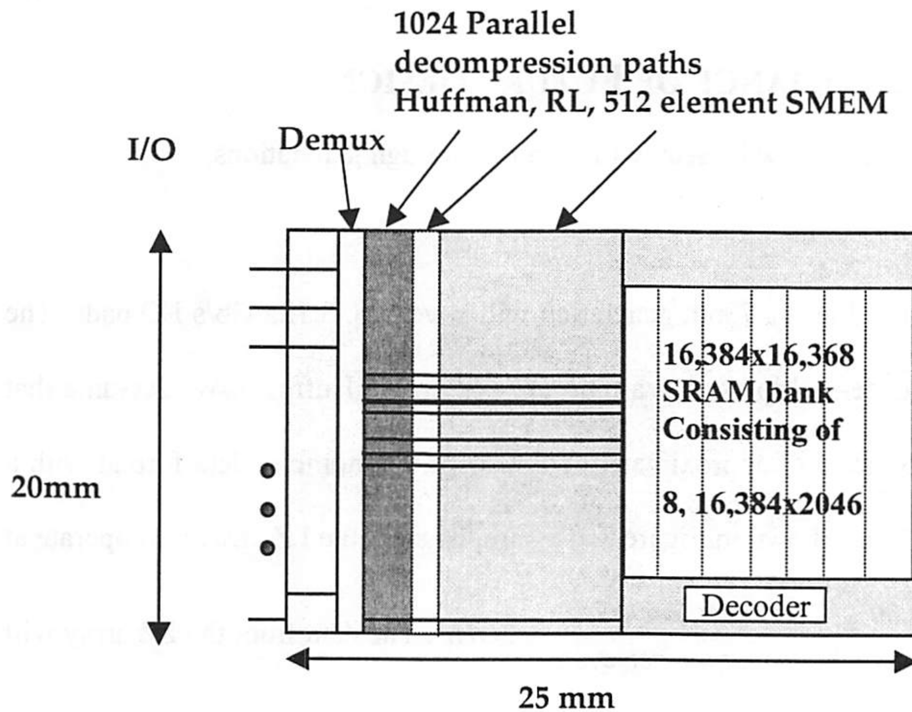


Figure 22 Floorplan for Chip in 70nm

This would then occupy approximately 4 times the area of a 128 element SMEM, giving us an area of 200,000μm^2*4 = 800,000μm^2. Scaling down to the 70nm generation we get that the area of the 512 element SMEM will be $800{,}000\mu m^2 * \left[\frac{70nm}{180nm}\right]^2 \approx 120{,}000\mu m^2$. From Figure 21 we get that the area due to the Huffman memories and FIFOs is 345,000μm^2. Scaling down to the 70nm we get $345{,}000\mu m^2 * \left[\frac{70nm}{180nm}\right]^2 \approx 52{,}000\mu m^2$. This gives us an estimate of 120,000 μm^2 +52,000 μm^2 = 172,000 μm^2 for the total area of the decompressor circuitry. This assumes that the SMEM occupies a majority of the standard cell area. With 1024 parallel paths, we obtain a total area of 176x10^6 μm^2. From our constraint that the chip height be 2cm, we get that the length of these rows will be approximately 0.9cm. The pitch of the rows will then be $\frac{2cm}{1024\ rows} \approx 19{,}5\ \mu m$.

This translates to approximately 8 standard cell rows in 70nm technology. With this configuration we get a chip floorplan implemented in 70nm as shown in Figure 22.

5.3.3 Power

The power consumption of the chip can be broken up into the power consumption of the decompression circuitry and that of the memory array. To estimate the power consumption of the decompression circuitry, we will use the fact that the SMEM block consumes a majority of the power, and the Huffman and Runlength are negligible in the calculation. Using the power figures from the custom designed 128 element SMEM block in Table 7, we estimate $32\ mW @ 0.5GHz * 3 * 6 * \left[\frac{70nm}{180nm}\right]^2 * 1024\ paths \approx 120\ W$. The factor of 4 comes from the fact that we are using a 512 element SMEM instead of 128 that was simulated. The factor of 6 comes from the fact that we are operating at 3 GHz in the 70nm generation. We scale power by the square of the scaling factor since we have been assuming full scaling. A better estimate of power consumption is obtained by taking into consideration the activity factor of the switching nodes. With a factor of 0.2, we get a more realistic power consumption of 0.2*120W = 24W. Implementing the same design on 20 chips would imply that we could run the circuitry 20 times slower, reducing the power consumption by a factor of 20. The power consumption of the parallel load SRAM architecture discussed in section 4.2 needs to be further evaluated. Table 3 compares the estimated performance of the test chip and the future chip.

Table 3 Chip Performance

Parameter	Test Chip	Future Full-Scale Chip
I/O paths	8	128
I/O bandwidth	800Mb/s	400Gb/s
Decompression paths	8	1024
Throughput	6.4 Gb/s	24 Tb/s
Mirror-interface Type	Standard 6T SRAM	Custom 5T SRAM
Mirror-interface size	64 X 1024	16,384 X 16,368
Mirror size	3μm X 3μm	1μm X 1μm
Chip dimensions	2mm X 5mm	2cm X 2.5cm
Power	560 mW	> 24 W

5.3.4 Further Optimizations for Power and Area Savings

Since all of the Huffman decoders will have the same tables, we can use dual-ported memories to share the Huffman tables across rows to save area and power. The optimal FIFO size is a parameter that depends on the layout data and could possibly be reduced. By designing a custom SRAM that has write-only capability it allows us more freedom to use a 5-transistor cell and still satisfy the SRAM cell stability conditions [13]. This will allow us to save in area and power. The design was chosen to be implemented with 70nm technology since we wanted to have the chip to be ready by the 50nm generation. Implementing the chip in 50nm technology could be a possibility as well that needs to be evaluated.

CHAPTER 6 CONCLUSION

6.1 Key Accomplishments

In this thesis we have examined the main challenges in the design of the data path for future maskless lithography systems. We showed that a data throughput to the writers of around 58Tb/s was required in the 50nm generation to write a layer of a wafer in 60 seconds. A system was proposed to handle the throughput problem. We designed the writer interface to be a single chip solution. The writer-interface consisted of a decompressor to reduce the input bandwidth requirement of the chip, and an SRAM memory to interface to the mirrors. The decompressor consisted of a Huffman decoder, runlength decoder and a systolic Lempel-Ziv decompressor. A 5T write-only SRAM mirror interface was proposed to allow for the incorporation of maximum aperture ratio writers on top of the array. We also presented a shift register architecture, as an alternative approach to implementing the writer interface.

A single chip solution was presented. The chip used a highly parallel architecture, consisting of 1024 independent decompressor paths and an SRAM writer interface. A scaled down version of the chip was designed to demonstrate functionality. The full scale version of the chip will be implemented with a 16,384 x 16,368 mirror array and is projected to have dimensions of 2cm x 2.5 cm. The chip is projected to have throughput of around 24 Tb/s,

allowing for 2.5 minute per layer write times. By implementing the solution on multiple chips, it would mean that we would not be pad-limited anymore, implying that we would not be required to use compression, eliminating the decompression circuitry all together.

6.2 Future Work

In the near future, we would like to analyze architectures for different compression techniques such as 2D-LZ, JBIG and Burrows Wheeler [2,20]. More complex schemes will allow us to reduce the complexity of the I/O pads with the disadvantage of consuming more area. We would also like to analyze a custom designed SRAM to reduce the area and power consumption. A DRAM memory interface was also shown to be a possible solution that might reduce the area and power consumption. The solution was projected to have 128, 3.125Gbps input data pins. Many issues such as per pin skew compensation, and equalization still need to be solved for a single chip solution with such a high throughput. We would also like to examine the required writer interace circuitry for different maskless lithography techniques such as e-beam based systems. An economic feasibility analysis to determine at what throughput it becomes more advantageous to use maskless lithography is needed. This would motivate designing different systems for different market segments such as low-volume ASIC manufactures.

ACKNOWLEDGEMENTS

I would like to dedicate this work to my mother, Nilie, the greatest mother in the world. I would like to say thank you to my parents Gary and Nilie Wild, for giving me love and support since I was born. Without them, I would have never had the opportunity to study in this great school, working with great people. Thanks to my brothers Jack and Eric for being my best friends, and the great advice they gave me since childhood.

Working on this project for the past year I have learned a great deal about lithography. This would not have been possible without the time that many people have spent, discussing all of the issues involved. I would like to thank Professor Bill Oldham, Vito Dai and Yashesh Shroff, who have helped me tremendously in learning about lithography. I would like to especially thank my advisor Professor Borivoje Nikolić who was always a friend to me, and helped guide me in the right direction. Thanks to my friend, Naratip Wongkomet for contributing greatly to the project. I would like to say a special thanks to Engling Yeo and Rhett Davis for helping me learn about all of the CAD tools. Thanks to Paul Park for helping out with the FIFO interface design. Thanks to Tina Smilkstein for help with the clocktree. Thanks to Amit Mahajan for the great advice on many topics. Thanks to all the companies involved in the SRC for helping fund this project.

APPENDIX A: SIMULATION SETUP AND RESULTS

A.1 Simulink Library

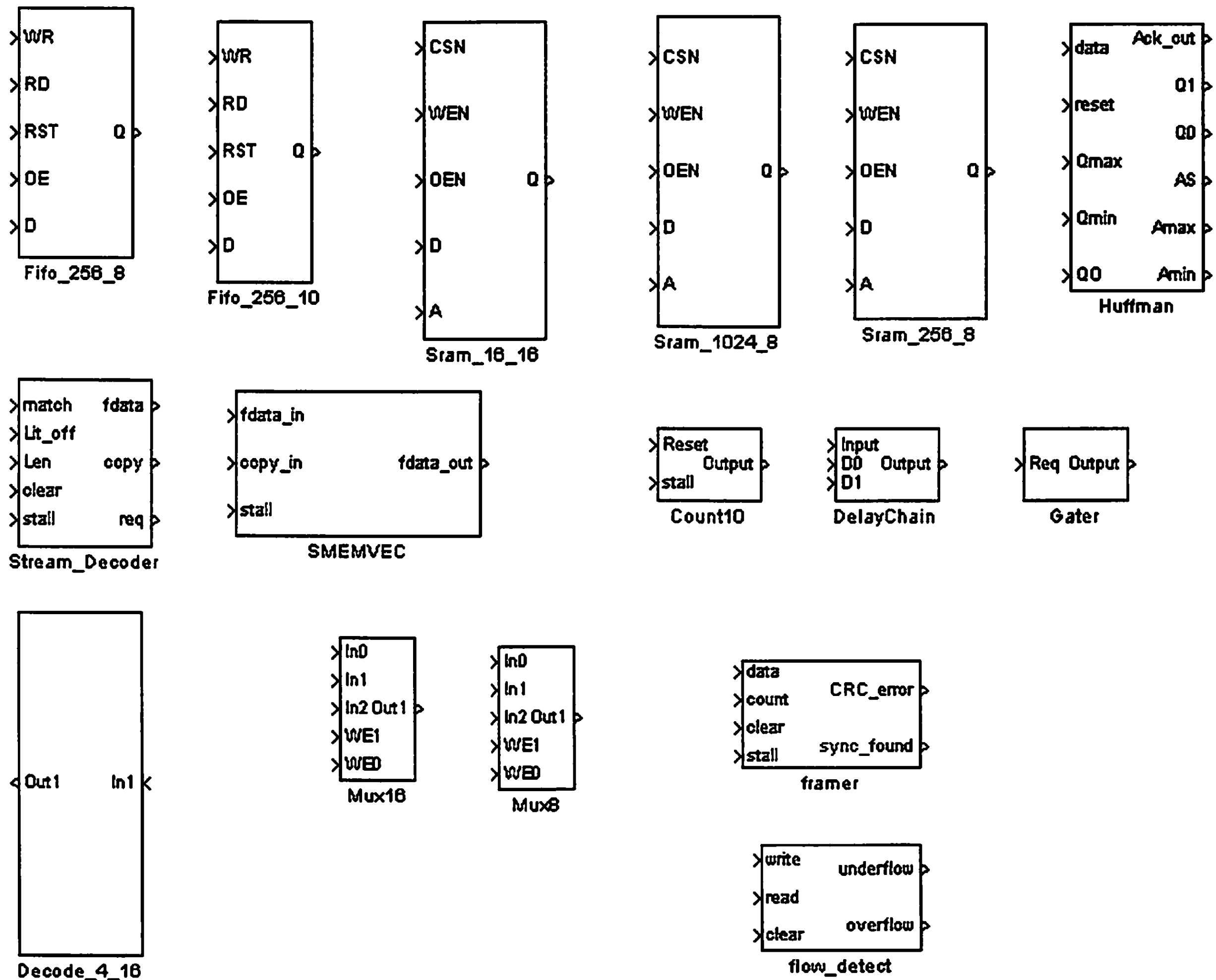


Figure 23 : Simulink Library

Figure 23 above is the library of components used in the simulink model. It includes all the main components of the design. These components are connected together to form the decompression system and the mirror interface circuitry. The SSHAFT flow will then start from this model to generate the layout. Table 4 describes the Simulink blocks.

Table 4 Simulink block description

Block	Description
Huffman	Huffman decoder
Stream_Decoder	Runlength decoder
SMEMVEC	128 element SMEM systolic LZ
Framer	Finds start bytes and checks CRC
Flow_detect	Checks for FIFO under/over flow
Fifo_256_8/Fifo_256_10	FIFO to interface Huffman with LZ
Sram_16_16	16x16 Huffman table lookup SRAM
Sram_256_8	256x8 Huffman symbol SRAM
Sram_1024_8	1024x8 SRAM writer interface
Count10	Address counter for writer interface
DelayChain	Delay chain to satisfy hold time of FIFO
Gater	Clock gater
Decode_4_16	4:16 decoder
Mux16/Mux8	3:1 16/8 bit multiplexer

A.1.1 Matlab Verification

Matlab code was written to generate test vectors, simulate the system and verify the overall functionality. The m file takes a message, appends the start bytes and generates and appends a one byte CRC word to the message. It then applies the Huffman and Lempel-Ziv algorithms to it. It then simulates the system, and checks that the system decompressed properly by cross checking the output of the system with the original message.

A.1.2 VHDL verification

VHDL simulations were run on the whole design. The VHDL code was stitched together from VHDL code provided from each of the Module Compiler blocks as well as VHDL code provided by ST Microelectronics for the memories and FIFOs. Figure 24 shows part of the simulation showing the input to the mirror memory matching.

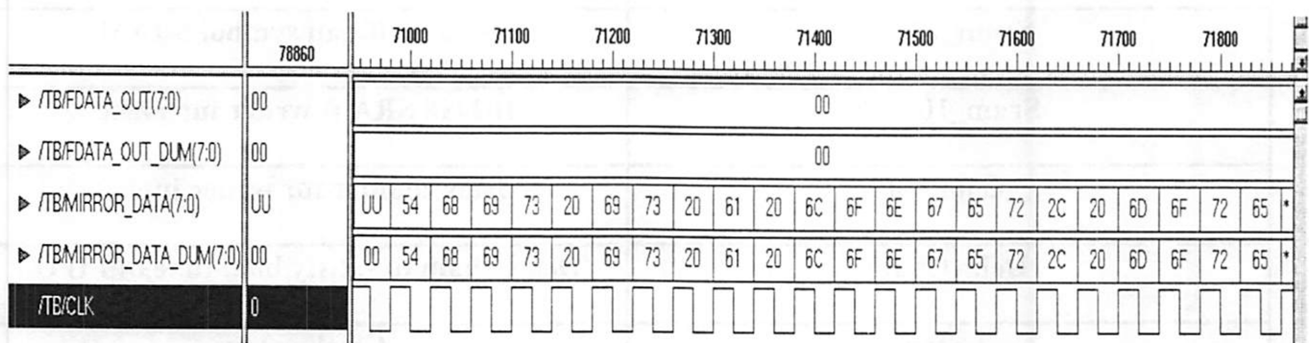


Figure 24: VHDL verification, showing Mirror Data

A.2 Module Compiler Blocks

The major subsystems of the design were built with Synopsys Module Compiler. The process used was 0.18μ with typical operating conditions. Table 5 below summarizes the performance of these blocks.

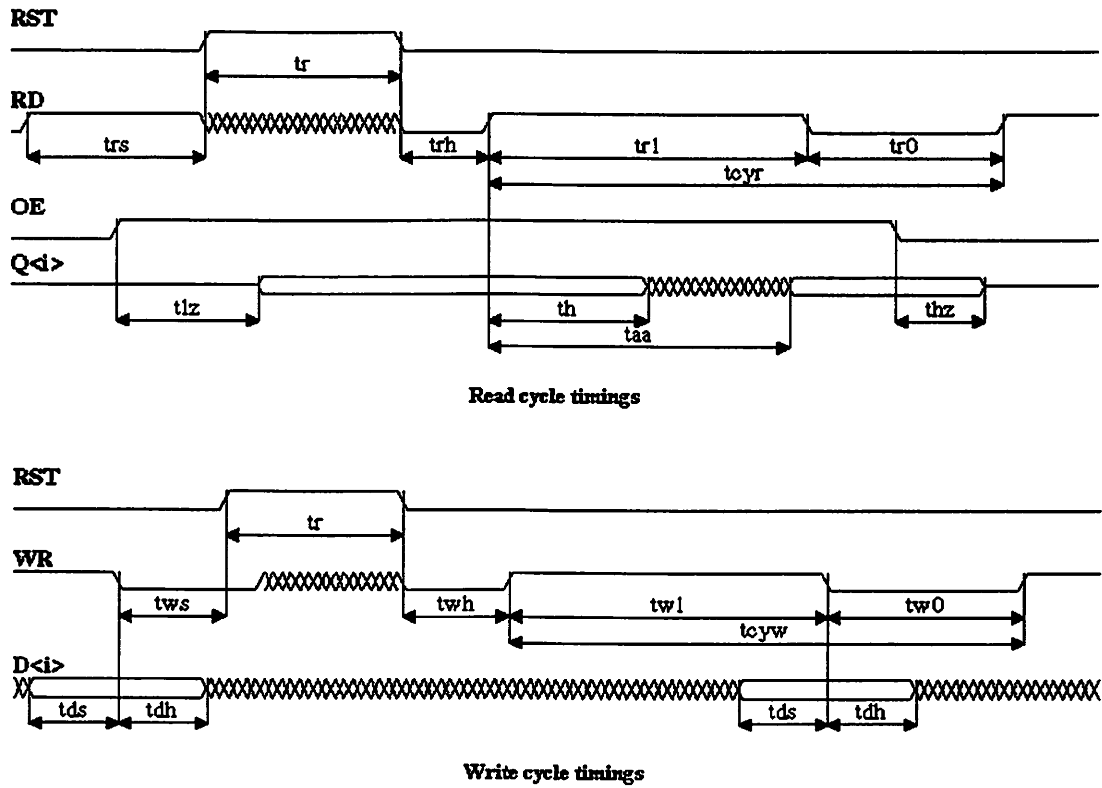
Table 5: Module Compiler and Memory Block Performance

Block	Max Speed(MHz)	Area(μm^2)	Power@100MHz(mW)
Huffman Decoder	150	6,982	0.63
Runlength Decoder	110	3,307	0.24
16x16 SRAM	820	8,379	2.7
256x8 SRAM	333	39,492	2.4
1024x8 SRAM	625	263,000	11.1
256x8 FIFO	238	73,771	5.8
256x10 FIFO	236	79,707	7
128 element SMEM Array	333	262,056	11.84
Framer	1400	17,167	0.8
FIFO level detector	840	6,490	0.55
10-Bit Address Counter	1180	2,512	0.22

Several small modifications were made to these blocks compared to the blocks described in [9]. In [9], separate Huffman decoders were used for the different streams of data. The length, offset and literal streams each had a separate coder. This was due to the different statistics of these streams. The three separate Huffmans were combined into one, with the Huffman memories multiplexed in based on which data was being processed. A state machine was built into the Huffman block to detect which type of character was being processed. Flip flops with stall enables were installed into the Runlength, Framer,10-bit Counter and SMEM array. This was needed in order to stall these components when the FIFO was empty.

A.3 FIFO Interface Circuitry

The FIFO used in this design was provided by ST Microelectronics. The timing diagram is provided below.



Read cycle timings

Write cycle timings

Figure 25 FIFO timing diagram

We see from the diagram, that write occurs on the falling edge of the WR signal. On the read cycle, the data is ready after a certain delay after the rising edge of the RD signal. A constraint that must be satisfied is that Read and Write cannot occur simultaneously. To satisfy these constraints the following interface circuitry is built around the FIFO.

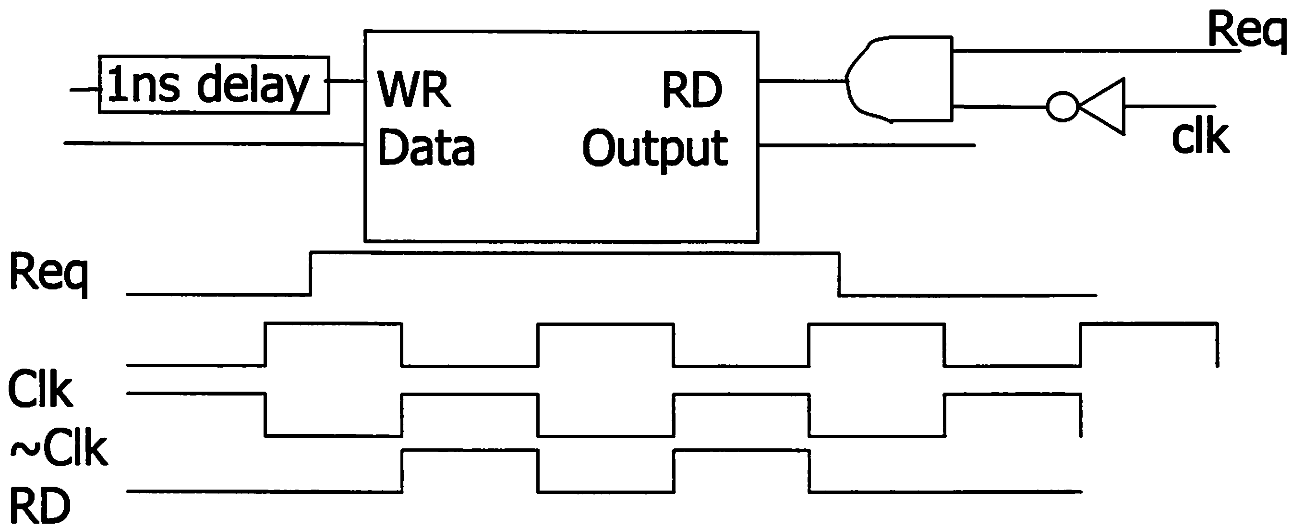


Figure 26: FIFO interface Circuitry and Timing

The write signal to the FIFO originates from the Huffman done signal. This is a synchronous signal that only transitions on the rising edge of the clock. To avoid conflicts with the read signal, the request signal from the runlength is gated with the inverted clock. This implies that data can only be requested from the FIFO on the falling edge of the clock. A 1ns delay is inserted before the write port of the FIFO to satisfy setup time constraints of the FIFO. For this specific FIFO, 600pS is necessary, although netlist simulations showed that the actual setup time of the FIFO is under 100pS.

APPENDIX B : FLIP FLOP ANALYSIS

B.1 Trade-off Between Different Dynamic Storage Elements

The basic storage element we will analyze is the D Flip Flop using a master slave configuration as well as a D Flip Flop using a pulse triggered latch configuration. These elements are shown below in Figure 27. The figure shows several configurations of dynamic flip-flops. The use of dynamic flip flops will allow us to save area in return for a decrease in noise tolerance, which will be analyzed in further detail. Table 6 below summarizes the performance of several of the flip-flop configurations. The simulations were done in 0.18μ, typical conditions.

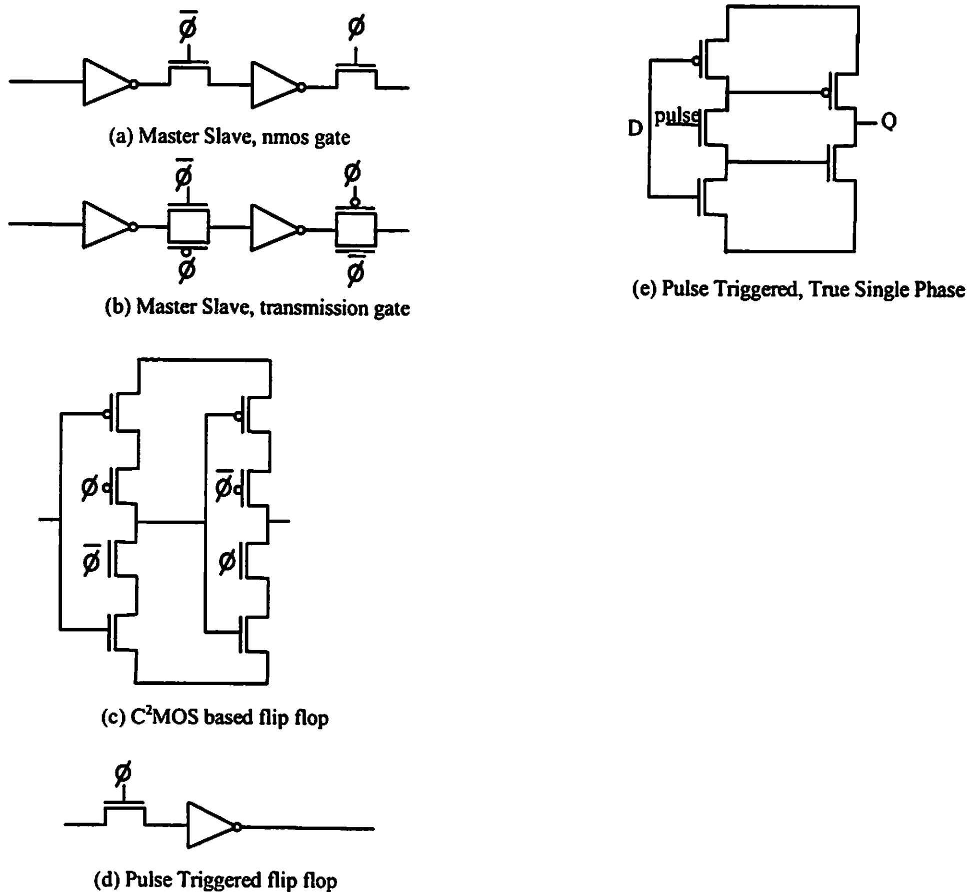


Figure 27: Dynamic Flip Flops

Table 6: Flip Flop Power Consumption.

Flip Flop	Power @ 1GHz
(a)	18uW
(b)	20.6uW
(c)	17.1uW
(d)	10.1uW

B.2 Custom Designed SMEM Array

A custom designed SMEM array with several flip-flop configurations mentioned above. The results are shown below in Table 7, which gives the area and power estimates of a 128 element SMEM based on the 4-element simulation. Comparing the performance of the custom designed SMEM to the standard cell based design, we see that we save approximately 50% in power and operate at twice the speed. The layout of the custom SMEM array was not done, so a comparison of the areas still needs to be evaluated. More details about the custom SMEM design can be found in [11].

Table 7: 128 element SMEM performance (custom design)

Flip Flop used (from appendix B)	Power@500MHz (mW)	Area (transistor widths,um)	Max Speed (GHz)
(a)	32	17760	2
(b)	31.2	20534	2

B.1.2 Pulse Triggered Latch Based Design Considerations

In order to save area, a pulse-triggered latch can be used as the basic storage element. This places stringent requirements on the pulse width in order for the system to work properly. Specifically, several banks of registers must operate with each register receiving an earlier clock relative to the previous register to prevent a race condition. At the boundaries between banks of registers a negative edge triggered flip flop must be placed to allow interfacing several banks. Figure 28 below shows a typical configuration of two banks of registers operating with n phases.

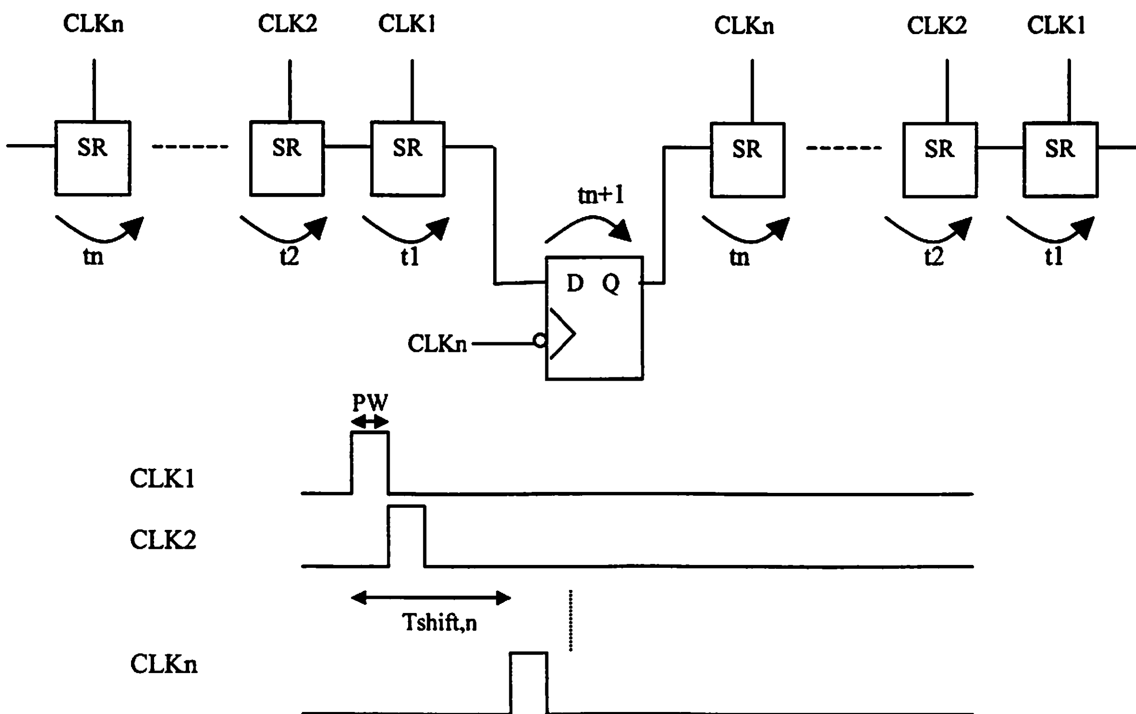


Figure 28: Typical configuration for 2 banks of pulse triggered latches

REFERENCES

[1] Avanti Coorporation. Corporate web site. http://www.avanticorp.com

[2] V. Dai, *Binary Lossless Layout Compression Algorithms and Architectures for Direct-write Lithography Systems,* Masters Thesis, U.C. Berkeley, June 2000

[3] T. Sandstrom, et. al. *New Laser Pattern Generator for DUV Using a Spatial Light Modulator.* Microelectronic Engineering, pp 23-29, September 2001.

[4] T. Sandstrom. *New Laser Pattern Generator for Maskless Lithography.* Half Moon Bay Workshop on Maskless Lithography, November, 2000.

[5] Etec Systems. *Mebes Extera Mask Writer Data Sheet* http://www.appliedmaterials.com/products/pdf/211654exara.pdf, September, 2001.

[6] N. Chokshi, Y. Shroff, W. G. Oldham, et al., "Maskless EUV Lithography", *Int. Conf. Electron, Ion, and Photon Beam Technology and Nanofabrication*, Macro Island, FL, June 1999.

[7] P. Rai-Choudhury. *Handbook of Microlithography, Micromachining, and Microfabrication.* Volume 1.

[8] S. Wolf, and R.N. Tauber, *Silicon Processing for the VLSI Era* Vol. 1, 2nd edition., 2000. pp. 411-3.

[9] V. Dai, Y. Shroff, M. Freed. *Decompression Circuitry for Direct Write Lithography Systems.* UC Berkeley EE225C class project. Oct. 2000. http://inst.eecs.berkeley.edu/~mfreed/ee225c/EE225C_Project_Final_Report.pdf

[10] International Technology Roadmap For Semiconductors. 2000 update. http://public.itrs.net/

[11] B. Wild, N. Wongkomet. *Data Handling Issues in Maskless Lithography*. UC Berkeley EE241 class project. March 2001. http://www-inst.eecs.berkeley.edu/~benwild

[12] MOSIS 0.25μm process parameters.
http://bwrc.eecs.berkeley.edu/classes/icdesign/ee241_s00/ASSIGNMENTS/TSMC025-n99y-params.txt

[13] A. Chandrakasan, W. Bowhill, F. Fox. *Design of High-Performance Microprocessor Circuits*. 3rd edition. IEEE Press, 2001.

[14] I.E. Sutherland, R.F. Sproull, *Logical Effort: Designing for Speed on the Back of an Envelope*, Advanced Research in VLSI, ARVLSI'91, Santa Cruz 1991.

[15] SSHAFT Reference Site.
http://bwrc.eecs.berkeley.edu/Research/IC_Design_Flow/ic_design_flow.htm

[16] L. Peterson, B. Davie. *Computer Networks*. Morgan Kaufman, 2000. pp. 96-102.

[17] V.Dai. *Compression Results for Thermometer Data Format*, notes. vdai@eecs.berkeley.edu

[18] S. Ross. *A First Course in Probability*. 5th edition. Prentice Hall, 1997.

[19] Texas Instruments. Network Processors white paper. Requested at http://www.ti.com

[20] S. Deorowicz, *Improvements to Burrows-Wheeler compression algorithm*. Software - Practice and Experience, vol.30, (no.13), Wiley, 10 Nov. 2000. p.1465-83.