VLSI COMPUTATION

H.T. Kung

Rapporteurs: Mr. R.C. Millichamp
Mr. D. Mundy

1. Issues in Special-Purpose VLSI Designs

A walk-through of major steps in the design of a special-purpose chip, illustrated by a concrete example.

Identification of important issues in special-purpose design, such as modular design, concurrency and communication, balancing computation with I/O, etc.

Principle of systolic architectures.

2. Systolic Architectures for VLSI Design

Examples of "good" chip architectures for convolution, filtering, matrix arithmetic, sorting, and several other important computations in signal and image processing, as well as in data processing.

Some general guidelines for deriving good algorithms and architectures for chip implementations. Useful building blocks for the construction of special-purpose systems.

3. VLSI Complexity Theory

Models for VLSI computation. Area-time tradeoffs; Rayout complexity and area-efficient layout and packaging schemes. I/O complexity and impacts of pin bandwidth limitation on designs.

1. Lecture 1: Issues in Special-Purpose VLSI Designs

This lecture reviews important issues in the design of special-purpose VLSI chips in general. The actual presentation will include a walk-through of major steps in a concrete design.

1.1. Understanding the Application

The first step in the development of a special-purpose VLSI system is to define the task. Ideally we hope to identify some system bottleneck to which a relatively inexpensive VLSI solution can improve the overall system in a substantial way. A typical area that could benefit from VLSI solutions is composed of those low level, compute-intensive, operations which are repetitive and well-understood. For example, in signal processing such an operation could be the inner product operation,

$$\sum_{i=1}^{n} a_i \cdot b_i.$$

The speed and accuracy of a signal processor is often dominated by the calculation of this operation (see, e.g., [27]). Thus it is worthwhile to build special-purpose chips for performing the inner product operation. Indeed, several commercially available chips have been built for this purpose, and found to be extremely cost-effective. Examples include the TRW multiplier-accumulator chips [25] and the NEC signal processor chip [23]. Other important operations that are frequently performed in signal and image processing include the FFT and convolution computations. Again, special-purpose chips such as the AMI 32-point FFT chip have proven to be useful. Choosing a proper task to be implemented in chips is most fundamental to the success of the special-purpose approach; it requires a good understanding of computational needs in a given application area.

1.2. Simple and Regular Design

Of genuine concern is the cost effectiveness of the special-purpose approach. The cost of a special-purpose system must be low enough to justify its limited applicability. For VLSI chips the dominant costs are typically on design and testing. If a design can truly be decomposed into few types of simple modules which are used repetitively with simple interfaces, then great savings in design and testing costs can be achieved [22]. In addition,

the resulting modularity makes the design easily adjustable to various design goals. Thus it is important that VLSI designs be simple and regular.

1.3. Concurrency and Communication

VLSI components based on low power technologies such as MOS are likely not much faster than their TTL counter parts of 10 years ago (see, e.g., [24]). Generally speaking, the power of VLSI comes from its potential of providing a large number of simultaneously executable processing elements, rather than fast components. Thus the performance of a VLSI chip is closely tied to the degree of concurrency in the underlying algorithm. When a large number of processing elements work simultaneously, their co-ordination and communication becomes non-trivial. The communication problem is especially significant for VLSI where routing costs dominate the area and power required to implement a circuit in silicon. The issue here is to design algorithms that support high degrees of concurrency, and in the mean time employ only simple and regular communications and controls so that their chip implementations would be easy and efficient.

1.4. Balancing Computation with I/O

When a special-purpose device is attached to a host from which it gets input data and to which it output results, I/O costs play an important role in the overall system performance. The ultimate performance goal of a special-purpose system is (and *should be no more than*) that of achieving a computation rate that balances the available I/O bandwidth with the host.

The pin bandwidth limitation is a significant constraint on the development of high-performance VLSI chips. The bottleneck can be greatly resolved if multiple computations are performed per I/O access. However, the repetitive use of a data item requires it to be stored inside the chip for a sufficient length of time. Thus the I/O cost is related not only to the available pin bandwidth, but also to the available memory internal to the chip. Tradeoffs between the memory size and the I/O bandwidth requirements must be carefully studied [11].

When a computation task is larger than what a single chip can handle, some multi-chip configurations will be adopted. In this case, chips communicate not only with the host but also among themselves. How to partition a task among multiple chips to preserve clean functional partitioning, and to achieve a balance between computation and inter-chip communication presents another challenge to a chip designer.

2. Lecture 2: Systolic Architectures for VLSI Design

The previous lecture indicated a number of architectural challenges concerning the design of special-purpose VLSI systems. Intended as a solution to these challenges, this lecture introduces the concept of *systolic architectures.* In the actual presentation illustrations and examples of systolic architectures will follow after the basic principle of a systolic architecture is described.

A *systolic system* consists of a set of interconnected *cells*, each capable of performing some simple operation. Because simple and regular communication and control structures have substantial advantages over complicated ones in design and implementation, cells in a systolic system are typically interconnected in a regular and modular way to form a *systolic array* or a *systolic tree.* Information in a systolic system flows (maybe in multiple directions and multiple speeds) between cells in a *pipelined* fashion resembling a factory assembly line, and communication with the outside world occurs only at the "boundary cells" of the system. For example, for a systolic array, only those cells on the array boundaries may be the I/O ports for the system.

Computational tasks can be conceptually classified into two families -- *compute-bound computations* and *I/O-bound computations.* If in a computation the total number of operations is larger than the total number of input elements, then the computation is compute-bound, otherwise it is I/O-bound. For example, the ordinary matrix-matrix multiplication algorithm represents a compute-bound task, since every entry in a matrix will be multiplied by *all* entries in some row or column of the other matrix. Adding two matrices, on the other hand, is I/O-bound, since the total number of adds is not larger than

the total number of entries in the two matrices. It should be clear that any attemp to speed up an I/O-bound computation must rely on an increase in memory bandwidth. Memory bandwidth can be increased by the use of either fast components (which could be expensive) or interleaved memories (which could create complicated memory management problems). Speeding up an compute-bound computation, however, may often be accomplished in a relatively simple and inexpensive manner, that is, by the systolic approach. The basic principle of a systolic architecture, a systolic array in particular, can be best illustrated using Figure 2-1. By replacing a single PE (processing element) with an array of PEs or cells in the terminology of these notes, a higher computation throughput can be achieved *without* increasing memory bandwidth. The function of the memory in the figure is analogous to that of the heart; it "pulses" data instead of blood through the array of cells. The crux of this architecture is to make sure that once a data item is brought out from the memory it can be used *effectively* at each cell it passes, as it marches along the array. We shall see that this is possible for a wide class of compute-bound computations; the fundamental reason is that in compute-bound problems multiple operations are performed per memory access, so they are potentially capable of supporting the principle of the systolic approcach, i.e., making multiple uses of each input data item.

Being able to use of each input data item a number of times (and thus achieving high computation throughputs with only a modest memory bandwidth) is just one of many advantages of the systolic approach. Other advantages include modular expandability, simple and regular data control flows, use of simple cells, elimination of broadcasting, and fast response time (possibly). These advantages have been demonstrated in various systolic designs listed in the following.

- Signal and Image Processing:
  - FIR, IIR Filtering [12, 13]
  - 1-D, 2-D Convolution and Correlation [17, 18, 19]
  - Discrete Fourier Transform [12, 13]

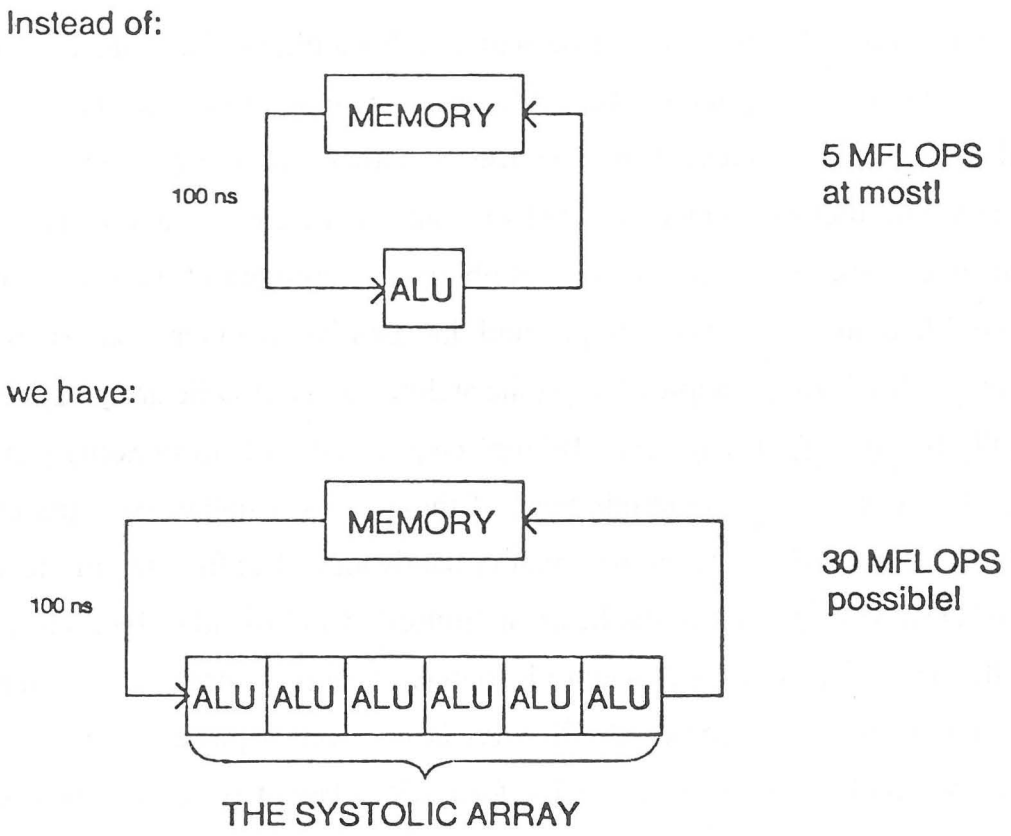

Figure 2-1: Principle of a systolic system.

- Interpolation
- 1-D, 2-D Median Filtering [6]
- Filer Designs [9]
- Geometric Warping [17]

• Matrix Arithmetic

- Matrix-Vector Multiplication [16]
- Matrix-Matrix Multiplication [16]
- LU-Decomposition (Matrix Inversion) [16, 9]
- QR-Decomposition (Eigenvalue, Least Square Computations) [3, 9]
- Solution of Triangular Linear Systems [16]

- Non-Numerical Applications
    - Sorting [20, 26]
    - Searching [2, 26]
    - Transitive Closure [10]
    - Priority Queue [20]
    - Minimum Spanning Trees [1]
    - Dynamic Programming [10]
    - Relational Database Operations [15]
- Language Recognizer
    - String Pattern Matching [7]
    - Recognizer for Regular Languages [8]

There are always many possible systolic designs for a given problem. For example, there are at least over half a dozen systolic designs for the convolution problem [14], and the same is true for matrix multiplication. It is also possible to have systolic arrays for which inside each cell there is a "cell memory" containing the complete set of weights. In fact, the systolic processor being built at ESL Inc. uses such a structure. Once one systolic design is obtained for a problem, it is likely that a set of other systolic designs can be derived similarly. The challenge is to understand precisely the strength and drawback of each design, so that an appropriate design can be chosen to fit a given environment. For example, for the convolution problem when there are more weights than available cells, it would be useful to know that a scheme where partial results stay would in general require less I/O than one where partial results move, since the latter scheme needs partial results to be output from and input to the systolic array multiple times.

3. Lecture 3: VLSI Complexity Theory

In this lecture we will discuss some theoretical issues in VLSI computation. Topics to be covered include models of VLSI computations [4, 21, 5, 28], area-time tradeoffs [4, 28], layout complexity [20] and I/O complexity [11, 26].

References

[1] Bentley, J.L.
A Parallel Algorithm for Constructing Minimum Spanning Trees.
Technical Report, Carnegie-Mellon University, Department of Computer Science, August, 1979.

[2] Bentley, J.L. and Kung, H.T.
A Tree Machine for Searching Problems.
In *Proceedings of 1979 International Conference on Parallel Processing*, pages 257-266. IEEE, August, 1979.
Also available as a CMU Computer Science Department technical report, August 1979.

[3] Bojanczk, A., Brent, R.P. and Kung, H.T.
Numerically Stable Solution of Dense Systems of Linear Equations Using Mesh-Connected Processors.
Technical Report, Carnegie-Mellon University, Department of Computer Science, May, 1981.

[4] Brent, R.P. and Kung, H.T.
The Chip Complexity of Binary Arithmetic.
In *Proceedings of the Twelfth Annual ACM Symposium on Theory of Computing*, pages 190-200. ACM, April, 1980.

[5] Chazelle, B.M. and Monier, L.M.
Towards More Realistic Models of Computation for VLSI.
In *Proceedings of the Second Caltech VLSI Conference.* Caltech, January, 1981.

[6] Fisher, A.L.
Systolic Algorithms for Running Order Statistics in Signal and Image processing.
Technical Report, Carnegie-Mellon University, Department of Computer Science, March, 1981.

[7] Foster, M. J. and Kung, H.T.
The Design of Special-Purpose VLSI Chips.
Computer Magazine 13(1):26-40, January, 1980.
A preliminary version of the paper, entitled "Design of Special-Purpose VLSI Chips: Example and Opinions", appears in *Proceedings of the 7th International Symposium on Computer Architecture*, pp. 300-307, La Baule, France, May 1980.

[8] Foster, M.J. and Kung, H.T.
Recognize Regular Languages With Programmable Building-Blocks.
In *VLSI 81*, pages 75-84. Academic Press, August, 1981.

[9] Gentleman, W.M. and Kung, H.T.
Matrix Triangularization by Systolic Arrays.
In *Prodeedings of SPIE Symposium, Vol. 298, Real-Time Signal Processing.* 1981.

[10] Guibas, L.J., Kung, H.T. and Thompson, C.D.
Direct VLSI Implementation of Combinatorial Algorithms.
In *Proceedings of Conference on Very Large Scale Integration: Architecture, Design, Fabrication*, pages 509-525. California Institute of Technology, January, 1979.

[11] Hong, J.-W. and Kung, H.T.
I/O Complexity: The Red-Blue Pebble Game.
In *Proceedings of the Thirteenth Annual ACM Symposium on Theory of Computing*, pages 326-333. ACM SIGACT, May, 1981.

[12] Kung, H.T.
Let's Design Algorithms for VLSI Systems.
In *Proceedings of Conference on Very Large Scale Integration: Architecture, Design, Fabrication*, pages 65-90. California Institute of Technology, January, 1979.
Also available as a CMU Computer Science Department technical report, September 1979.

[13] Kung, H.T.
Special-Purpose Devices for Signal and Image Processing: An Opportunity in VLSI.
In *Proceedings of the SPIE, Vol. 241, Real-Time Signal Processing III*, pages 76-84. The Society of Photo-Optical Instrumentation Engineers, July, 1980.

[14] Kung, H.T.
Why Systolic Architectures.
To appear in *Computer Magazine* 1981.

[15] Kung, H.T. and Lehman, P.L.
Systolic (VLSI) Arrays for Relational Database Operations.
In *Proceedings of ACM-SIGMOD 1980 International Conference on Management of Data*, pages 105-116. ACM, May, 1980.
Also available as a CMU Computer Science Department technical report, August 1979.

[16] Kung, H.T. and Leiserson, C.E.
Systolic Arrays (for VLSI).
In Duff, I. S. and Stewart, G. W. (editors), *Sparse Matrix Proceedings 1978*, pages 256-282. Society for Industrial and Applied Mathematics, 1979.
A slightly different version appears in *Introduction to VLSI Systems* by C. A. Mead and L. A. Conway, Addison-Wesley, 1980, Section 8.3.

[17] Kung, H.T. and Picard, R.L.
Hardware Pipelines for Multi-Dimensional Convolution and Resampling.
Workshop on Computer Architecture for Pattern Analysis and Image Database Management, November, 1981.

[18] Kung, H.T. and Song, S.W.
A Systolic 2-D Convolution Chip.
Technical Report CMU-CS-81-110, Carnegie-Mellon University, Department of Computer Science, March, 1981.
to appear in *Non-Conventional Computers and Image Processing: Algorithms and Programs*, Uhr, L. (editor), Academic Press, 1981.

[19] Kung, H.T., Ruane, L. and Yen, D.
Two-Level Pipelined Systolic Arrays for Convolution.
In *VLSI Systems and Computations.* Computer Science Press, 1981.

[20] Leiserson, C.E.
Systolic Priority Queues.
In *Proceedings of Conference on Very Large Scale Integration: Architecture, Design, Fabrication*, pages 199-214. California Institute of Technology, January, 1979.
Also available as a CMU Computer Science Department technical report, April 1979.

[21] Leiserson, C.E.
Area-Efficient VLSI Computation.
PhD thesis, Carnegie-Mellon University, 1981.

[22] Mead, C.A. and Conway, L.A.
Introduction to VLSI Systems.
Addison-Wesley, Reading, Massachusetts, 1980.

[23] Nishitani, T., Y. Kawakami, R. Maruta and A. Sawai.
LSI Signal Processing Development for Communications Equipment.
In *Proceedings of ICASSP80*, pages 386-389. IEEE Acoustics, Speech and Signal Processing Society, April, 1980.

[24] Robert N. Noyce.
Hardware Prospects and Limitations.
In Dertouzos, M.L. and Moses, J. (editor), *The Computer Age: A Twenty-Year View*, pages 321-337. IEEE, 1979.

[25] Schirm IV, L.
Multiplier-Accumulator Application Notes.
Technical Report, *TRW* LSI PRODUCTS, January, 1980.

[26] Song, S.W.
On a High-Performance VLSI Solution to Database Problems.
PhD thesis, Carnegie-Mellon University, Department of Computer Science, 1981.

[27] Swartzlander, E.E., Jr., Gilbert, B.K., and Reed, I.S.
Inner Product Computers.
IEEE Transactions on Computers C-27(1):21-31, 1978.

[28] Thompson, C.D.
A Complexity Theory for VLSI.
PhD thesis, Carnegie-Mellon University, Department of Computer Science, 1980.

Discussion

P.J. Brown Is it as easy and clean to replicate cells as you suggest?

H.T. Kung Yes it is - it even works with off-the-shelf chips. You can use the components to build boards, and then use as many boards as necessary to build the system. The only broadcast signal is the clock, which doesn't need to be broadcast in many applications where there are no global communications.

P.J. Brown This is only true if the circuits are self-timed. Doesn't this make them more complex?

H.T. Kung Not if you only have one cell.

D.J. Rees Your concept of a machine is one that is general purpose, with special-purpose functions. Wouldn't it be better to refine special purpose machines?

H.T. Kung A complicated question to answer. It is often possible to design one basic machine that will perform several tasks. If it can be designed cheaply you can afford only limited use.

J. Katzenelson If you process a lot of data won't you need a different bus structure?

H.T. Kung Yes, we are currently looking at this problem.

J. Katzenelson If one day you want to increase the size of the matrix you are using the bus will become a problem.

H.T. Kung With a two dimensional design you will need increased bandwidth. You can however design a one dimensional system that can be extended indefinitely.