THE IMPACT OF RENT'S RULE ON MASSIVE PARALLELISM

P. J. Koopman D. P. Siewiorek
ECE Department CS Department
Carnegie Mellon University
Pittsburgh, PA

ABSTRACT: Rent's Rule is an empirical relationship stating that the number of pins on a chip increases as the number of gates on the chip increases. In massively parallel systems, every extra pin is multiplied by the number of processors. This causes a rapid increase in system complexity, cost, and failure rate. The key to more efficient massively parallel systems is finding a way around Rent's Rule. By studying the effects of re-implementing a system of fixed complexity using different integration levels, we have found that Rent's rule does not apply to systems which place program memory on the same chip as the processor. This suggests that a focus for massively parallel systems might be to use processing elements simple enough to completely fit on a single chip, rather than faster but more complex processors that use external memory devices.

Keywords: Rent's Rule, integration level, system complexity.

INTRODUCTION

Rent's Rule (Ref. 1) is an empirical relationship between the number of gates and the number of I/O pins a single chip. The relationship is given by:

$$IO = AS * G^{R}$$

In this equation, **IO** is the number of input/output pins on the chip. AS is the complexity of a single logic gate on the chip as measured by the number of inputs for the gate. **G** is the number of logic gates on the chip. **R** is the Rent Exponent, which is a circuit-dependent "magic" number between 0 and 1, which is often near 0.5.

The trend in VLSI processor design has been: given the availability of more silicon real estate, put more sophisticated functions or wider data paths into a single chip. Thus, memory chips have progressed from 256K bits to 1M bits to 4M bits. Also, microprocessors have evolved from 8 bits wide to 16 bits to 32 bits. These chips all obey the Rent's Rule prediction of a logarithmic increase in the number of pins as the number of gates on the chip increases. This increase in the number of pins has important implications for the builders of massively parallel systems.

THE COST OF TOO MANY PINS

Since the innovation of standardized integrated circuits we have progressed from the introduction of the 14-pin dual inline package (DIP) to common use of pin grid array packages (PGAs) with hundreds of pins. The addition of extra pins to a chip has some obvious as well as hidden costs.

The most obvious cost is the manufacturing cost of the chip package itself. Small DIPs are very inexpensive to manufacture since they use stamped metal pins. As chips require more pins, DIPs become impractical, and packages such as leadless chip carriers (LCCs) are used. Each contact on an LCC costs more than a DIP pin, because it must be more precisely manufactured and placed around four sides of the package. At the high end of the spectrum, PGAs use precision-machined round pins that are precisely placed in a matrix on the back of the chip. It is not unusual for each pin of a PGA to be several times more expensive than an entire DIP package. Thus, there is a very steeply increasing cost curve for the entire chip as the number of pins is increased.

The direct cost of the chip package only begins to describe the costs of adding pins, however. At the on-chip level, every pin must have an on-chip pad. This pad consumes valuable silicon real estate. But, more importantly, each pad consumes power. As geometries become smaller, the amount of power used by a chip to drive its output pins can dwarf power consumption for on-chip logic. The problem is especially severe with CMOS technology, which is coming into favor for high density circuits.

At the system level, the footprint of the package on the printed circuit board increases as the number of pins on the package increases, costing valuable printed circuit board real estate. Increased printed circuit board sizes result in bigger cabinets and, ultimately, more boards with expensive and slow interconnect structures. Dense pin arrangements such as those found on PGAs further aggravate the problem by requiring expensive multi-layer boards.

Many indirect costs are associated with chips that have large pin counts. These costs include the use of very expensive automated chip testers when they are manufactured. Also, every extra pin in a finished computer reduces the over-all system reliability, since interconnect failure is a frequent system failure mode. (Ref. 2)

In a parallel computer, these pin costs are multiplied by the amount of parallelism in the design. Since the premise behind a massively parallel system is that more processors are better, all massively parallel designs will ultimately be limited in processing power by the number of processors that can be afforded within a given space/power/cost budget. The number of pins in each processing element within the system can therefore directly affect the ultimate computational power of a massively parallel processor.

BREAKING THE RULE

One should not infer from the previous discussion that the use of VLSI chips with large pinouts is bad. These chips, while expensive, are less expensive than the total system cost of using a large number of less complex chips instead. The question is: can we do better?

Rent's Rule predicts that increased VLSI chip complexity will lead to an inexorable increase in pin count. A key to making massively parallel systems faster and more cost-effective is to find a way to break out of Rent's empirical relationship. One way to accomplish this goal is to find an implicit assumption in the relationship that can be altered.

There is an historic relationship between chip complexity and overall system complexity. As chips have become more dense, computer systems have not only become more highly integrated, but have also become more complex. Adding more complexity to a system makes sense in a uniprocessor environment, where the added complexity squeezes the most possible performance from the machine. Rent's Rule applies to computer systems as they have been built over the years, so it accounts for this implicit assumption. But what if we violate that assumption, and hold system complexity as a constant?

The answer to this question may be found by conducting an experiment that holds system complexity as a constant for varying integration levels. In order to do this, we built a hierarchical description of a 16-bit microprocessor system (Ref. 3) starting at the gate level. All circuit functions were ultimately reduced to combinations of 2-input NAND gates for simplicity. Then, we did a redesign of the system using six different integration levels ranging from SSI (all 2-input NAND gates) to high density VLSI (entire system on a single chip). Each integration level was chosen to correspond to a reasonable method for partitioning the system components. Figure 1 shows a graph of chip complexity versus pin count for the various implementations, as well as the curve for a Rent exponent of 0.38. There are more than six data points in this graph, since most implementations had several chips in the design. RAM chips are not shown as they obey a Rent curve with a different slope that clutters the diagram. Power supply pins are not accounted for since they vary with implementation technology.

Integration Levels 1 and 2, which correspond to SSI and MSI components, obey a classic Rent's Rule curve with an exponent value of approximately 0.38. Integration Level 4, which corresponds to a 3-chip system, also falls neatly on this curve. Integration Level 5, which corresponds to a standard micro-processor 2-chip system (processor chip and memory chip) is somewhat off the curve, but is still a reasonable fit. Integration Level 3 turned out to be an awkward level of integration, which forced a very poor partitioning of the system, resulting in a very high pin count for one of the chips.

The really interesting point on the graph is Integration Level 6. This design is nowhere near the curve! Integration Level 6 corresponds to a single-chip system, which incorporates program memory and the processing logic on the same chip. This implementation appears to break Rent's Rule.

INTERPRETING THE RESULTS

Figure 2 shows a curve that helps us interpret the results of the experiment. If we ignore Integration Level 3 as a "bad data point", then what is really happening is that the designs obey Rent's rule quite well through Integration Level 4. Then, as we reach very high levels of system integration, the number of pins on the chips begins to decrease. If the entire system is on a single chip, only a few pins for system I/O are needed. While the microprocessor seems to be near the break in the curve, the break is not really noticeable until the system-on-a-chip approach is taken.

The results, once one thinks about the situation, are rather straightforward. A system-on-a-chip needs off-chip interconnection only for I/O, so it needs very few pins. Why hasn't this concept been exploited then? The reason is that it is of limited use in the uniprocessor world. Most high-performance uniprocessors are too complex to allow enough room for on-chip memory.

The situation in a massively parallel processor environment is quite different than in a uniprocessor environment. Since massive parallelism is cost effective only in applications which can achieve roughly linear speed improvements as processors are added, N processors that perform at 1/Nth the speed of a given uniprocessor are roughly equal in processing power to that same uniprocessor.

The approach that is supported by these findings is one of building relatively simple processor/memory systems that can fit on a single chip. Since these chips will be much less expensive to manufacture and use in a system, more processors can be included in a system.

There are several methods of implementing this strategy. One method is to simplify a given processor implementation as much as possible, probably sacrificing speed-enhancing hardware features for overall system size. With current technology (1 to 2 micron CMOS), this approach can lead to simple 16-bit processing elements with small program memories. Of course, appropriate software techniques to keep code size small are vitally important. This approach is probably the most attractive for MIMD machines.

Another possible method is to reduce the word-size of each processing element. The ultimate extension of this philosophy is bit-serial machines which can, in fact, have multiple processing/memory elements per chip. This approach is obviously well-suited to SIMD machines.

CONCLUSIONS

In the near term, the challenge to achieving the maximum level of processing element integration is to find design styles and programming methodologies that can fit enough functionality onto available chip real estate to go beyond the Rent's Rule breaking point. Current architectures which may be able to do this include: bit-serial processors, which can pack several processors with memory onto a single chip; 8-bit microcontrollers, which are probably not powerful enough to be of interest in their currently available form; and stack-oriented processors, with their small program memory size requirements. In the future, chip sizes may increase enough to allow RISC processors to have a full-sized on-chip cache and slow serial interfaces to their program memories. CISC processors may eventually reach this point as well, but only if they are frozen at a particular complexity level.

Some parallel processor architectures, especially SIMD architectures, are clearly already embracing the philosophy of simple computational elements that can fit on a single chip. What we have explored are some of the theoretical underpinnings of this approach, and why it makes sense for massively parallel architectures.

REFERENCES

1. Landman, B.S. and Russo, R.L., "On a Pin Versus Block Relationship For Partitions of Logic Graphs," *IEEE Transactions on Computers,* December 1971, **C20**(12), pp. 1469-1479

2. Siewiorek, D.P. and Swarz, R.S., *The Theory and Practice of Reliable System Design,* Digital Press, Bedford MA, 1982

3. Koopman, P.J., *CPU/16 Technical Reference Manual,* WISC Technologies, La Honda CA, 1986.

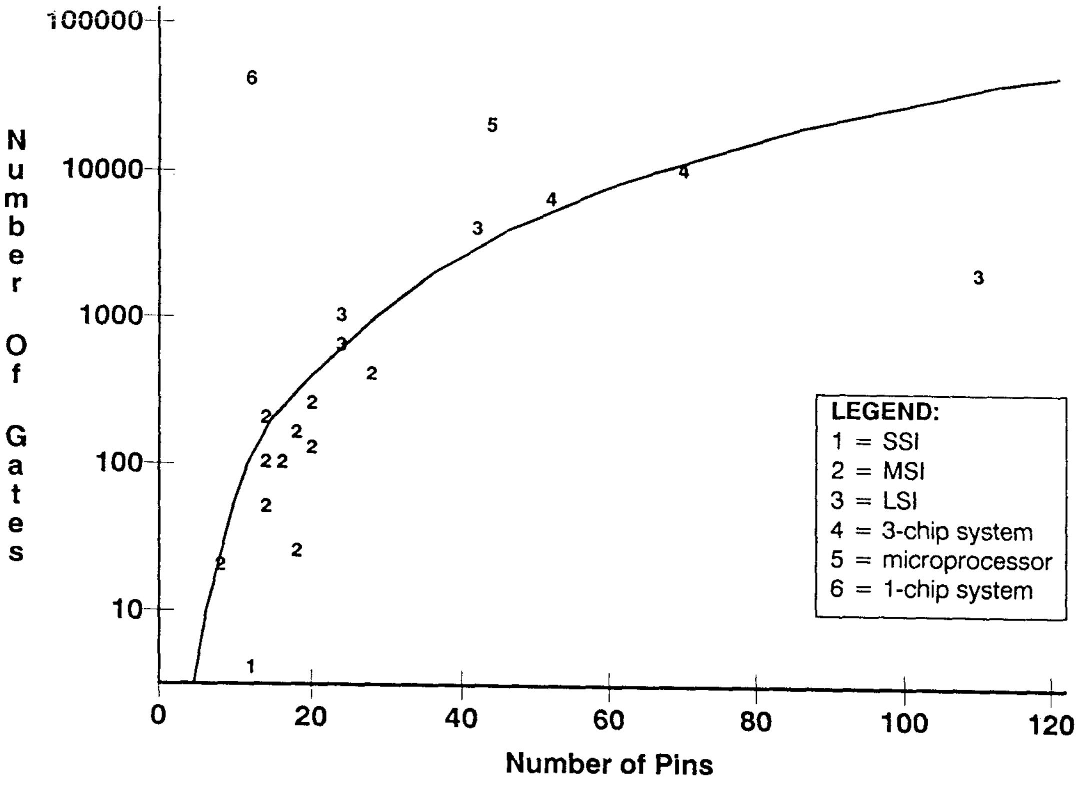


Figure 1. Experimental data shown with a Rent curve (R=0.38)

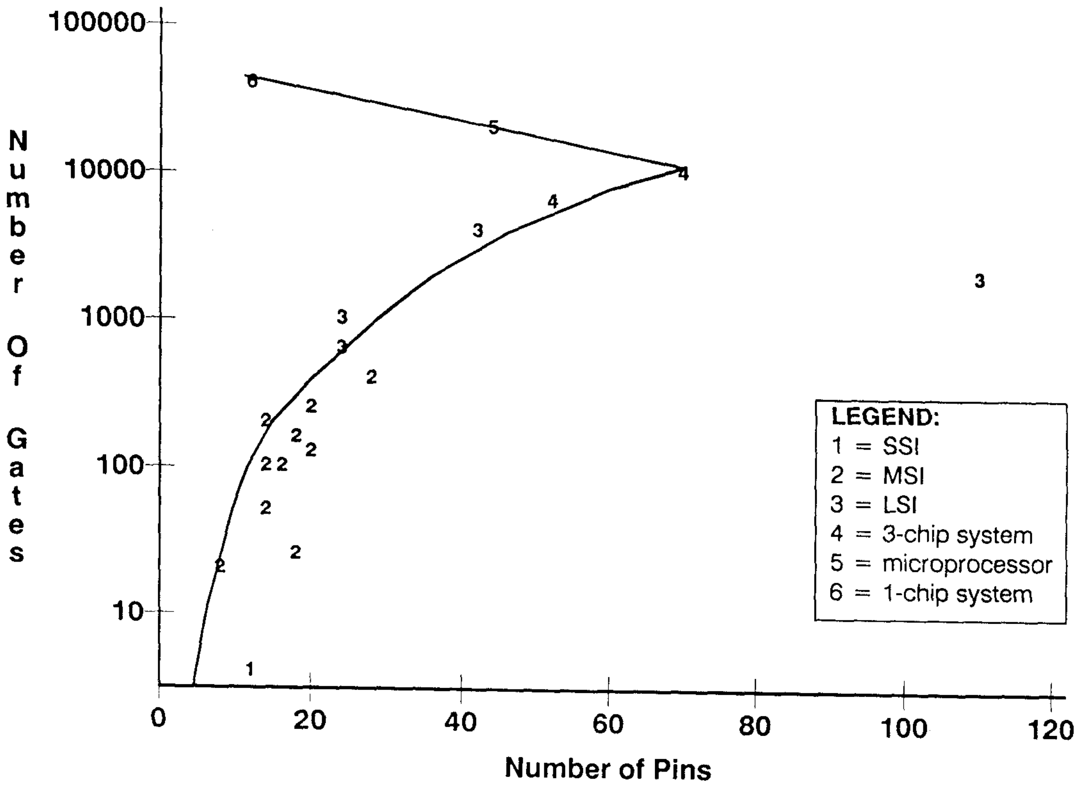


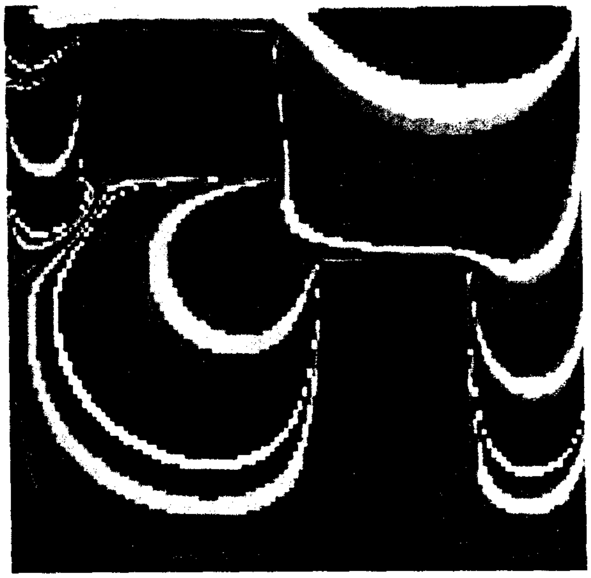
Figure 2. A different curve that better fits the data

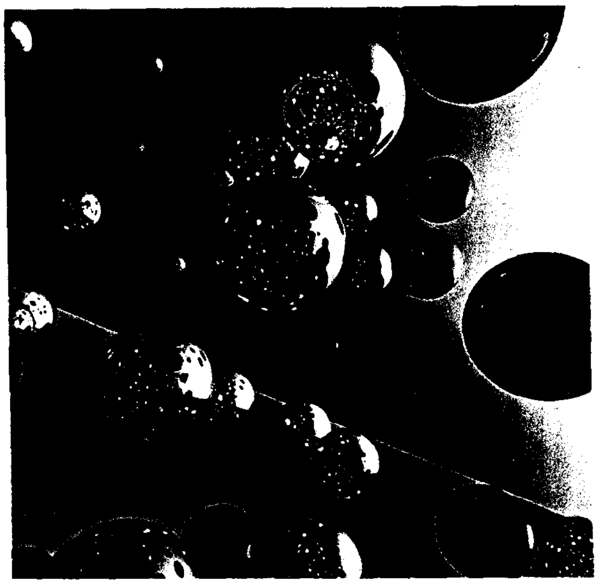
Proceedings

FRONTIERS '88

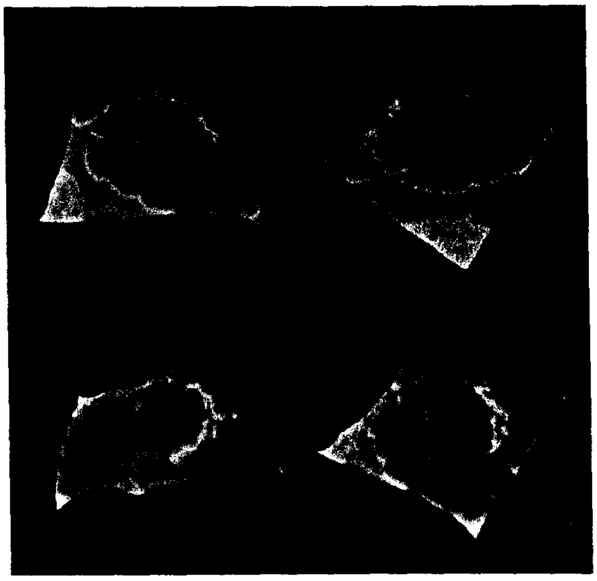
The 2nd Symposium on the

Frontiers of Massively Parallel Computation





October 10–12, 1988
George Mason University
Fairfax, Virginia



IEEE Computer Society Order Number 892
Library of Congress Number 88–82088
IEEE Catalog Number 88CH2649-2
ISBN 0–8186–0892–7
SAN 264–620X

IEEE COMPUTER SOCIETY

THE INSTITUTE OF ELECTRICAL AND ELECTRONICS ENGINEERS, INC.

IEEE

COMPUTER SOCIETY PRESS

IEEE–NATIONAL CAPITAL AREA COUNCIL

George Mason University

NASA
National Aeronautics and Space Administration