Layout Decompression Chip for Maskless Lithography

Borivoje Nikolić, Ben Wild, Vito Dai, Yashesh Shroff,
Benjamin Warlick, Avideh Zakhor, William G. Oldham

Department of Electrical Engineering and Computer Sciences
University of California, Berkeley, CA, USA, 94720-1770

ABSTRACT

Future maskless lithography systems require data throughputs of the order of tens of terabits per second in order to have comparable performance to today's mask-based lithography systems. This work presents an approach to overcome the throughput problem by compressing the layout data and decompressing it on the chip that interfaces to the writers. To achieve the required throughput, many decompression paths have to operate in parallel. The concept is demonstrated by designing an interface chip for layout decompression, consisting of a Huffman decoder and a Lempel-Ziv systolic decompressor. The 5.5mm x 2.5mm prototype chip, implemented in a 0.18μm, 1.8V CMOS process is fully functional at 100MHz dissipating 30mW per decompression row. By scaling the chip size up and implementing it in a 65nm technology, the decompressed data throughput required for writing 60 wafers per hour in 45nm technology is feasible.

Keywords: Data compression, maskless, lithography, CMOS, digital design.

1. INTRODUCTION

As minimum feature sizes in CMOS technology scale, the cost of critical dimension masks dramatically increases. Mask costs in 90nm technology are exceeding 1 million dollars. An alternative to mask-based optical lithography is maskless lithography, where the layout data is directly written onto a wafer. Various approaches have been investigated, including e-beam, micro-machined mirror projection, and nano-jet printing [1]. To achieve the required 1nm edge placement with 25nm pixels in 45nm technology, a 5-bit per pixel data representation is needed, resulting in a total of over 500Tb of information on a 300mm wafer. To be competitive with conventional optical systems, any future lithography system should be capable of projecting one layer per minute, resulting in approximately 12Tb/s of raw data throughput. Adding the necessary redundancy and communications overhead to the data stream would likely increase the required throughput by another 25%, reaching 15Tb/s ranges.

This paper presents a maskless lithography interface and circuitry that provide the required throughput. It is designed to work with a micro-machined mirror array integrated on a chip exposed by extreme ultra-violet (EUV) light in a conceptual system as shown in Figure 1. In this approach, the conventional mask is replaced by a programmable one, which is reprogrammed between consecutive light flashes. Either storing or continuously feeding the required amount of data to the writer chip is not feasible. The key idea of the work presented here is the compression of rasterized data on a main storage unit and the continuous decompression on-the-fly, as the data is being downloaded on the writer [2]. With compression ratios of about 20, the chip input bandwidth would be 800Gb/s [3]. This is feasible in 90nm or 65nm technologies, which could be used for processing data with 45nm feature sizes.

The basic design of a data processing system capable of delivering tera-pixel data rates necessary to achieve next-generation maskless lithography is shown in Figure 2. This design consists of storage disks, a processor board with memory, and a decoder-writer chip with data-decoding circuitry fabricated together with a massive array of pixel writers. Layout data for all layers of a single chip is compressed off-line and stored on the disks. Before the writing process begins, only a single compressed layer is transferred from disks to the processor board memory and stored there. As the writers write a stripe across the wafer, compressed data is streamed from the processor board to the decoder-writer chip in real-time as needed. The on-chip decoding circuitry, in real-time, expands the compressed data stream into the data signals necessary to control the writers. The key challenge for such a system is the design of a high-throughput on-chip data decompression architecture and the circuitry that implements it. This paper demonstrates the design of such a high-throughput decompression chip. To simplify the solution, a binary interface to the writers using an SRAM array is implemented. This avoids the handling of 5-bit grayscale values that would require analog control of mirror positions

using analog memory. However this is not a good choice for a practical solution as it results in 6 times higher required data throughput (5-bit grayscale codes are replaced with 32-bit thermometer codes).

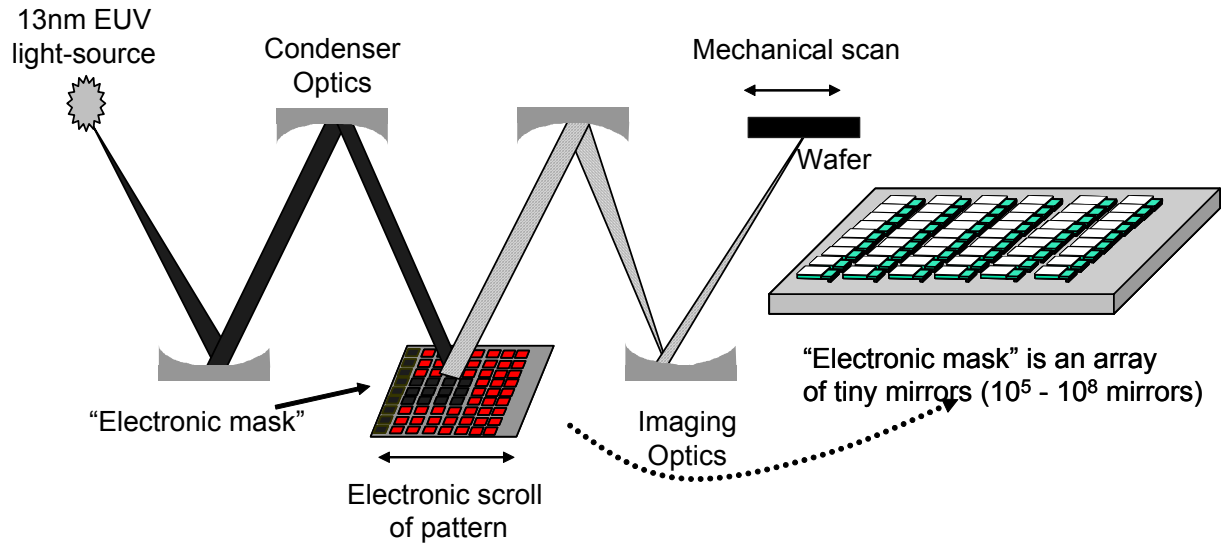


Figure 1. Maskless writing using micromirrors.

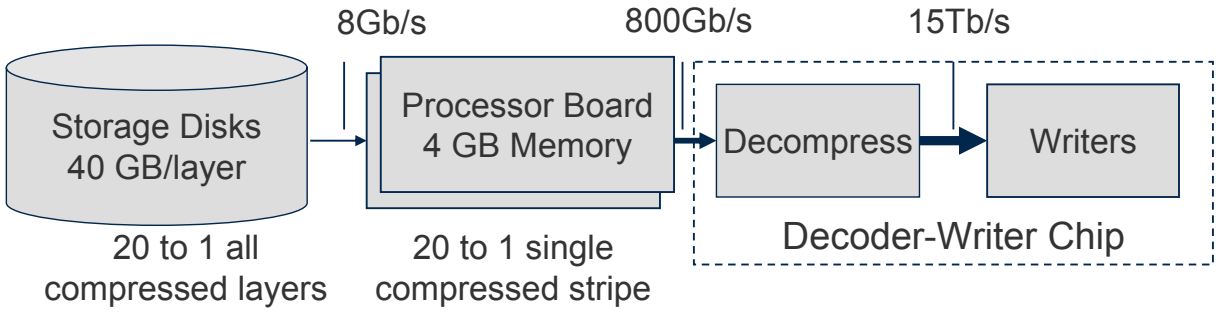


Figure 2. Architecture of a data-delivery system for maskless lithography.

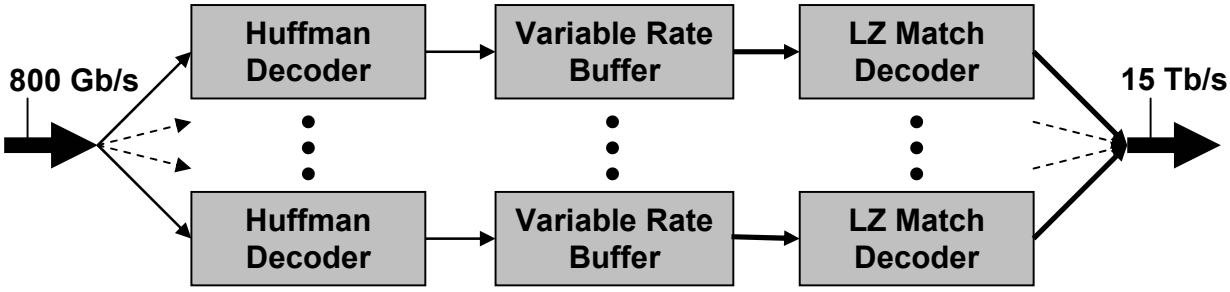


Figure 3. Multiple LZ77 decoders operating in parallel.

2. DESIGN OF A HIGH-THROUGHPUT DECODER

When the data compression and decompression algorithms are being designed, they are optimized for operation on a general-purpose microprocessor, and rarely a hard-wired hardware implementation is targeted. The primary difficulty of using compression to solve the data transmission problem associated with maskless lithography is that in order to be effective, the compressed representation must be decoded in a timely manner to sustain the required throughput, with reasonable computational and memory resources. To achieve this, a compression algorithm has to be designed, and then mapped into an architecture and an implementation that expands the compressed data at a rate of 800 Gb/s to 15 Tb/s of output data. We consider the design of a high-speed hardware decoder based on ZIP, the simplest compression algorithm presented in prior research [4] which achieves good compression on lithography data. The basic algorithm behind ZIP compression is Lempel-Ziv 1977 (LZ77) [5], and the design parameters of a LZ77 decoder are optimized to minimize circuit usage while maintaining compression efficiency.

2.1. Fast LZ77 Decoding Architecture

The design of a fast LZ77 decoder, shown in Figure 3, consists of two decoding blocks, a Huffman decoder, and an LZ match decoder. Because each decoder block has the potential to expand the data, and because the expansion factor varies in time with the compression ratio of the data, buffers are necessary to smooth both input and output data rates. In general, this would require three buffers, one in front of the Huffman decoder, one after the match decoder, and one between the two decoders. To reduce the buffering requirements, we choose an implementation of the Huffman decoding algorithm which observes a constant input rate, and we choose an implementation of the LZ77 decoding algorithm which observes a constant output rate. Consequently the variability in data rate is completely absorbed by a single buffer, capable of both variable input and output data rates.

For simplicity, a Huffman decoder with an input data rate of 1 bit per cycle is chosen. Also, a match decoder with a constant output rate of one pixel per cycle is chosen which corresponds to an output rate of 5 bits per cycle for 32 gray level data. To keep the operating frequencies within reasonable limits, a Huffman algorithm that decodes *m*-bits per cycle and match decoders that output *n*-pixels per cycle, are used, with a corresponding increase in hardware complexity. The simpler alternative which we adopt is to use *k* independent LZ77 decoders to multiply both input and output data rate by a factor of *k*. To accomplish this, the layout data must be divided into blocks and compressed independently, Figure 3, though this may come at some cost to compression efficiency as discussed in [4]. The number of decompression paths needed depends on the maximum operating frequency of the Huffman decoder and LZ match decoder of the final design.

2.2. Huffman Decoding

The essence of Huffman encoding is that it assigns shorter codewords to more frequent data, thus reducing the average number of bits required for representation. With layout data tested in [4] it achieves a typical compression ratio of approximately 5. This implementation uses the canonical Huffman table because of its simple representation that lends itself to a less complex decoder implementation. The algorithm description can be found in [6]. The Huffman decoder architecture is shown in Figure 4. The coded data is input sequentially into a shift register. For every bit that is shifted in, the counter is incremented. The output of the counter is used to address *mincode*, *maxcode* and *index* tables. The word in the shift register is compared to the output of the *maxcode* table. If the shift register word is less than the output of the *maxcode* table then the decoding is done. To get the decoded word, the shift register word is added to the output of the index table, and then subtracted from the output of the *mincode* table. This value is then used to address a symbol table which holds the decompressed symbols. To increase the decoding speed, the architecture is pipelined at the cut-sets shown in Figure 4, leaving only the symbol memory lookup time in the critical path.

When implemented in 0.18μm CMOS technology, the area of this Huffman decoder is about 6mm x 120μm. This large aspect ratio is chosen to allow operation of multiple parallel paths on the same chip. Scaling this design to 65nm technology would make its interfacing to the other parts of the decoder and the writer array feasible.

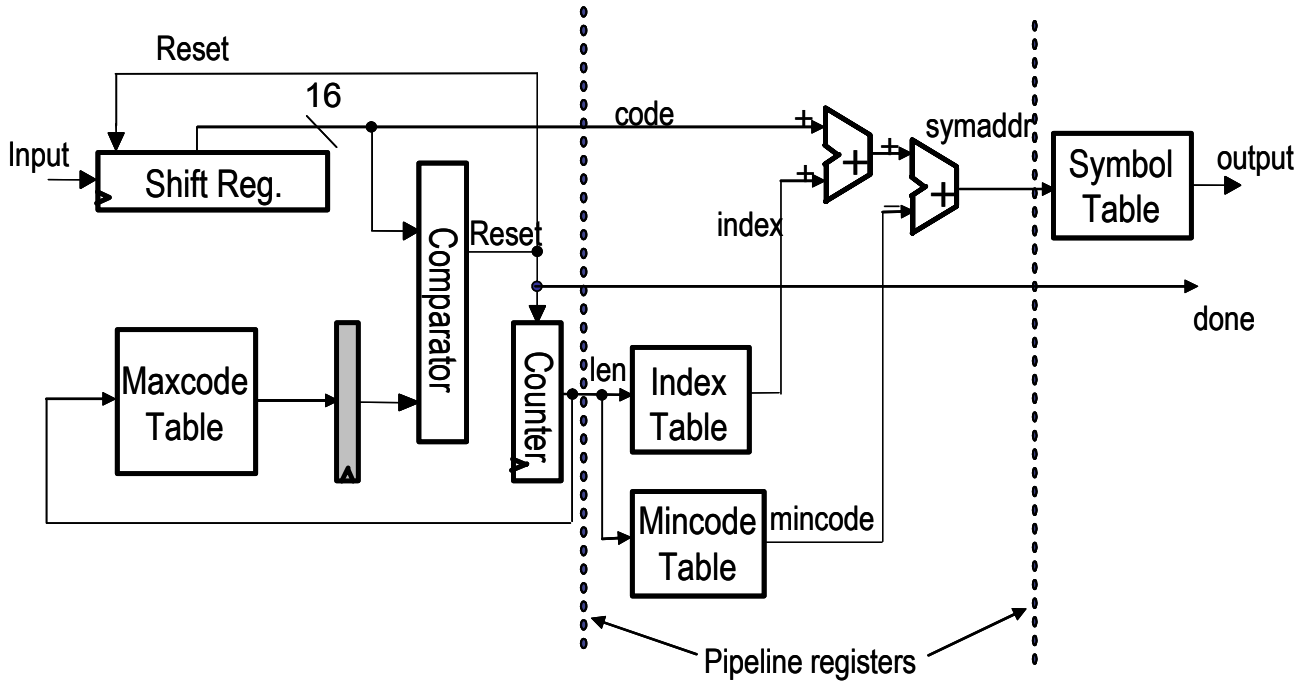


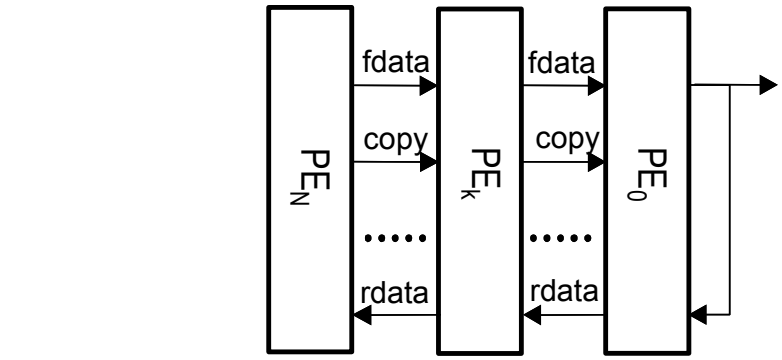
Figure 4. Block diagram of the Huffman decoder.

2.3. Lempel-Ziv Matching Algorithm

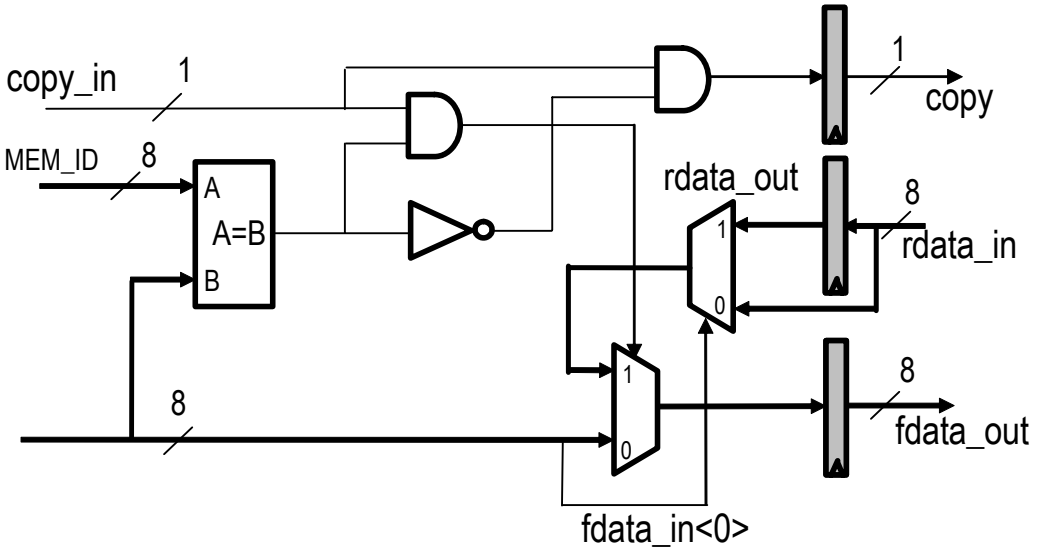
The Lempel-Ziv algorithm replaces repeating sequences of symbols with a pointer to a history buffer. This pointer indicates where in the buffer to retrieve the data (offset), and how many symbols to copy (length). Uncompressed symbols are literals. The architecture chosen for this design consists of a systolic array processor, where the data pass bi-directionally from one processing element to neighboring elements in a regular pattern [7]. Each processing element consists of one forward buffer and one reverse buffer and stores two 8-bit symbols. Data flows from left to right through the forward registers and then wraps back, flowing from right to left through the dictionary registers. Figure 5 illustrates the functionality of the systolic array processing elements. The decoding is done locally, avoiding any long wires to the history lookup buffer that would be unavoidable in the non-systolic implementation. To get better compression, the literal, offset, and length symbols are independently Huffman coded. To allow this, separate Huffman tables must be multiplexed in, depending on the current symbol being decoded. Furthermore, in the systolic implementation, a runlength decoder is used. This decoder simply sets a flag indicating when an <offset, length> pair is to be decoded by the processing elements. It transmits the offset symbol to the systolic array <length> times. If the input symbol is a literal then this is copied to the systolic array. Two critical parameters in implementing the LZ algorithm are the match length and the size of the buffer. Figure 6 shows the tradeoff between the compression ratio and the buffer size and match length. A buffer size of 1024 and a match length of 256 achieve the near optimum compression ratios for layout data [4].

The systolic implementation of the LZ algorithm provides redundancy for the writing process. Since the complete pixel exposure is achieved by multiple EUV light flashes simultaneously with cycling the data through the systolic array, the writing process can be designed to compensate for non-functional mirrors.

The layout of this LZ decoder with buffer length of 256 and match length of 256 is about 7.8mm x 64µm. This layout aspect ratio is chosen to allow integration of 1000 parallel decoding paths in a full-throughput chip in 65nm technology. It should be noted that LZ achieves better compression per unit area in the hardware implementation than the Huffman decoder. However the LZ algorithm alone does not achieve the desired compression ratio, so in this experiment we implemented both the LZ and the Huffman coding [4].



a)



b)

Figure 5. a) Systolic structure, consisting of processing elements, b) Processing element of LZ decompressor.

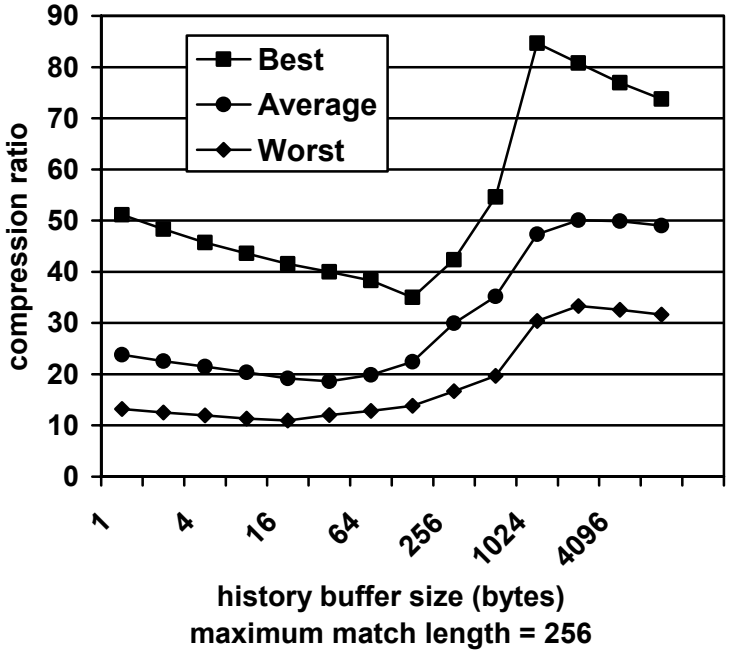
3. CHIP ARCHITECTURE

To meet the throughput requirements for this application, many decompression paths must operate in parallel on the pattern generator chip. The block diagram of each decompression path is shown in Figure 7. The main blocks are a Huffman decoder and a Lempel-Ziv decoder. An asynchronous FIFO is required in between these blocks because the blocks operate at different, time-varying rates. A CRC block is implemented at the end of each frame to check for any data errors. The data is framed into blocks consisting of the 1024 bytes of data, a 1-byte CRC check and an 8-byte synchronization block to synchronize the writer-interface circuitry to the writers. In the case of a micro-mirror based system, the writer-interface consists of a simple SRAM memory array.

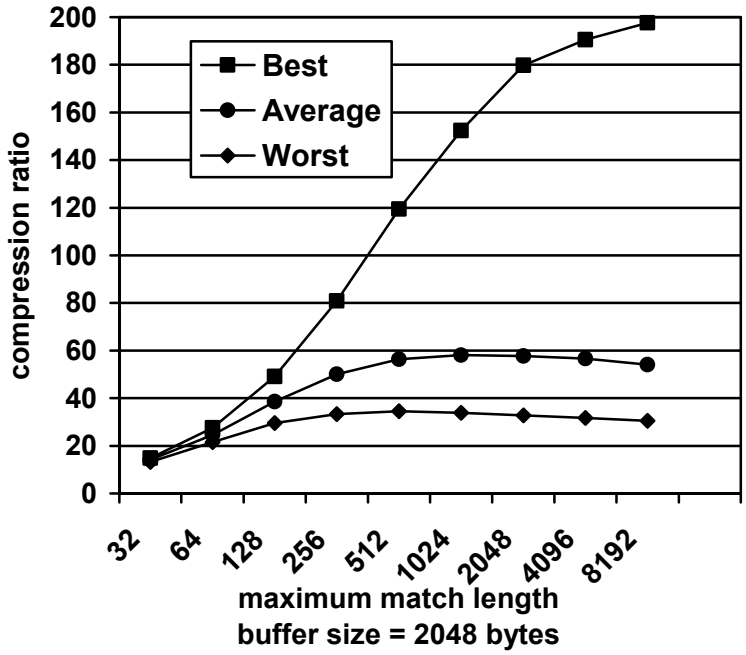
3.1. Design Process

The design is captured as a high level Simulink model. Datapath blocks are created using the basic building blocks in Simulink such as registers, adders, and multipliers. Memory blocks are created using Matlab M-functions. The design is functionally verified in Simulink and then directly mapped into the standard cell library using the SSHAFT automated design flow [8]. The datapath blocks in Simulink are mapped into Synopsys Module Compiler blocks. Module Compiler then builds the netlists for the datapath blocks using the standard cell library and generates a VHDL description that can

be simulated with the same test vectors used in Simulink to verify functionality of each of the datapath blocks. Memory blocks are created using memory generators. From the top-level dataflow graph in Simulink the decompression rows are mapped into standard cells and automatically placed and routed.



a)



b)

Figure 6. Compression performance vs. a) history buffer or b) match length for 16 KB uncompressed data.

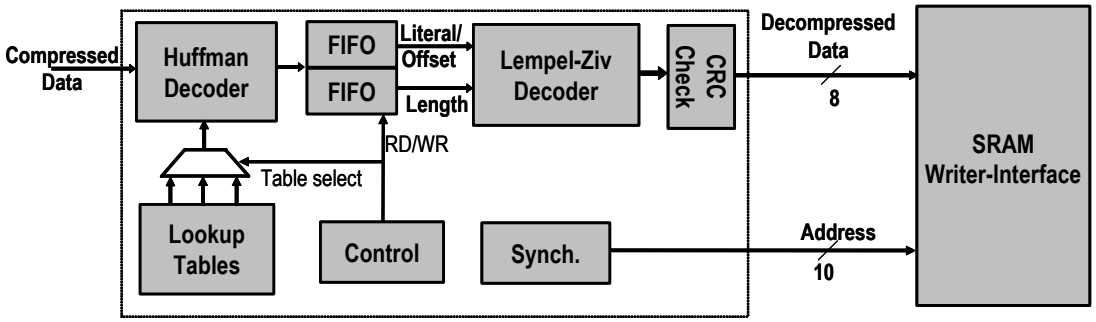


Figure 7. Decompressor row block diagram.

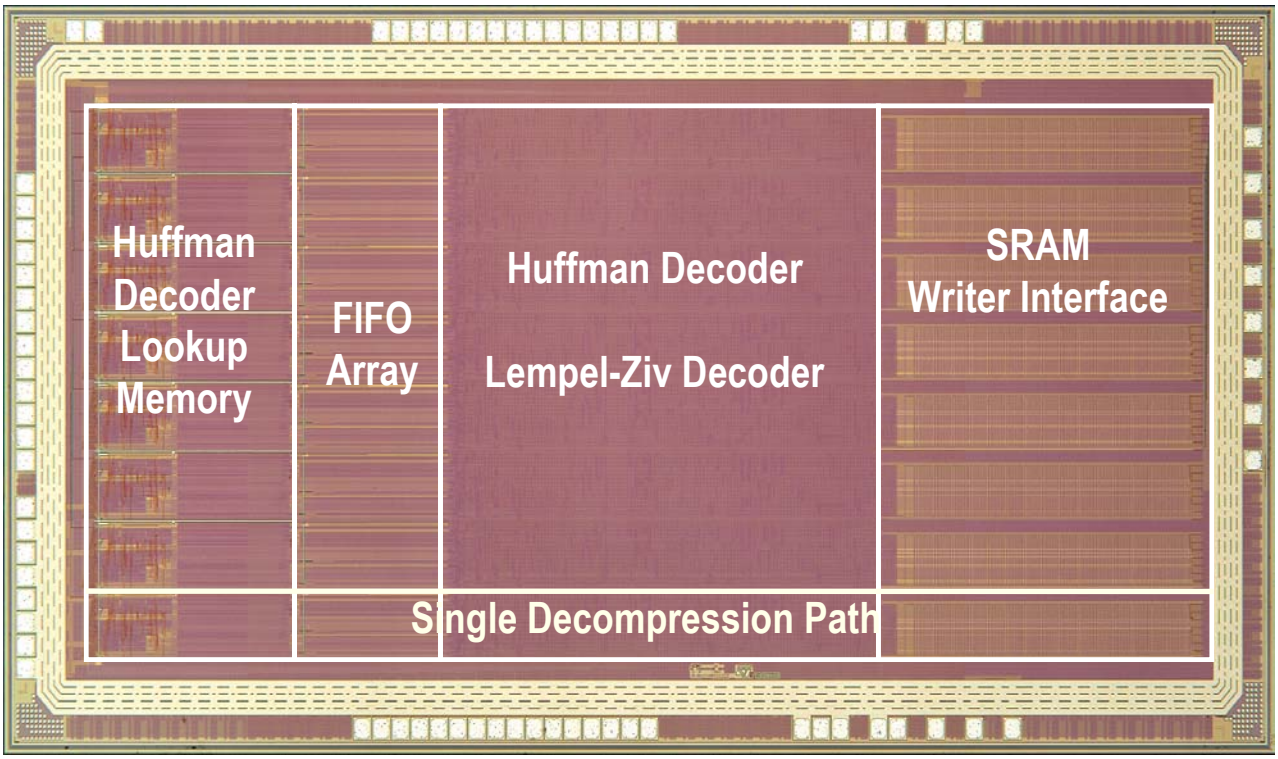


Figure 8. Die photo of the prototype chip.

4. EXPERIMENTAL RESULTS

To demonstrate the feasibility of this approach to data delivery for maskless lithography, a prototype chip in 5-metal 1.8V 0.18µm was designed. The chip consists of 8 parallel decompression paths, each with a Huffman block, an LZ block consisting of a 256-byte history lookup and a 256 word FIFO. All the FIFOs and buffers are imported as SRAM blocks. The total chip size is 2.5x5.5mm^2. The history buffer is limited to 256 bytes due to prototype chip area constraints. The chip is fully functional at 100 MHz dissipating 30mW per decompression row. The chip has been designed to operate at 500MHz, although the test setup limits the verification up to 100MHz. Each decompression path loads 8 bits of data per clock cycle; since the chip contains 8 identical paths, a maximum output throughput of 32Gb/s can be achieved. The die photo is shown in Figure 8. To achieve the required throughput, a complete chip implemented in 65nm technology for printing 45nm features would require two rows of 1024 parallel decompression paths, each

operating at 1GHz. The LZ symbols would be 16 bits, with appropriately increased buffer size. The decompressed data will be de-multiplexed and fed to two 16,384 x 8,184 SRAM arrays that will interface to the mirrors.

Table 1 compares the performance of the test-chip versus that of the full-scale chip. By scaling up the die size to occupy a full reticle and scaling down the feature sizes to use a 65nm technology, the required throughput is achievable.

5. CONCLUSIONS

A real-time layout decompression architecture and its implementation on a silicon chip were presented. The architecture consists of parallel decompression paths, where each path is composed of a Huffman decoder and a Lempel-Ziv systolic decompressor. A fully functional smaller scale prototype chip demonstrates the feasibility of this approach. A scaled up decompression chip would be able to deliver 32Tb/s of data to the array of 256 million mirrors. Besides the micro-mirror based approach, this architecture can be easily extended to other types of maskless lithography systems such as e-beam direct write, or to any other high data rate application where data compression is possible.

ACKNOWLEDGEMENT

This research is conducted under the Research Network for Advanced Lithography, supported jointly by the funding of the Semiconductor Research Corporation (01-MC-460) and the Defense Advanced Research Project Agency (MDA972-01-1-0021).

REFERENCES

[1] Y. Shroff, Y. Chen, W.G. Oldham, "Fabrication of parallel-plate nanomirror arrays for EUV maskless lithography", *Journal of Vacuum Science and Technology*, Nov. 2001.

[2] V. Dai and A. Zakhor, ``Lossless Layout Compression for Maskless Lithography Systems" in *Emerging Lithographic Technologies IV, Proceedings of the SPIE*, San Jose, California, March 2000, Vol. 3997, pp. 467-477.

[3] Rambus Redwood Interface, http://www.rambus.com/products/redwood/

[4] V. Dai and A. Zakhor, "Lossless compression techniques for maskless lithography data" in *Emerging Lithographic Technologies VI, Proceedings of the SPIE*, San Jose, California, March 2002, vol. 4688, p. 583-594.

[5] J. Ziv, and A. Lempel, “A universal algorithm for sequential data compression”, *IEEE Transasctions on Information Theory*, IT-23 (3), pp.337-43, 1977.

[6] J. Miano, Compressed Image File Formats, pp. 65-75, ACM Press, 1999.

[7] C. Chen, C. Wei, “VLSI design for LZ-based data compression,“ *IEE Proc. – Circuits, Devices and Systems*, vol. 146, no. 5, pp. 268-277, Oct. 1999.

[8] W. R. Davis, et. al., "A Design Environment for High Throughput, Low Power Dedicated Signal Processing Systems", *IEEE Journal of Solid-State Circuits*, Mar. 2002.

Table 1: Performance comparison: The test chip, the test chip scaled down to 65nm technology, 65nm design expanded to full reticle.

	Prototype in 0.18µm CMOS	Prototype scaled to 65nm	Full-scale writer
Technology	180nm	65nm	65nm
Input bandwidth	800Mb/s	N/A	400Gb/s
Decompression paths	8	8	2 x 1024
Path pitch	240µm	80µ	40µm
History buffer	256	256	1024
Max throughput	32Gb/s	96Gb/s	32Tb/s
Mirror array	8 x 8 x 1024	8 x 8 x 1024	2 x (16 x 1024 x 8,184)
Mirror size	3µm x 3µm	1µm x 1µm	1µm x 1µm
Chip dimensions	2mm x 5mm	0.7mm x 1.7mm	24mm x 26mm
Power	560mW	60mW	~15W