Todd Goldfinger
Dr. Schumacher

Technology Forecast - Asynchronous Logic

Modern computers use almost exclusively what is known as synchronous design. This means the entire CPU is governed by one clock. The clock is set just slower than the slowest operation on the chip so that every electric signal is guaranteed to be a completed calculation. Most people are familiar with clock speed; it's usually the first thing displayed in an advertisement and it's what many people use as a performance measure. Asynchronous design is an alternative design methodology that uses no system clock. This design was used as early as the late forties, but it was quickly dropped in favor of the easier to implement synchronous design. The issue of asynchronous design went largely ignored until 1989 when Ivan Sutherland wrote a paper that brought the subject back to life [1].

Since that time, several experimental and commercial asynchronous processors have been built. Some of these include the Amulet series out of the University of Manchester, the processor(s) designed by the Cal-Tech spin-off Asynchronous Digital Design, and an asynchronous version of the Pentium engineered by Intel [1]. Furthermore, a yearly conference that started in 1994 is now held for asynchronous researchers and enthusiasts. There is also interest in applications for asynchronous logic that did not exist 50 years ago; encryption in smart cards is one. It is clear that there is resurgence in the interest of this technology. But where is this all going? Is it possible this is just a research fad? And if it's not, what might some of the possible outcomes be, and what are some useful strategies to take advantage of these outcomes? First, I look at the likelihood that this technology has a future. Then I look at a handful of scenarios and last some possible strategies for these outcomes.

Stop the Clock?

Exponential Growth of Clock Speed

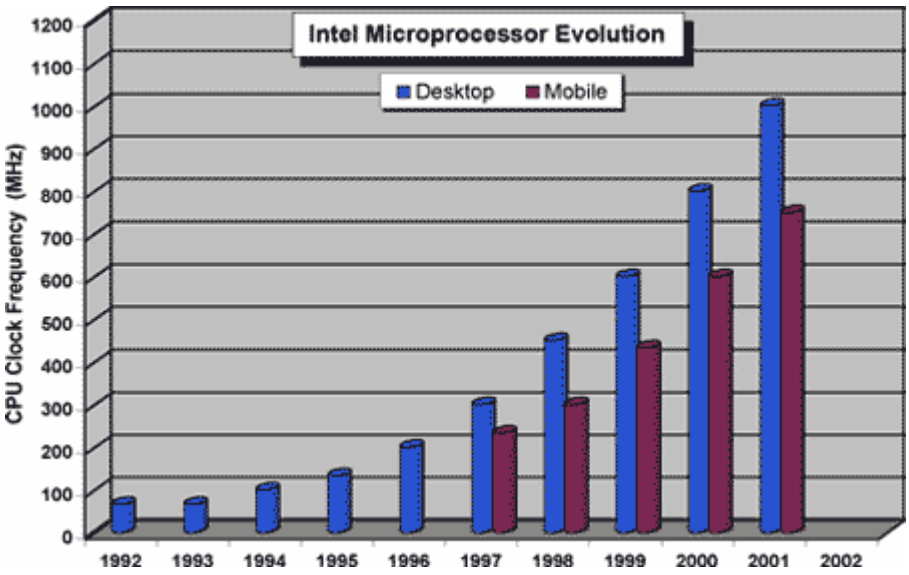

Fig 1. Plot of Clock Frequency vs Year [4]

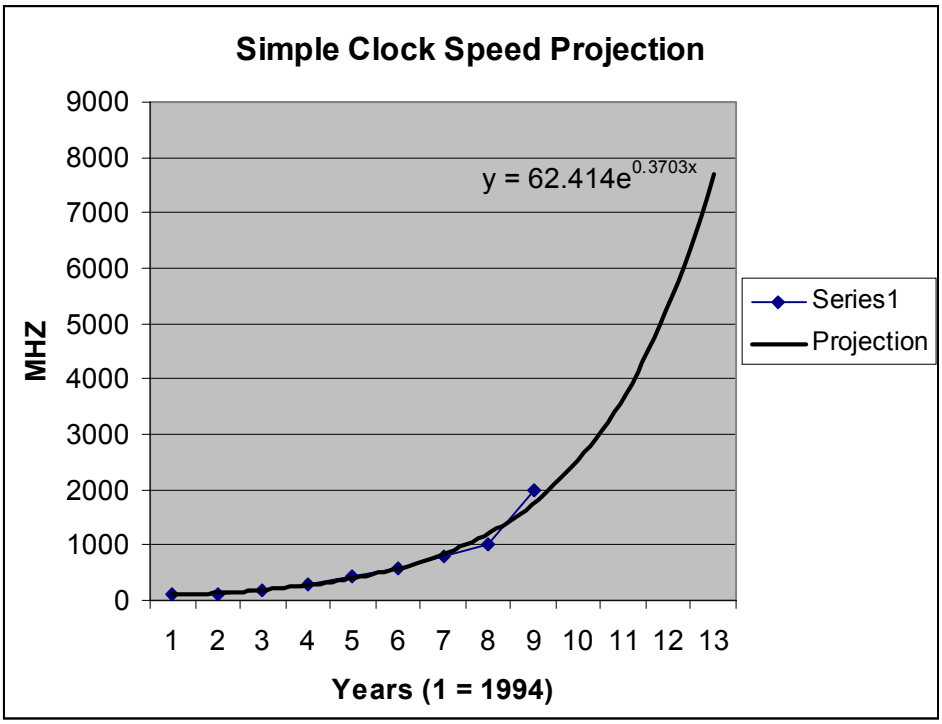

Fig 2. The series one line represents the data points (1,95), (2,115), (3,190), (4,295), (5,435), (6,590), (7,790), (8,1000), (9,2000). These are the points I took from Fig 1, as well as the last point, which represents Intel's current fastest microprocessor clock. My justification for dropping the first two points is that Intel had very little competition in the early 90s. Now there is AMD among others.

Figure 1 shows the progression of Intel's clock speeds over time. Based on Intel's size and dominance, I believe this is an accurate reflection of the industry as well. Figure 2 is a simple exponential projection of these points done in excel. The reason I was not worried about being very rigorous with this plot is because of its exponential nature. Under the assumption that it is in fact exponential (which seems clear to me from the first plot), even if the exponent is a little off, clock speeds could easily reach over one quadrillion hertz within my lifetime. If electrons could move at the speed of light (and they don't), then they could only go a tenth of a micrometer in a clock tick. This is the size of a transistor on a modern day microprocessor. Go just a few orders of magnitude smaller and this is the diameter of an atom of silicon. And if it seems like there are serious problems at those frequencies, keep in mind that engineers are already running into clock skew problems. This is in addition to the problems of electrons quantum tunneling [2,3] and increasing electronic interference. Hence it seems very unlikely that this curve will remain exponential for many decades longer. It will likely hit an inflection point and become an s-curve with a relatively short growth phase and a quick flattening out phase. I claim that staying with solely with synchronous design for the duration of my lifespan is extremely unlikely.

Reemergence of Asynchronous

Why is it that asynchronous logic makes a comeback now instead of some earlier or later decade? In previous decades, synchronous worked well and there was no need to look at other technologies. However, speed and number of transistors per chip have increased several orders of magnitude, and we are at a point where it's becoming clear that the number of problems with the current design will become more difficult very soon. I see it as no coincidence that asynchronous design methodologies have found their way back into industry just recently. It may be that there is a collective realization that there has to be change, or that a few key people have forced engineers in the industry to accept this. Either way, I think asynchronous has come back because limits to the current technology are in sight.

Scenarios

In the following situations I make a few assumptions. The rate of increase of clock speeds will slow dramatically or run into physical limitations within my life span. At some point in the future, one or more radically different technologies will replace or at least integrate with our current model of computing; it will solve all EMI and power consumption problems as well. I see clocked silicon based chips as one step in the evolution of computing. It started with people, moved into mechanical computing devices, calculators, vacuum tube based computers, transistor based computers, etc. If one company is very successful in designing and selling a fully functional asynchronous microprocessor, the rest of the industry will follow. Those who don't may quickly fall behind. Last, engineers are smart and find ways around apparent obstacles.

1) Asynchronous is rapidly accepted; radical technologies lie far in the future.

The current enthusiasm for asynchronous design accelerates and spreads through academia. As a result, new tools are developed, there is more experience floating around, classes in asynchronous design are being taught to undergraduates. Gradually this filters through into the industry and asynchronous becomes the accepted standard. This happens in a couple different ways. The first is in the mobile computing and DSP areas. Asynchronous design will be easily accepted in this area because it solves electromagnetic interference problems, uses less power in devices that use batteries, and design is easier for some small-scale devices. The adoption of asynchronous design should happen relatively quickly, especially where it may be absolutely necessary or at least makes the problems far easier to solve. The second area is in personal computers and other large-scale designs. The goal in this area is to make a general-purpose computer as fast as possible. There is no need to worry about power and EMI problems. So even though asynchronous design may be quickly accepted in academics, it may take a little longer for personal computers. In this scenario it takes several years longer to become mainstream than in the mobile computer area. After this time, different companies will jump from the synchronous s-curve to the asynchronous s-curve over a period of a few more years. Along the way, however, bits and pieces of modern synchronous processors will be converted to asynchronous but not the entire processor.

Finally, asynchronous design prospers in the industry for at least two to three decades before a newer technology comes into play. At this point asynchronous is well into its s-curve. Instead of being completely replaced, it is merged with this new technology and continues to live on in niche markets.

2) The synchronous s-curve is pushed to the limit; radical technologies lie far in the future.

Interest in asynchronous design cannot be sustained. It never catches on in the majority of major universities, and corporations decide not to invest any more money in the technology. Engineers think they can push the synchronous s-curve for several decades longer. They succeed in doing so by overcoming all sorts of obstacles time and time again. In the mobile computing industry, asynchronous does catch on, but only where it is absolutely necessary to continue developing better products. There is only limited experience and unsophisticated tools for use by companies who routinely use asynchronous design. The scenario breaks into two at this point.

A) Synchronous design continues to be successful until a radical new technology is developed. Asynchronous is almost completely forgotten except for in niche markets.

B) Physical limitations that cannot be solved begin to occur well before a new technology comes along. At this point engineers have no choice but to take another look at asynchronous design. Large companies and the few companies with asynchronous experience will thrive in this market. Other companies will struggle or go bankrupt because they don't have the money and resources to switch gears all of a sudden. As a result, there is a weeding out period in which many companies disappear because they are not able to keep up when the industry suddenly switches to another technology. This also means that in the following years there will be less competition and the computer industry will have a brief slow down. After a few years it will pick back up again. Soon a radically new technology will arrive, and asynchronous will be seen as a short bridge to gap the time between synchronous design and this new technology. The end result is that asynchronous still falls off into obscurity, but many companies that did not have experience with it will be wiped out.

3) Asynchronous is rapidly accepted; radical technologies are very near.

As in scenario one, asynchronous quickly becomes popular and is quickly accepted. This time, however, another technology comes in at about the same time and is superior to asynchronous design in many respects. It still takes time for this technology to gain acceptance because it is so different and no one has any experience with it. In the mean time asynchronous begins to replace synchronous and competes with this new technology for a time. Eventually the new technology becomes popular and well understood; asynchronous goes to niche markets and finds some applications in combination with the new technology. Because all three technologies are around within the same period of time, engineers are knowledgeable of both synchronous and asynchronous systems. So, many solutions to engineering problems (at least during the transition period) use a combination of synchronous, asynchronous, and this new technology.

4) The synchronous s-curve is pushed to the limit; radical technologies are very near.

There is never any need to switch to asynchronous design except for possibly in niche markets and in mobile computing for a short period. This new technology is quickly accepted and solves power, EMI, and problems of distributing the clock. While the industry gets over the learning curve, synchronous continues to thrive for several more years and ultimately is combined with the new technology in many applications.

5) Asynchronous competes with many new technologies.

Just as asynchronous is becoming widely accepted in academics and the engineering community, there is an explosion in new and fundamentally different technologies. These technologies aren't necessarily superior to synchronous or asynchronous design, but each solves at least one problem with current day microprocessors. Some of them fix EMI problems, others are extremely low power, and others have the potential to grow exponentially faster for at least another generation or two. Some of them are used in niche markets and have entirely different uses than modern processors, and some have a combination of these problems. The hardware

market has essentially become a scaled down software market in terms of choices of different technologies. Some companies adopt asynchronous design; specifically the ones that have already begun to invest in it. In the personal computer, asynchronous never takes hold because there are other technologies in their infancy that offer far better prospects for speed. Individual engineers are familiar with at most two or three of these technologies, and the managers are responsible for knowing which technology to use to solve a particular problem. So asynchronous is well understood by only certain people and there are university classes that teach it, but not many.

6) Synchronous outlives asynchronous.

As asynchronous begins to become popular, it becomes apparent that synchronous still has a way to go. Eventually, asynchronous turns out to be mostly hype and the industry continues to ride the synchronous s-curve while leaving asynchronous behind. However, asynchronous still finds applications in mobile computing and as a way of saving power in critical situations. There is a lot of experience in asynchronous, but one cannot assume that every engineer is familiar with this design methodology.

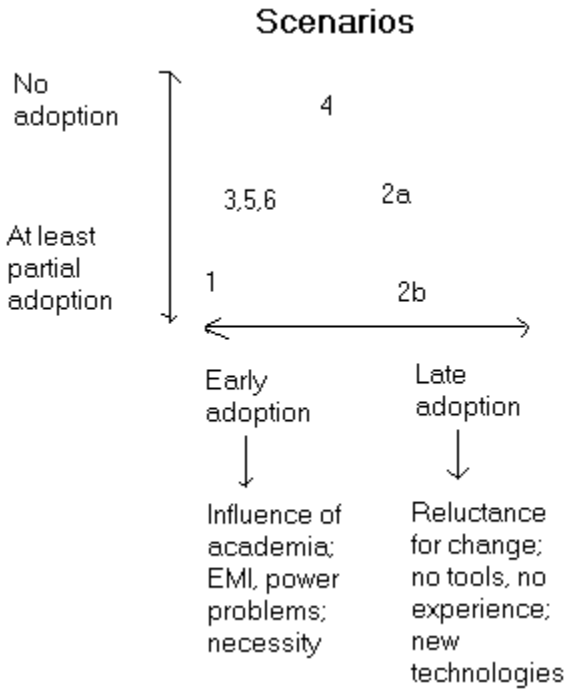

Fig 3. A picture showing where these strategies lie.

Strategies

Now that we have seen some possible outcomes of the resurgence of asynchronous design, we will look at what choices a company that is looking at asynchronous design has. For each scenario, there are three strategies. 1) Ignore asynchronous design. 2) Make the switch as early as possible (now). 3) Wait until the last possible minute. For each strategy there is a possible range of values of -4 to 4. I add a point if the strategy results in choosing the right technology for the mobile computing market at an early stage. I add another point if it also works out in the long run. I do the same for the personal computer and super computer industry. The strategy gets a negative point if they choose the wrong technology. No points are assigned for situations where it doesn't matter.

	Strategies 1	2	3
Scenarios 1	-4	4	2
2a	+4	-4	-2
2b	0	+4	+2
3	-1	+3	+1
4	+3	+1	+1
5	0	0	0
6	0	0	0

Fig 4. Evaluation Matrix

From figure 4 it seems that strategy two dominates strategy three in most cases. However, it doesn't do significantly better than strategy one; so, I don't draw any conclusions there. Strategy three seems the safest strategy because it has the least varying results; it has the least chance for reward but also the least chance for disaster.

Unfortunately, this doesn't tell us much more because of the simplicity. There are a number of problems. Some of the numbers were difficult to choose and so ended up being somewhat random looking. Also, some situations don't even make sense. For strategy three, no one would enter the market late if asynchronous design doesn't catch on. Choosing more sophisticated evaluation criteria can solve this. One might include the fact that getting into asynchronous early is a huge advantage if it turns out to be successful. There may also be many more strategies. A company might only want to train its engineers in asynchronous design instead of both training them and buying expensive new equipment and designing special purpose programs for design. This could save a lot of money if asynchronous design turns out to be bogus, but if it's successful, the company will lose several months buying and installing new equipment. Furthermore, this chart doesn't take into account extreme scenarios like 2b. In this scenario, not getting into asynchronous early could be a terminal error for small corporations. A much more in depth analysis using some of these ideas may show more information from these simple situations I have presented.

Closing Remarks

In the last decade, asynchronous design has become a feasible option in the minds of some researchers. It has gained some acceptance and there are now commercial computer chips that are fully or mostly asynchronous. I believe that asynchronous design has a reasonable chance of replacing synchronous design in the near future if there is no other superior technology. This is due to physical limitations of synchronous logic as well as other inherent problems. By looking at a number of possible future scenarios, we see that in some situations asynchronous can thrive and in others it is lost forever. However, I believe that the most likely scenarios are the middle ground ones, where asynchronous is semi-successful for at least a short time and eventually it becomes just another tool engineers can use to solve computing problems. The extreme scenarios are upper and lower bounds for the range of possibilities. We can see from analyzing these simple scenarios that if a company decides to take the route of asynchronous logic, most scenarios favor early adoption. However, only scenario 2b proves to be very detrimental to completely ignoring asynchronous logic.

Sources

[1] Tristram, Claire. Clockless Chips. Technology Review; pp 37. October 2001.

[2] Gates, Brian. Interview with roommate. Undergraduate Chemical Engineer.

[3] The Future of Transistors.
http://www.pbs.org/transistor/background1/events/transfuture.html.

[4] Glover, Kerry; Jones, Steve. System Power Management.
http://www.chipcenter.com/analog/c040.htm.