A

Seminar report

On

Asynchronous Chips

Submitted in partial fulfillment of the requirement for the award of degree
Of CSE

SUBMITTED TO:

www.studymafia.org

SUBMITTED BY:

www.studymafia.org

Preface

I have made this report file on the topic **Asynchronous Chips;** I have tried my best to elucidate all the relevant detail to the topic to be included in the report. While in the beginning I have tried to give a general view about this topic.

My efforts and wholehearted co-corporation of each and everyone has ended on a successful note. I express my sincere gratitude to …………..who assisting me throughout the preparation of this topic. I thank him for providing me the reinforcement, confidence and most importantly the track for the topic whenever I needed it.

Acknowledgement

I would like to thank respected Mr…….. and Mr. ……..for giving me such a wonderful opportunity to expand my knowledge for my own branch and giving me guidelines to present a seminar report. It helped me a lot to realize of what we study for.

Secondly, I would like to thank my parents who patiently helped me as i went through my work and helped to modify and eliminate some of the irrelevant or un-necessary stuffs.

Thirdly, I would like to thank my friends who helped me to make my work more organized and well-stacked till the end.

Next, I would thank Microsoft for developing such a wonderful tool like MS Word. It helped my work a lot to remain error-free.

Last but clearly not the least, I would thank The Almighty for giving me strength to complete my report on time.

CONTENTS

INTRODUCTION

Computer chips of today are synchronous. They contain a main clock, which controls the timing of the entire chips. There are problems, however, involved with these clocked designs that are common today.

One problem is speed. A chip can only work as fast as its slowest component. Therefore, if one part of the chip is especially slow, the other parts of the chip are forced to sit idle. This wasted computed time is obviously detrimental to the speed of the chip.

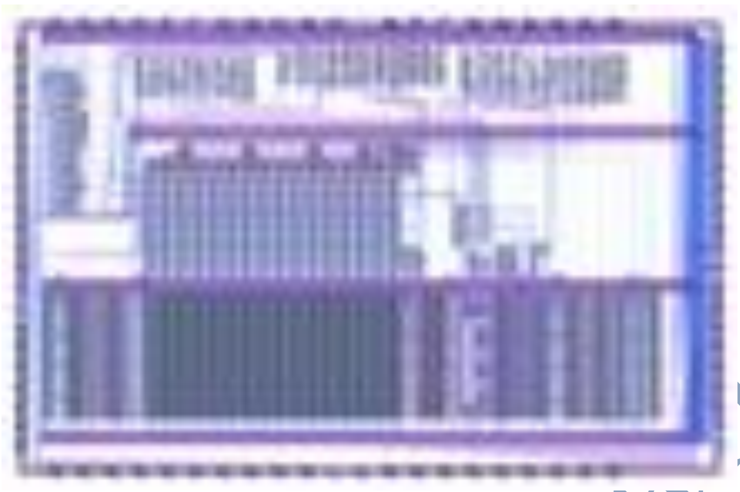
New problems with speeding up a clocked chip are just around the corner. Clock frequencies are getting so fast that signals can barely cross the chip in one clock cycle. When we get to the point where the clock cannot drive the entire chip, we'll be forced to come up with a solution. One possible solution is a second clock, but this will incur overhead and power consumption, so this is a poor solution. It is also important to note that doubling the frequency of the clock does not double the chip speed, therefore blindly trying to increase chip speed by increasing frequency without considering other options is foolish.

The other major problem with c clocked design is power consumption. The clock consumes more power that any other component of the chip. The most disturbing thing about this is that the clock serves no direct computational use. A clock does not perform operations on information; it simply orchestrates the computational parts of the computer.

New problems with power consumption are arising. As the number of transistors on a chi increases, so does the power used by the clock. Therefore, as we design more complicated chips, power consumption becomes an even more crucial topic. Mobile electronics are the target for many chips.

These chips need to be even more conservative with power consumption in order to have a reasonable battery lifetime.

The natural solution to the above problems, as you may have guessed, is to eliminate the source of these headaches: the clock.



The Caltech Asynchronous Microprocessor is the world's first asynchronous microprocessor (1989).

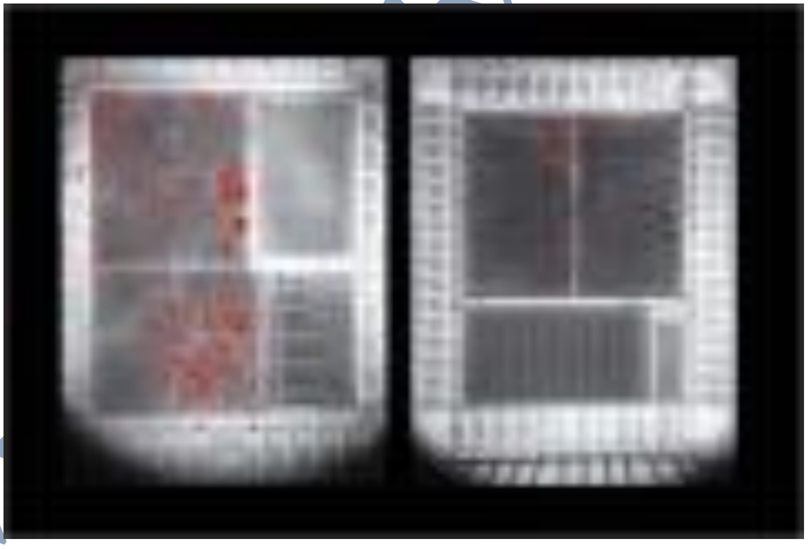
DISCUSSION

Asynchronous, or clock less, design has advantages over the synchronous design.

The first of these advantages is speed. Chips can run at the average speed of all its components instead of the speed of its slowest component, as was the case with a clocked design. Also the need to have a clock running at a speed such that the signal can reach all parts of the chip is eliminated. Therefore, the speed of an asynchronous design is not limited by the size of the chip.

An example of how much an asynchronous design can improve speed is the asynchronous Pentium designed by Intel in 1997 that runs three times as fast as the synchronous equivalent. This speedup is certainly significant and proves the usefulness of a clock less design.

The other advantage of a clock less design is power consumption.



Special light emission measurements of a synchronous chip (left) and an asynchronous chip (right) with the same digital functionality under the same operational conditions indicate hoe much power the chips dissipate.

The above graphic illustrates the power saving characteristic of a clock less design. The reason for this is that asynchronous chips use power only during computations, while a clocked chip always consumes power because the chip is always running. Remember that the clock is the component which consumes the most power. Therefore, eliminating the clock eliminates the largest component of power consumption.

One example of improved power consumption is the same Intel Pentium asynchronous chip. This design, which ran up to three times as fast as the clocked version, runs on half the power of the clocked version. This is incredible support for a clock less design. A second example of improved power consumption is a Philips prototype chip that runs on one-third of the power of its clocked counterpart.

Clock less design is inevitable in the future of chip design because of the two major advantages of speed and power consumption, but where will we first see these designs in use?

The first place we'll see, and have already seen, clock less designs are in the lab. Many prototypes will be necessary to create reliable designs. Manufacturing techniques must also be improved so the chips can be mass-produced.

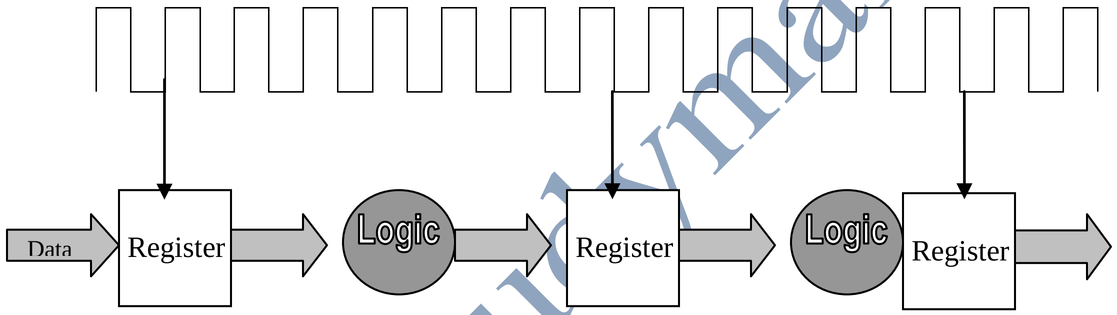
The second place we'll see these chips are in mobile electronics. This is an ideal place to implement a clock less chip because of the minimal power consumption. Also, low levels of electromagnetic noise creates less interference, less interference is critical in designs with many components packed very tightly, as is the case with mobile electronics.

The third place is in personal computers (PCs). Clock less designs will occur here last because of the competitive PC market.

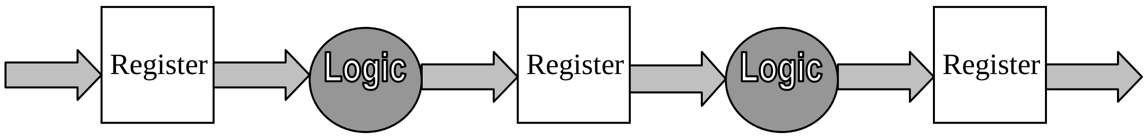
It is essential in that market to create an efficient design that is reasonably priced. A manufacturing cost increase of a couple of cents per chip can cause an entire line of computers to fail because of the large cost increase passed onto the customer. Therefore, the manufacturing process must be improved to create a reasonably priced chip.

A final place that asynchronous design may be used is encryption devices. The reason for this is there are no regularly timed signals for hackers to look for. This becomes even more critical as computers all over the world become more closely connected and are sharing confidential material. Security in the United States has increased greatly in recent times; therefore, a clock less design will be welcomed because of its encryption abilities.

In summary, clock less designs have limitations, specifically a limited speed and high power consumption. Fortunately, these limitations can be solved with a clock less design. These asynchronous designs will be seen in many areas of technology, but it will take time before these chips can be perfected.



CONVENTIONAL CHIPS operate under the control of a central clock, which samples data in the registers at precisely timed intervals.



CLOCK LESS CHIPS dispense with the timepiece. In one scheme, data moves instead under the control of local "handshake" signals that indicate when work has been completed and is ready for the next logic operation.

ASYNCHRONOUS LOGIC

Data-driven circuits design technique where, instead of the components sharing a common clock and exchanging data on clock edges, data is passed on as soon as it is available. This removes the need to distribute a common clock signal throughout the circuit with acceptable clock skew. It also helps to reduce power dissipation in CMOS circuits because gates only switch when they are doing useful work rather than on every clock edge.

There are many kinds of asynchronous logic. Data signals may use either "dual rail encoding" or "data building". Each dual rail encoded Boolean is implemented as two wires. This allows the value and the timing information to be communicated for each data bit. Bundled data has one wire for each data bit and another for timing. Level sensitive circuits typically represent a logic one by a high voltage and a logic zero by a low voltage whereas transition signaling uses a change in the signal level to convey information. A speed independent design is tolerant to variations in gate speeds but not to propagation delays in wires; a delay insensitive circuit is tolerant to variations in wire delays as well.

The purest form of circuit is delay-insensitive and uses dual-rail encoding with transition signaling. A transition on one wire indicates the arrival of a zero, a transition on the other the arrival of a one. The levels on the wires are of no significance. Such an approach enables the design of fully delay-insensitive circuits and automatic layout, as the delays introduced by the layout compiler can't affect the functionality (only the performance). Level insensitive designs can use simpler, stateless logic gates but require a "return to zero" phase in each transition.

COMPUTERS WITHOUT CLOCKS

Asynchronous chips improve computer performance by letting each circuit run as fast as it can.

How fast is your personal computer?

When people ask this question, they are typically referring to the frequency of a minuscule clock inside the computer, a crystal oscillator that sets the basic rhythm used throughout the machine. In a computer with a speed of one Gigahertz, for example, the crystal “ticks” a billion times a second. Every action of he computer takes place in tiny step; complex calculations may take many steps. All operations, however, must begin and end according to the clock’s timing signals.

Since most modern computers use a single rhythm, we call them synchronous. Inside the computer’s microprocessor chip, a clock distribution system delivers the timing signals from the crystal oscillator to the various circuits, just as sound in air delivers the beat of a drum to soldiers to set their marching space. Because all parts of the chip share the same rhythm, the output of any circuit from one step can serve as the input to any other circuit for the next step. The synchronization provided by the clock helps chip designers plan sequences of actions for the computer.

The use of a central clock also creates problems. As speeds have increased, distributing the timing signals has become more and more difficult. Present day transistors can process data so quickly that they can accomplish several steps in the time that it takes a wire to carry a signal from one side of the chip to the other.

Keeping the rhythm identical in all parts of a large chip requires careful design and a great deal of electrical power.

Each part of an asynchronous system may extend or shorten the timing of its steps when necessary, much as a hiker takes long or short steps when walking across rough terrain. Some of the pioneers of the computer age, such as mathematician Allen M Turing, tried using asynchronous designs to build machines in the early 1950's. Engineers soon abandoned this approach in favour of synchronous computers because common timing made the design process so much easier.

Now asynchronous computing is experiencing a renaissance. Researchers at the University of Manchester in England, The University of Tokyo and The California Institute of Technology had demonstrated asynchronous microprocessors. Some asynchronous chips are already in commercial mass production. In the late 1990's Sharp, the Japanese electronics company used asynchronous design to build a data driven media processor – a chip for editing graphics, video and audio – and Philips Electronics produced an asynchronous microcontroller for two of its pagers. Asynchronous parts of otherwise synchronous systems are also beginning to appear; the UltraSPARC IIIi processor recently introduced by SUN includes some asynchronous circuits developed by our group. We believe that asynchronous systems will become ever more popular as researchers learn how to exploit their benefits and develop methods for simplifying their design. Asynchronous chipmakers have achieved a good measure of technical success, but commercial success is still to come. We remain a long way from fulfilling the full promise of asynchrony.

BEAT THE CLOCK

What are the potential benefits of asynchronous systems?

First, asynchrony may speed up computers. In a synchronous chip, the clock's rhythm must be slow enough to accommodate the slowest action in the chip's circuits. If it takes a billionth of a second for one circuit to complete its operation, the chip cannot run faster than one gigahertz. Even though many other circuits on that chip may be able to complete their operations in less time, these circuits must wait until the clock ticks again before proceeding to the next logical step. In contrast each part of an asynchronous system takes as much or as little time for each action as it needs.

Complex operations can take more time than average, and simple ones can take les. Actions can start as soon as the prerequisite actions are done, without waiting for the next tick of the clock. Thus the systems speed depends on the average action time rather than the slowest action time.

Coordinating as actions, however, also takes time and chip area. If the efforts required for local coordination are small, an asynchronous system may, on average, be faster than a clocked system. Asynchrony offers the most help to irregular chip designs in which slow actions occur infrequently.

Asynchronous design may also reduce a chip's power consumption. In the current generation of large, fast synchronous chips, the circuits that deliver the timing signals take up a good chunk of the chip's area. In addition, as much as 30% of the electrical power used by the chip, must be devoted to the clock and its distribution system. Moreover, because the clock is always running, it generates heat whether or not the chip is doing anything useful.

In asynchronous systems, idle parts of the chip consume negligible power. This feature is particularly valuable for battery-powered equipment, but it can also cut the cost of larger systems by reducing the need for cooling fans and air-conditioning to prevent them from overheating. The amount of power saved depends on the machine's pattern of activity. Systems with parts that act only occasionally benefit more than systems that act continuously. Most computers have components, such as the floating-point arithmetic unit, that often remain idle for long periods.

Furthermore, as systems produce less ratio interference than synchronous machines do. Because of a clocked system uses a fixed rhythm, it broadcasts a strong radio signal at its operating frequency and at the harmonics of that frequency. Such signals can interfere with cellular phones, televisions and aircraft navigation systems that operates t the same frequencies. Asynchronous systems lack a fixed rhythm, so they spread their radiated energy broadly across the radio spectrum, emitting less at any one frequency.

Overview / clock less systems

- ❖ Most modern computers are synchronous: all their operations are coordinated by the timing signals of tiny crystal oscillators within the machines. Now researchers are designing asynchronous systems that can process data without the need for a governing clock.

- ❖ Asynchronous systems rely on local coordination circuits to ensure an orderly flow of data. The two most important coordination circuits are called the Rendezvous and the Arbiter.

- ❖ The potential benefits of asynchronous systems include faster speeds, lower power consumption and less radio interference. As integrated circuit become more complex, chip designers will need to learn asynchronous techniques.

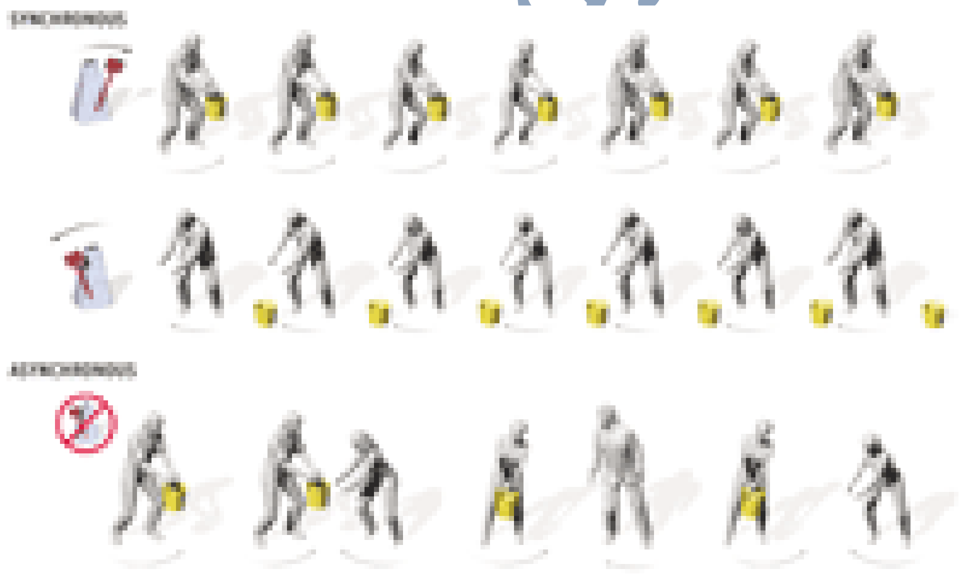
Yet another benefit of asynchronous design is that it can be used to build bridges between clocked computers running at different speeds. Many computing clusters, for instance, link fast PCs with slower machines. These clusters can tackle complex problems by dividing the computational tasks among the PCs. Such a system is inherently asynchronous: different parts march to different beats. Moving data controlled by one clock to the control of another clock requires asynchronous bridges, because data may be “out of sync” with the receiving clock.

Finally, although asynchronous design can be challenging, it can also be wonderfully flexible. Because of the circuits of an asynchronous system need not share a common rhythm, designers have more freedom in choosing the systems’ parts and determining how they interact. Moreover, replacing any part with a faster version will improve the speed of the entire system. In contrast, increasing the speed of a clocked system usually requires upgrading every part.

LOCAL OPERATION

To describe how asynchronous systems work, we often use the metaphor of the bucket brigade. A clocked system is like a bucket brigade in which each person must pass and receive buckets according to the tick tock rhythm of the clock. When the clock ticks, each person pushes a bucket forward to the next person down the line. When the clock tocks, each person grasps the bucket pushed forward by the preceding person. The rhythm of this brigade cannot go faster than the time it takes the slowest person to move the heaviest bucket. Even if most of the buckets are light, everyone in the line must wait for the clock to tick before passing the next bucket.

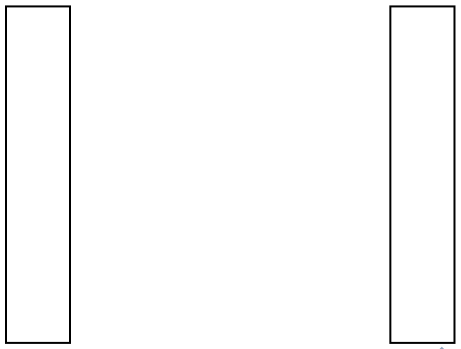
Local cooperation rather than the common clock governs an asynchronous bucket brigade. Each person who holds a bucket can pass it to the next person down the line as soon as the next person's hands are free. Before each action, one person may have to wait until the other is ready. When most of the buckets are light, however, they can move down the line very quickly. Moreover, when there's no water to move, everyone can rest between buckets. A slow person will still hinder the performance of the entire brigade, but replacing the slowpoke will return the system to its best speed.



Bucket brigade

Bucket brigades in computers are called pipelines. A common pipeline executes the computer’s instructions. Such a pipeline has half a dozen or so stages, each of which acts as a person in a bucket brigade.

For example, a processor executing the instruction “ADD A B Chip” must fetch the instruction from memory, decode the instruction, get the numbers from addresses A and B in memory, do the addition and store the sum in memory address C.



Pipeline diagram

Here a “bundled data” self-timing scheme is used, where conventional data processing logic is used along with a separate request (Req) line to indicate data validity. Requests may be delayed by at least the logic delay to insure that they still indicate data validity at the receiving register. An acknowledge signal (ack) provides flow control, so the receiving register can tell the transmitting register when to begin sending the next data.

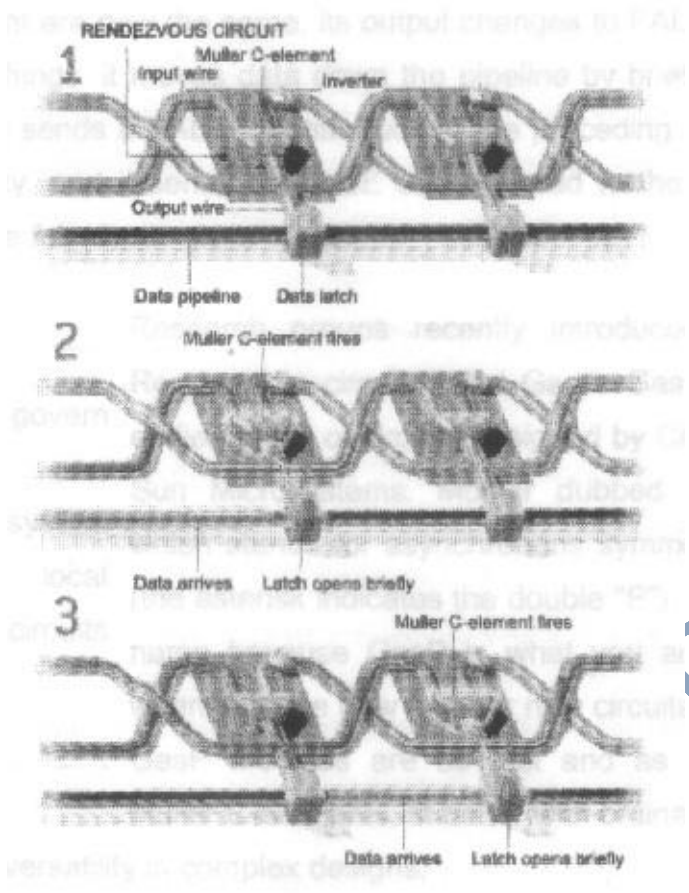
A clocked pipeline executes these actions in a rhythm independent of the operations performed or the size of the numbers. In an asynchronous pipeline, though, the duration of each action may depend on the operation performed the size of the numbers and the location of the data in memory (just as in bucket brigade the amount of water in a bucket may determine how long it takes to pass it on).

Without a clock to govern its actions, an asynchronous system must rely on local coordination circuits instead. These circuits exchange completion signals to ensure that the actions at each stage begin only when the circuits have the data they need. The two most important coordination circuits are called the Rendezvous and the Arbiter circuits.

A Rendezvous element indicates when the last of two or more signals has arrived at a particular stage. Asynchronous systems use these elements to wait until all the concurrent actions finish before starting the next action. For instance, an arithmetic division circuit must have both the dividend (say, 16) and the divisor (say, 2) before it can divide one by the other (to reach the answer 8).

One form of Rendezvous circuit is called the Muller C-element, named after David Muller, now retired from a professorship at the University of Illinois. A Muller C-element is a logic circuit with two inputs and on output. When both inputs of a Muller C-element are TRUE, its output becomes TRUE. When both inputs are FALSE, its output becomes FALSE. Otherwise the output remains unchanged. For therefore, Muller C-element to act as a Rendezvous circuit, its inputs must not change again until its output responds. A chain of Muller C-elements can control the flow of data down an electronic bucket brigade.

RENDEZVOUS CIRCUITS



Rendezvous circuit

Rendezvous circuit

It can coordinate the action of an asynchronous system, allowing data to flow in an orderly fashion without the need for a central clock. Shown here is an electronic pipeline control by a chain of Muller C-elements, each of which allows data to pass down the line only when the preceding stage is "full" – indicating that data are ready to move – and the following stage is "empty."

Each Muller C-element has two input wires and one output wire. The output changes to FALSE when both inputs are FALSE and back to TRUE when both inputs are TRUE (in the diagram, TRUE signals are shown in blue and FALSE signals are in red.). The inverter makes the initial inputs to the Muller C-element differ, setting all stages empty at the start. Let's assume that the left input is initially TRUE and the right input FALSE (1). A change in signal at the left input from TRUE to FALSE (2) indicates that the stage to the left is full – that is, some data have arrived. Because the inputs to the Muller C-element are now the same, its output changes to FALSE. This change in signals does three things: it moves data down the pipeline by briefly making the data latch transparent, it sends a FALSE signal back to the preceding C-element to make the left stage empty, and it sends a FALSE signal ahead to the next Muller C-element to make the right stage full (3)

ups recently introduced a new kind of Rendezvous circuit called GasP. GasP evolved from an earlier family of circuits designed by Charles E. Molnar, at SUN Microsystems. Molnar dubbed his creation nchronous symmetric pulse protocol (the asterisk indicates the to the name because GasP is what you are supposed to do when you ts go. It is found that GasP modules are as fast as and as energy- ents, fit better with ordinary data latches and offer much greater gns.

Without a clock to govern its actions, an asynchronous system must rely on local coordination circuits instead.

An arbiter circuit performs another task essential for asynchronous computers. An arbiter is like a traffic officer at an intersection who decides which car may pass through next. Given only one request, an Arbiter promptly permits the corresponding action, delaying any request until the first action is completed. When an Arbiter gets two requests at once, it must decide which request to grant first.

For example, when two processors request access to a shared memory at approximately the same time, the Arbiter puts the request into a sequence, granting access to only one processor at a time. The Arbiter guarantees that there are never two actions under way at once, just as the traffic officer prevents accidents by ensuring that there are never two cars passing through the intersection on a collision course.

Although Arbiter circuits never grant more than one request at a time, there is no way to build an Arbiter that will always reach a decision within a fixed time limit. Present-day Arbiters reach decisions very quickly on average, usually within about a few hundred picoseconds. When faced with close calls, however, the circuits may occasionally take twice as long, and in very rare cases the time needed to make a decision may be 10 times as long as normal.

The fundamental difficulty in making these decisions causes minor dilemmas, which are familiar in everyday life. For example, two people approaching a doorway at the same time may pause before deciding who will go through first. They can go through in either order. All that needed is a way to break the tie.

An Arbiter breaks ties. Like a flip-flop circuit, an Arbiter has two stable states corresponding to the two choices. One can think of these states as the Pacific Ocean and The Gulf of Mexico. Each request to an Arbiter pushes the circuit toward one stable state or the other, just as a hailstone that falls in the Rocky Mountains can roll downhill toward The Pacific or the Gulf. Between the two stable states, however, there must be a meta-stable line, which is equivalent to the Continental Divide. If a hailstone falls precisely on the Divide, it may balance momentarily on that sharp mountain ridge before tipping toward The Pacific or the Gulf. Similarly, if two requests arrive at an Arbiter within a few picoseconds of each other, the circuit may pause in its meta-stable state before reaching one of its stable states to break the tie.

THE NEED FOR SPEED

Research group at Sun Microsystems concentrates on designing fast asynchronous systems. We have found that speed often comes from simplicity. Our initial goal was to build a counter flow pipeline with two opposing data flows – like two parallel bucket brigades moving in opposite directions. We wanted the data from both flows to interact at each of these stages; the hard challenge was to ensure that every "northbound" data element would interact with every "southbound" data element. Arbitration turned out to be essential. At each joints between successive stages, an Arbiter circuit permitted only one element at a time to pass.

This project proved very useful as a research target; we learned a great deal about coordination and arbitration and built test chips to prove the reliability of our Arbiter circuits.

The experiments at Manchester, Caltech and Philips demonstrate that asynchronous microprocessors can be compatible with their clocked counterparts. The asynchronous processors can connect to peripheral machines without special programs or interface circuitry.

A CHALLENGING TIME

Although the architectural freedom of asynchronous systems is a great benefit, it also poses a difficult challenge. Because each part sets its own pace, that pace may vary from time to time in any one system and may vary from system to system. If several actions are concurrent, they may finish in a large number of possible sequences. Enumerating all the possible sequences of actions in a complex asynchronous chip is as difficult as predicting the sequences of actions in a school yard full of children. This dilemma is called the state explosion problem.

Can chip designers create order out of the potential chaos of concurrent actions?

Fortunately, researchers are developing theories for tracking this problem. Designers need not worry about all the possible sequences of actions if they can set certain limitations on the communication behavior of each circuit. To continue the schoolyard metaphor, a teacher can promote safe play by teaching each child how to avoid danger.

Another difficulty is that we lack mature design tools, accepted testing methods and widespread education in asynchronous design. A growing research community is making good progress, but the present total investment in clock-free computing parlances in comparison with the investment in clocked design. Nevertheless, we are confident that the relentless advances in the speed and complexity of integrated circuits will force designers to learn asynchronous techniques. We do not know yet whether asynchronous systems will flourish first within large computer and electronics companies or within start-up companies eager to develop new ideas. The technological trend, however, is inevitable: in he coming decades, asynchronous design will become prevalent.

CONCLUSION

Clocks have served the electronics design industry very well for a long time, but there are insignificant difficulties looming for clocked design in future. These difficulties are most obvious in complex SOC development, where electrical noise, power and design costs threaten to render the potential of future process technologies inaccessible to clocked design.

Self-timed design offers an alternative paradigm that addresses these problem areas, but until now VLSI designers have largely ignored it. Things are beginning to change; however, self-timed design is poised to emerge as a viable alternative to clocked design. The drawbacks, which are the lack of design tools and designers capable of handling self-timed design, are beginning to be addressed, and a few companies (including a couple of start-ups, Theseus Logic Inc., and Cogency Technology, Inc.) have made significant commitments to the technology.

Although full-scale commercial demonstrations of the value of self-timed design are still few in number, the examples available, demonstrates that there are no “show stoppers” to threaten the ultimate viability for this strategy. Self-timed technology is poised to make an impact, and there are significant rewards on offer to those brave enough to take the lead in its exploitation.

REFERNCES

- www.google.com
- www.wikipedia.com
- www.studymafia.org