TECHNICAL MEMORANDUM

Department of

Computer Science

CALIFORNIA INSTITUTE OF TECHNOLOGY
Computer Science

3857:TM:80

VLSI Architecture and Design

by

Lennart Johnsson

Presented at
National Electronics Conference
Chicago, Illinois, October, 1980

The research described in this paper was sponsored by the
Defense Advanced Research Projects Agency, ARPA Order number 3771,
and monitored by the
Office of Naval Research
under contract number N00014-79-C-0597

VLSI ARCHITECTURE AND DESIGN

Lennart Johnsson

Computer Science
California Institute of Technology
Pasadena, California 91125

Abstract

Integrated circuit technology is rapidly approaching a state where feature sizes of one micron or less are tractable. Chip sizes are increasing slowly. These two developments result in considerably increased complexity in chip design. The physical characteristics of integrated circuit technology are also changing. The cost of communication will be dominating making new architectures and algorithms both feasible and desirable. A large number of processors on a single chip will be possible. The cost of communication will make designs enforcing locality superior to other types of designs.

Scaling down feature sizes results in increase of the delay that wires introduce. The delay even of metal wires will become significant. Time tends to be a local property which will make the design of globally synchronous systems more difficult. Self-timed systems will eventually become a necessity.

With the chip complexity measured in terms of logic devices increasing by more than an order of magnitude over the next few years the importance of efficient design methodologies and tools become crucial. Hierarchical and structured design are ways of dealing with the complexity of chip design. Structered design focuses on the information flow and enforces a high degree of regularity. Both hierarchical and structured design encourage the use of cell libraries. The geometry of the cells in such libraries should be parameterized so that for instance cells can adjust there size to neighboring cells and make the proper interconnection. Cells with this quality can be used as a basis for "Silicon Compilers".

Introduction

The integrated circuit technology has evolved rapidly. The number of components per chip has approximately doubled every year [1]. The cost of definition, design and layout of a chip set has almost remained constant measured in dollars per gate. Without a significant reduction in this cost a VLSI chip would cost tens of millions of dollars beforereaching the production stage. It would also take in the range of 10 - 100 years to get there.

New design disciplines, notations and languages for design as well as improved computer aided design and design automation are needed. The key issues are complexity managemant and correctness. Both these issues have several flavors that results in different requirements on design disciplines, notations and tools.

For the effective use of silicon area it is desirable to develop new architectures. The physical characteristics of VLSI technology is part of the reason for this desire.

This paper is written with nMOS technology in mind but many viewpoints are not technology specific. The physical characteristics of integrated circuit technology that are of particular interest to VLSI are reviewed. Architectural implications of these characteristics are considered. Strategies for dealing with some of the problems involved are proposed and desirable qualities of design disciplines and notations stated. Finally an example of high level design automation is given.

Characteristics of VLSI Technology

VLSI is here understood as a statement about complexity and not feature size as such. Chips of normal size manufactured with a feature size of one micron or less are considered as VLSI circuits. Such circuits will have in the order of one million transistors. The most complex chips manufactured today contains in the order of one hundred thousand transistors. Feature sizes are typically in the range 3-5 microns.

To appreciate the change in complexity the following analogy due to Seitz [2] may be helpful. The conductors on a chip are pictured as streets on a city roadmap withspacing between conductors corresponding to blocks. In the early 1960's the complexity of a chip equalled that of a small city (Pasadena), in the late 1970's it was like that of the Los Angeles basin and is in the mid 1980's expected to be comparable to the entire states of California and Nevada covered with streets at urban density. In the limit of the technology [3] the complexity will be like that of the North American Continent covered with streets at urban density. In order to effectively deal with a complexity of this kind a new approach to chip design is necessary.

Today the minimum gate length of transistors and the width of wires are determined by lithography and process technology. Physical limits are reached at about a quarter of a micron. In scaling feature sizes it is desirable to maintain the same relative topology. Hence the thickness of the various layers are scaled as the feature sizes are. Asensible way of scaling is also to maintain a constant electric field in the gate. This means that voltage levels will decrease in the future. To account for statistical variations in threshold voltages the supply voltage has to be higher than direct scaling would suggest [4]. A supply voltage of 0.7 V for quarter micron devices is predicted to be proper.

An electronic circuit that is designed to perform a computation consists of transistors (switches) and wires connecting them. The transition time of the switches decreases proportionally to the feature size. The wires have resistance and capacitance associated with them. Thus they introduce a delay. The resistivity per unit length increases

proportionally to the square of the scaling factor. The capacitance stay constant per unit length of the wire. Hence, if the length of a wire is scaled with the featuresizes the delay will stay constant but if the length of a wire is kept constant the delay increases proportionally to the square of the scaling factor. Wires will become slower in relation to switching devices. For a wire of constant physical length the ratio increases with the third power of the scaling factor.

Algorithms and Architecture

One of the consequences of decreased feature sizes is that the chip area essentially is determined by wiring needs. The actual computing elements can largely be "hidden" under the wires. Hence architectures that offers for instance a fairly complex computation on local data (no additional wiring) need not require more chip area than an architecture with only very simple operations per communication event.

Efficient use of silicon is important. One measure of efficiency is the fraction of time the devices on a chip are actively participating in the computation. One of the goals in the development of concurrent computing machines is to achieve high performance through the concurrent use of several processors that eventually can be put on a chip. Long wires should be avoided since they decrease the performance of the system. They also havean impact on the total chip area. Short wires makes it possible to complete the computations fast, i.e. to have short cycle times in synchronous systems. The desire to have few and short wires constrains processor intercommunication.

Locality of design is of premium importance. This has algorithmic and architectural consequences. It is desirable to devise algorithms and find architectures such that a meaningful computation can be performed by passing the result from one computation on to its nearest neighbors. Recognizing the wiring constraint and the fact that silicon is planar the number of nearest neighbors is small. The communication pattern of the algorithms and the machine becomes very important. A perfect match is the ideal with respect to performance. Precedence relationships in algorithms can drastically reduce achievable concurrency.

Several architectures suitable for VLSI technology have been proposed. Some of them can be considered as special purpose machines while others like the hexagonal array proposed by H.T. Kung et al [6] and the orthogonal array proposed by S.Y. Kung [7] are fairly general. An interesting structure is the Tree Machine studied by Browning [8,9]. It can be considered either as a special purpose machine and be tailored to a particular application or be designed with the objective of being general. In the latter case the tree is made balanced. There is also a question as to which fan-out to use. If the machine or physical tree has a fan-out larger than the algorithm to be executed on it a number ofprocessors that grows exponentially with the depth of the tree will not be used. On the other hand if the fan-out of the physical tree is smaller than that of the algorithm the larger fan-out has to be simulated on the machine. The simulation often results in a number of processors not being used ideally. For a given mismatch in fan-out less processors are wasted if the physical tree has the smaller fan-out. This argument makes binary trees attractive. A binary tree can also very economically be layed out on a planar surface, Figure 1.

Browning [9] devised a large number of algorithms with distributed control for the Tree Machine. Most of the algorithms are related to graphs. The time complexity for the algorithms studied is $O(n^2)$, where n is the number of nodes in the graph. In most cases the number of processors engaged in the computation is of the same order as the time complexity of the problems on a sequential machine.

One goal in designing processors for a concurrent computing machine in VLSI is to make each processor small but to have many of them instead. For the set of algorithmsin [9] the processors met this goal. A processor with 2048 bits of storage, 16 8-bit registers for data, an instruction register, program counter, address logic, ALU and port handlers would suffice. The instruction set proposed has seven control flow, four communication and three data flow instructions. The two remaining instructions (4-bit format) are used as escape codes to a second 4-bit word specifying either a unary or a binary operator.

Programming a concurrent computing machine with explicit knowledge about its communication structure is not an easy task. It is necessary that the notation or language used rules out some of the possible mistakes and makes program verification not too difficult. Several notations have been proposed [9,10,11,12]. Another view on the programming issue is that where the program is written without any knowledge about the machine architecture but in such a notation that the concurrency inherent in the problem to be solved can be exploited to the extent possible on the machine on which the program is executed. Applicative or functional programming languages [13] are intended to serve this purpose. The Homogeneous Machine, Locanthi [14], is based on a functional programming notation, dynamic allocation of computational tasks to processors in a concurrent computing environment and garbage collection in such an environment.

One of the key problems in circuit fabrication is defects. Decreasing feature sizes will make this problem of even greater concern in the future. Typically chips that are not functioning correctly are discarded and hence contributing to low yield. Fault tolerance techniques adresses the problem of making a system containing defective parts behave correctly viewed from the terminals of the circuit or device considered. Using fault tolerance techniques at the chip level serves the twofold purpose of increasing yield and the ability for the chip to deliver a correct function during future static or intermittent faults. Hierarchical Redundant Memories, Barton [5], are fault tolerant memories based on error detection and correction organized hierarchically. Hamming codes were used by Barton. The elementary parts of the system are storage arrays and coder/decoders. With modest increase in area a substantial improvement in yield is possible. Also, larger memory circuits are economically feasible if redundancy techniques are used.

The concept of locality as discussed above on the architectural level is also relevant on the circuit level. It is of interest to devise algorithms for low level functions such as floating point add and multiply that do not exhibit global properties. Some of our recent design efforts have made it clear that an understanding has yet to be developed as to which properties of circuits and low level functions inherently prohibits an entirely local design in a planar medium. It is necessary to gain experience through the design, layout, fabrication and testing of alternative ways of implementing functions typical in electronic

systems. The Multi Project Chip (MPC) facility available to ARPA sponsored research establishments is a very valuable asset in this regard. Efforts are now being made notably by XEROX PARC but also by others to create a library of good VLSI designs outof the designs continuously submitted to the MPC facility by the approximately 30 organisations now designing chips according to the rules proposed in [4].

Design disciplines

In this paragraph design methodologies will be discussed with emphasis on electrically-logically correct operation.

Correct timing is fundamental for the proper operation of any digital electronic circuit. It has been pointed out above that wires introduce a delay. Some figures may be helpful [2]. In nMOS technology with 5 micron minimum feature size the delay caused by a 1 mm long wire is as follows: polysilicon 2 nsec, diffusion 1 nsec and metal 0.001 nsec. The delay caused by a 10 mm wire is a factor of 100 larger. Scaling down feature sizes by a factor of 10 causes delays to be multiplyed by a factor of 100 so that for a metal wire we would get a delay of 10 nsec. At the same time the gate delay has decreased to about 0.025 nsec. Thus even for a wire in metal 10 mm corresponds to a delay equal to 400 times the transit time in a gate. The length of a wire introducing a delay equal to onetransit time is 0.01 mm in polysilicon, 0.02 mm in diffusion and 0.5 mm in metal.

The properties of wires in submicron technology makes synchronous systems less tractable because the clock period has to be long. This is due to the fact that the clock skew will be of the order of 100 transit times and the convention in synchronousdesign to assure that signals can propagate from any place on a chip to any other place on the chip within one clock cycle.

Self-timed design, Seitz [2, chap. 7 in 4] is a methodology for the design of asynchronous systems. Self-timed systems consists of elements interconnected according to certain rules that assure that the self-timed property is maintained during composition. Sequence and time are related to each other only in the interior of elements. The terminal behaviour is designed to be in accordance with the function desired. The correct sequential behaviour of a system is obtained through the proper interconnection of elements. The time required to compute the output once all the inputs are applied depends on the delays imposed by the elements and the interconnection delays.

The distribution of clock signals is but one function that has to be done properly to assure a correct operation of a circuit. Signals on a chip may deteriorate for instance by passing through too many pass transistors. This can cause the circuit to generate a faulty output. One remedy for this problem is to restore the signals when necessary. Restoring logic is a class of logic that with proper accompanying design rules as developed by Rem and Mead [15] assures that systems composed of restoring logic will always be operating with well defined signals and can themselves be considered as a restoring logic circuit. The notation developed for describing the functional behaviour of circuits and the syntax proposed assures that every syntactically correct description specifies a restoring logic circuit. The notation can relatively easy be translated into transistor diagrams for CMOS.

Complexity management

In VLSI the need for formal ways of managing the complexity in chip design becomes urgent. It is the general understanding that the human brain has a short term memory that is capable of handling at about 7 different ideas or concepts simultaneously. The approach that humans typically take to complex problems is that of divide and conquer which matches the capabilities of the short term memory well.

Hierarchical design, Rowson [16] is based on two important concepts: abstraction and composition. Abstraction is used at the various levels of the design to limit the amount of information. Only the information necessary to properly perform the design task at a given level shall be present. The detail of lower levels are refered to those levels where it is needed for making the correct decisions about the design.

Composition is used to create a whole out of pieces. In carrying out a design task several composition rules are used. In the previous section composition rules to make the self-timed property propagate up through a hierarchy was mentioned. Another rule mentioned was that for propagating the restoring logic property. There are also composition rules related to the layout (geometry) of designs.

A design carried out in a truly hierarchical fashion can be characterized as consisting of two kinds of entities, the elements being interconnected and the interconnect structure itself. Using a terminology [16] related to chip design a leaf cell is defined to be an atomic unit that can not be subdivided further. Composition cells define interconnections between cells. A leaf cell may be very simple like for instance a single transistor or a set of wires or as complex as a PLA. The leaf cells are the functional units out of which the system function shall be created. The interconnection specified in the composition cell is implemented through a composition rule. It is important that the rule does not introduce any new functionality.

Following these guidelines for hierarchical design have several advantages. For instance given alternative descriptions of the leaf cells such as their layout or their logical behaviour and the corresponding composition rules a layout and a logic simulation can be generated from the same description of the hierarchy. The simulation will correctly correspond to the layout if for each cell the description of the logic behaviour correctly matches the layout and the composition rules are correct. Hence the problem of proving correctness reduces to proving correctness of each leaf cell and composition rule. This is of considerable importance.

Hierarchical design is not necessarily a "top down" design discipline. It is of importance to notice this fact since it is certainly desirable to be able to use leaf cells previously defined. With a strict "top down" approach this may not be possible. However to be able to use cells previously defined it is not sufficient to allow a certain degree of a "bottom up" type approach. The cells also have to be designed according to certain standards.

Creating a useful library of cells is not an easy task. Typically the implementation of the function a cell performs is technology dependent. The dependency may be such that it is not possible or desireable to implement the same function in different technologies. Even within the same technology there may be significant differences in cell characteristics. For instance cells may be self-timed or not or having the restoring property or not. Some properties can be parameterized and algorithms found that relate the parameters to each other or to precisely stated design goals. Such goals may be high performance or low power of the circuit to be designed. Other desirable properties of cells will be discussed in the following.

Structured design, Mead and Conway [4], is another way of managing the complexity involved in designing VLSI chips. Hierarchical and structred design can with great advantage be used together. Structured design focuses on the communications and wiring problems, which are of premium importance in nMOS VLSI technology. Chips designed according to the stuctured design methodology have a very regular appearence. A good example is the OM chip set [4].

The use of floor plans is an important step in the planning of a chip. Traditionally the size and shape of areas for various functions that shall be part of a chip set areestimated by experienced designers. This information is used to divide the over all function among the chips and to place the functions on the chips in such a way that the area is minimized. The floor plans forms the basis for estimating chip size and performance.

In structured design the use of floor plans is more extensive. In creating a floor plan the primary consideration is the flow of information, data and control. The various functions are placed in such an order that the wiring is minimized, Figure 2. The cells are designed so that they "abut" perfectly, i.e. the wires that shall be interconnected appear at the edges of neighboring cells exactly at the same location. Cells that may appear in arrays are designed in such a way that placing instances of the cells next to each other automatically provides the necessary interconnection. The amount of random wiring is kept to a minimum.

The emphasis on the information flow in structured design typically means that buses are routed through cells instead of around them. With "abutting" cells this strategy may result in a smaller area than routing wires around the cells. With suitable algorithms and arrangement of wires the desired computation can be performed as the information passes by. A good example of a design of this kind is the barrel shifter in the OM data path [4].

Computer Aided Design and Design Automation

The design techniques outlined above will help in the design of chips that operate correctly and reduce the cost of design. The techniques form a good basis for automating the design process. Efficient design tools will not only reduce design cost but also the calendar time for the design of chips.

There are many decisions that have to be made before a layout of a chip can be generated. Some of the decisions concern the architecture. It is very difficult to forsee all

the consequences with respect to for instance power and area requirements of such decisions. Evaluating a few alternative designs by completing the entire design process for each before the decision is made about which to use requires efficient high level tools. It is desirable to have tools that accepts a behavioural description of the chip and can be used to generate layouts, simulations, etc.

A notation for the behavioural description of a chip shall support a hierarchical design discipline. It has to have an abstraction capability and proper means of dealing with composition and communication. The notation shall also help in writing correct descriptions and make verification a reasonable task. To our knowledge there does not yet exist such a notation. An early attempt in this direction is Communicating Sequential Processes (CSP) [10]. Browning [9] has modified CSP to better support a hierarchical design style and better match communication on chips. Approaches that currently seem to hold good promise are those taken by Milne [11], Chen and Mead [12] and Rem and Mead [15].

In automating the design process one should obviously start with those parts that are least creative since the creative process is not well understood. A sensible goal in design automation is that the tools shall generate systems and layouts that are competitive with entirely manually created systems or layouts.

Low level cell design is an example of where in most cases it is difficult by means of automation to come anywhere near the results that human ingenuity generates. Futhermore each cell is often of very low complexity. This means that they are designed both quickly and correctly by people. Hence there are several good reasons for leaving cell design to people. This is however not to say that computer based tools should not be provided for this part of the design process.

In producing layouts for cells much time can be saved by not having to worryabout all the detail of the geometry and associated design rules. Again a proper abstraction and supporting tools is of great value. Sticks notation offers such an abstraction. A Sticks diagram defines the constituents of a cell, their interconnection and a set of constraints on their placement. A Sticks diagram does not specify geometrical detail or dimensions. It can form the basis for programs performing various tasks that have to be completed in order to arrive at a layout. Such programs can for instance flesh out the Sticks representation into a layout. Other programs can be used to compact the layout. Compaction programs shall know about geometric design rules so that only layouts obeying these rules are generated. It is also of interest to augment the Sticks notation with a capability to specify for instance line widths and transistor sizes.

In order to make a cell reusable in a structured design discipline its geometrycan not be fixed. Width and height as well as wire and transistor dimensions has to be adjustable. Cells in the Bristle Blocks system, Johannsen [17] have this property.A similiar approach is taken in [16]. Connection points are specified as to their location and nature. Thelocation is given relative to some fixed point of the cell and is computable to allow for the matching with neighboring cells. This matching is made while obeying geometric design rules.

The Bristle Blocks system is designed to handle various representations of cells and systems. Presently the following are included: geometry (layout), Sticks, transistors, logic, text, simulation (logic) and block. The block description is used to draw a block diagram that shows buses and core elements of the chip.

The Bristle Blocks system was the original Silicon Compiler. Our present experience of this kind of compiler is related to chips that have a floor plan that consists of a core of computing elements controlled by an instruction decoder and pads around the edge. The core elements have communicated with either of two buses that run through the elements. The control signals have been derived from microcode. A two phase clock has been used. The area of the layouts produced by the compiler have been within 10% of the area achieved by hand layouts. Sometimes the compiler generates the more compact layout of the two alternatives.

The Silicon Compiler takes care of all wire routing, dimensioning of power and ground wires, timing, adds properly sized drivers, buffers etc. The input to the compiler consists of three sections. The first section defines the microcode, instruction width and partioning into fields. The second section defines the width of the data word and the buses running through the core of the chip. The last section defines the elements of the core and provides necessary parameters for these elements. The execution time for the compilation of a small processor chip is in the order of a few minutes. For a large chip a few tens of minutes should be expected.

Conclusions

Two concepts that VLSI technology will make of prime importance are locality and abstraction. The desire for locality will make concurrent computing machines with many processors on a chip tractable. The behavioural description of this kind of machines makes new notations desirable. Such notations shall reflect the locality of communication, treat the synchronisation problem and make verification of correctness a sensible task.

The concept of abstraction is crucial in managing the complexity involved in VLSI design. A notation for description of the behaviour of a chip or system shall also have anabstraction capability to support hierarchical and structured design. These two techniques directly adress the complexity management issue. The experience with them is good so far. These techniques also form a good basis for computer aided design and design automation.

The idea of generating a chip layout from a high level description of its behaviour is a viable one. Given the instruction set of a microprocessor it is today possible to generate a layout within a few days and several minutes of computer time by the use of a Silicon Compiler. This is a very encouraging result. The design of VLSI chips may not be a prohibitive task but economically feasible.

Acknowledgement

The research reported here has been partially supported by the Defense Advanced Research Projects Agency unnder contracts #N00123-78-C-0806 and #N00014-79-C-0597.

References

1. G Moore. Are we really ready for VLSI. *Proceedings of the Caltech Conference on VLSI*, 1979.

2. C Seitz. Self-timed systems. *Proceedings of the Caltech Conference on VLSI*, 1979.

3. B Honeisen and C Mead. Fundamental limitations in microelectronics-I. MOS technology. *Solid-State Electronics*, vol. 15, 1972, pp. 819-829.

4. C Mead and L Conway. *Introduction to VLSI systems*. Addison-Wesley, 1980.

5. T Barton. *A Fault-Tolerant Integrated Circuit Memory*. Ph.D. thesis, Computer Science, Caltech, 1980.

6. H T Kung and C E Leiserson. *Algorithms for VLSI Processor Arrays*. In [4].

7. S Y Kung. VLSI Matrix Computation Array Processor. *The MIT Conference on Advanced Research in Integrated Circuits*.

8. S A Browning. A Tree Machine. *Lambda*, vol. 1, no. 2, pp. 31-36, 1980.

9. S A Browning. *The Tree Machine: A Highly Concurrent Computing Environment*. Ph.D. thesis, Computer Science, Caltech, 1980.

10. C A R Hoare. Communicating Sequential Processes. *CACM*, vol. 21, 1978, no. 8, pp. 666-677.

11. G Milne. *An Algebraic Approach to Concurrency*. Lecture notes, 1980.

12. M Chen and C Mead. *A Notation for Designing Concurrent Systems*. Internal working document, Computer Science, Caltech, 1980.

13. J Backus. Can Programming Be Liberated from the von Neuman Style? A Functional Style and Its Algebra of Programs. *CACM*, vol. 21, 1978, no. 8, pp. 613-664.,

14. B Locanthi. *The Homogeneous Machine*. Ph.D. thesis, Computer Science, Caltech, 1980.

15. M Rem and C Mead. *A notation for designing restoring logic circuitry in CMOS*. Internal working document, Computer Science, Caltech and Eindhoven University, 1980.

16. J Rowson. *Understanding Hierarchical Design*. Ph.D. thesis, Computer Science, Caltech, 1980.

17. D Johannsen. Bristle Blocks: A Silicon Compiler. *Proceedings of the Caltech Conference on VLSI*, 1979.