The Return of Asynchronous Logic

There is a world-wide resurgence of interest in asynchronous logic design techniques. After two decades during which clocked logic has imposed its discipline across all corners of the world of digital logic, the older and more anarchic approach seems poised to make a come-back.

Asynchronous design

Virtually all digital design today is based on a synchronous approach. The total system is designed as the composition of one or more subsystems where each subsystem is a clocked finite state machine; the subsystem changes from one state to the next on the edges of a regular clock. The state is held in a set of flip-flops (registers), and combinatorial logic is used to derive the new state and outputs from the old state and inputs. The new state is copied through the flip-flops on every rising edge of the clock signal. Special techniques are required whenever a signal crosses into the domain of a particular clock (either from outside the system or from the domain of a different clock within the same system), but otherwise the system behaves in a discrete and deterministic way provided a few rules are followed; these rules include managing the delays of the combinatorial logic so that the flip-flop set up and hold times are met under all conditions.

Asynchronous design does not follow this methodology; in general there is no clock to govern the timing of state changes. Subsystems exchange information at mutually negotiated times with no external timing regulation.

In the beginning

Early digital computers embodied a variety of design styles. Though most designs were based on a central timing generator which kept all the functions of the machine in lock-step, there were some examples of asynchronous machines. Examples in the USA include the ORDVAC built at the University of Illinois and the IAS built by John von Neumann's group at the Institute for Advanced Study at Princeton University. Both these machines were first operational in 1951-52 and operated without any central clock. The pioneering British teams preferred the synchronous approach, though later the MU5 built at Manchester University between 1969 and 1974 used asynchronous control.

Ruled by the clock

For the last two decades asynchronous design has all but disappeared from sight. The clocked approach has led to dramatic progress in the architectures of machines and in the productivity of designers. When all the state in a design changes at the same time, verifying the design becomes a matter of checking the delays in the combinatorial logic functions between the registers. This is a straightforward process compared with validating an asynchronous circuit, where the design must be examined in very fine detail for critical races and unstable states, and then the higher levels checked for liveness and similar properties.

As synchronous techniques have improved, tools have been developed which automate most of the design processes. The design of a complete chip can now be synthesized from a high level behavioural description with minimal manual intervention. Synchronous design is at the top of the learning curve and seems set to continue its domination of digital systems for the forseeable future.

Clock limitations

Though synchronous design has enabled great strides to be taken in the design and performance of computers, there is evidence that it is beginning to hit some fundamental limitations. A circuit can only operate synchronously if all parts of it see the clock at the same time, at least to a reasonable approximation. However clocks are electrical signals, and when they propagate down wires they are subject to the same delays as other signals. If the delay to particular part of the circuit takes a significant part of a clock cycle-time, that part of the circuit cannot be viewed as being in step with other parts.

For some time now it has been difficult to sustain the synchronous framework from chip to chip at maximum clock rates. On-chip phase-locked loops help compensate for chip-to-chip tolerances, but above about 50MHz even this isn't enough.

Building the complete CPU on a single chip avoids inter-chip skew, as the highest clock rates are only used for processor-MMU-cache transactions. However, even on a single chip, clock skew is becoming a problem. High-performance processors must dedicate increasing proportions of their silicon area to the clock drivers to achieve acceptable skew, and clearly there is a limit to how much further this proportion can increase. Electrical signals travel on chips at a fraction of the speed of light; as the tracks get thinner, the chips get bigger and the clocks get faster, the skew problem gets worse. Perhaps the clock could be injected optically to avoid the wire delays, but the signals which are issued as a result of the clock still have to propagate along wires in time for the next pulse, so a similar problem remains.

Even more urgent than the physical limitation of clock distribution is the problem of heat. CMOS is a good technology for low power as gates only dissipate energy when they are switching. Normally this should correspond to the gate doing useful work, but unfortunately in a synchronous circuit this is not always the case. Many gates switch because they are connected to the clock, not because they have new inputs to process. The biggest gate of all is the clock driver, and it must switch all the time to provide the timing reference even if only a small part of the chip has anything useful to do. Often it will switch when none of the chip has anything to do, because stopping and starting a high-speed clock is not easy.

Early CMOS devices were very low power, but as process rules have shrunk CMOS has become faster and denser, and today's high-performance CMOS processors can dissipate 20 or 30 watts. Furthermore there is evidence that the trend towards higher power will continue. Process rules have at least another order of magnitude to shrink, leading directly to two orders of magnitude increase in dissipation for a maximum performance chip. (The power for a given function and performance is reduced by process shrinking, but the smaller capacitances allow the clock rate to increase. A typical function therefore delivers more performance at the same power. However you can get more functions onto a single chip, so the total chip power goes up.) Whilst a reduction in the power supply voltage helps reduce the dissipation (by a factor of 3 for 3 Volt operation and a factor of 6 for 2 Volt operation, relative to a 5 Volt norm in both cases), the end result is still a chip with an increasing thermal problem. Processors which dissipate several hundred watts are clearly no use in battery powered equipment, and even on the desktop they impose difficulties because they require water cooling or similar costly heat-removal technology.

As feature sizes reduce and chips encompass more functionality it is likely that the average proportion of the chip which is doing something useful at any time will shrink. Therefore the global clock is becoming increasingly inefficient.

Asynchrony fights back

If global synchrony is becoming ever more difficult to establish and the clock is resulting in increasing power inefficiencies, it seems natural to revisit the whole issue of asynchronous versus synchronous design. In academic circles asynchronous techniques have always retained a niche because they provide a good framework for some mathematical techniques for proving the correctness of circuits. Now, however, industrial research organisations are becoming active in asynchronous design. There aren't any products on sale today which embody the results of this renewed interest, but there is a strong feeling that interest is increasing and asynchronous approaches are set to make a come-back.

What has happened to all the problems that caused the world to go synchronous in the first place? These have not just ceased to exist, but they have become less overwhelming for the designer. Over the years the academics have improved the methodologies which can be applied to the design of asynchronous circuits, so a lot more is known about the reliability and correctness of potential solutions. But perhaps most important is the increasing capability of VLSI technology. This enables ready-made solutions to standard problems to be prepackaged in a design library, removing the need for each designer to understand fully all the intricacies of asynchronous design. There are now so many transistors on a chip that the inefficiency of using a standard asynchronous solution is negligible compared with the extra design cost of customising each occurrence.

Asynchronous design

It is a common misconception to view asynchronous design as a single alternative to synchronous design. It is more accurate to view synchronous design as a special case representing a single point in a multi-dimensional asynchronous design space. There are many different flavours of asynchronous logic, and they are as different from each other as they are from synchronous design. However there are a few key features which describe most current approaches, and these can be seen as binary choices:

- dual rail encoding vs. data bundling. In dual rail encoded data, each boolean is implemented as two wires. This allows the value and the timing information to be communicated for each data bit. Bundled data, on the other hand, has one wire for each data bit and a separate wire to indicate the timing.

- level vs. transition encoding. Level sensitive circuits typically represent a logic one by a high voltage and a logic zero by a low voltage. Transition signalling uses a change in the signal level to convey information.

- speed-independent vs. delay-insensitive design. 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 signalling. 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 is therefore ideal for automatic transformation into silicon layout as the rather arbitrary delays introduced by the layout compiler can't affect the functionality (though they will affect the performance).

The most popular form in recent years has been dual-rail encoding with level sensitive signalling. Full delay insensitivity is still achieved, but there must be a "return to zero" phase in each transaction, and therefore more power is dissipated than with transition signalling. The advantage of this approach over transition signalling is that the logic processing elements can be much simpler; familiar logic gates process levels whereas the circuits required to process transitions require state and are generally more complex. The Tangram compiler developed at Philips Research Laboratories in the Netherlands uses this style to transform high-level function specifications written in an Occam-like language into silicon layout, a technique named "VLSI programming" by its inventors [1].

In 1987, Ivan Sutherland chose the topic of asynchronous design for his Turing Award lecture. He presented an approach developed by Sutherland, Sproull and Associates called "Micropipelines" [2]. This approach uses bundled data with a transition signalled handshake protocol to control data transfers. This is illustrated in figures 1 and 2. Figure 1 shows the interface between the sender and receiver. There is a *bundle* of data which carries information (using one wire for each bit) and two control wires: *Request* from the sender to the receiver carries a transition when the data is valid; *Acknowledge* from the receiver to the sender carries a transition when the data has been used. The protocol sequence is illustrated in figure 2. This defines only the sequence in which events must occur - there is no upper bound on the delays between consecutive events.

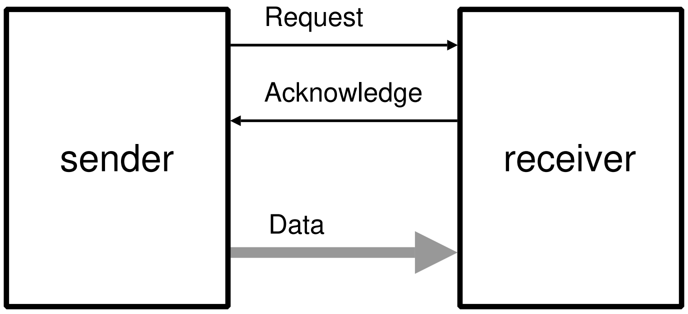

Figure 1: The bundled data interface used in a micropipeline

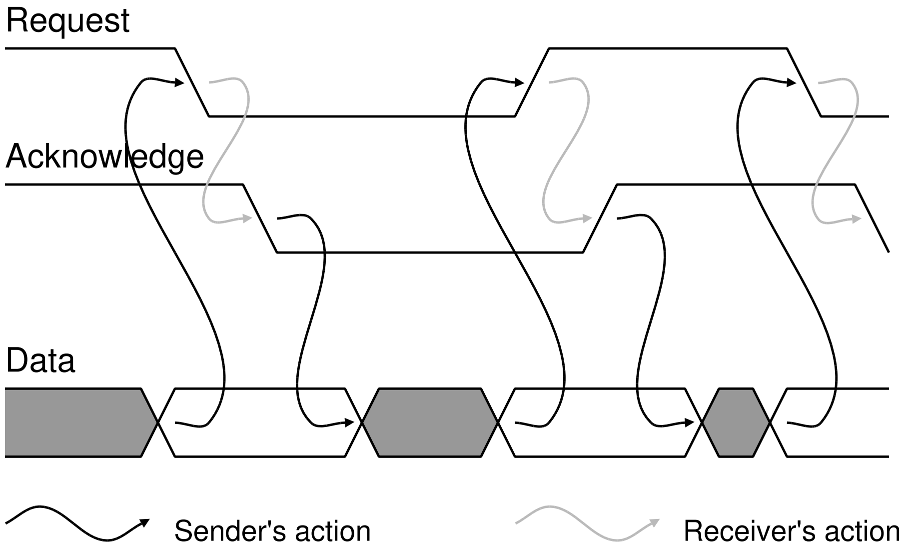

Figure 2: The bundled data interface protocol

1. Sender prepares data
2. Sender issues Request
3. Receiver accepts data
4. Receiver issues Acknowledge
5. Sender may remove data at will

Once the data bundling constraints are met (ie the order of events corresponds to that shown in figure 2), a micropipeline is delay-insensitive. The event processing blocks which are required to control the operation of all but the simplest micropipelines are well defined and relatively straightforward to design using existing asynchronous design techniques. Once designed, these blocks can be placed in a library and reused to construct arbitrarily complex control structures. No sophisticated design techniques are required to put these circuits together; conventional behavioural simulation is a useful aid, as is a large whiteboard. The only uniquely asynchronous problem which recurrs is the high-level issue of liveness (absence of deadlock).

The AMULET project at Manchester University adopted the micropipeline approach for the work described below.

The AMULET project

The AMULET project was established in October 1990 to investigate the application of asynchronous techniques to the power reduction of high-performance microprocessors. This activity was part of the ESPRIT funded OMI-MAP project (Open Microprocessor systems Initiative - Microprocessor Architecture Project), and was one component of a broad look at power dissipation issues at the gate, chip and board level. The objective was to investigate thoroughly the potential of asynchronous approaches by designing and building a complete implementation of the ARM processor in Sutherland's micropipeline style.

Although a complete asynchronous microprocessor has been built before [3], this is the first attempt at a full-functionality design including the difficult area of exact exceptions for memory faults. New solutions were required for several areas of the design, for instance a novel register locking technique was developed to manage dependencies between the registers used by successive instructions [4]. The ALU exploits data dependent timing to deliver significantly faster operation for typical operands than for worst-case values [5]. The processor allows a considerable level of internal concurrency, with all the pipeline stages operating autonomously and only interacting with nearest neighbours when results are ready to be passed on. One result of the autonomy is that the depth of instruction prefetching beyond a branch is fundamentally non-deterministic (though it is bounded, and the instructions which are allowed to execute are deterministically defined!).

The initial phase of this work is now drawing to a close. The asynchronous ARM design is complete, and was submitted to fabrication in February 1993. Although the final verdict must await inspection of the silicon, it is already clear that this scale of design is now quite practical with existing design tools. The design resource required to complete the design was comparable with the clocked part, and although the transistor count and die area are rather larger than the synchronous version, most of the difference may be accounted to the deeper pipeline of the asynchronous design. A synchronous implementation which used the same pipeline structure would require perhaps 10 or 20% less area.

Future prospects

It is possible that all the renewed interest in asynchronous techniques will come to nothing, though this seems unlikely. It is also possible that industry will suddenly see the asynchronous light and switch completely to he new approach. This seems even more unlikely!

What seems more likely is that areas will be identified where asynchronous approaches have really worthwhile advantages; these will be niches in otherwise synchronous designs. The things to look for are power sensitive (rather than ultimate performance) applications and functions with a highly variable workload (such as CD error correctors and personal computer processors). A variable workload makes the most of the power saving potential of asynchronous logic, as a synchronous approach requires the clock to be set to match the peak load. Look for places where there is no external clock constraint; long serial lines and VDU drivers interface to explicit or implicit clocks at the other end of the wire, so these are not likely to go asynchronous for a while!

The CPU/cache combination on portable equipment is a subsystem which seems to meet all these requirements, so the AMULET work is continuing within the ESPRIT OMI/DE project (which is concerned with Deeply Embedded processor macrocells, which are CPUs embedded together with application specific logic functions on the same chip) to add a cache to the current integer unit.

Asynchronous logic can be expected to begin winning niches in the digital electronics business within the next few years. It will share circuit boards with clocked chips and integrated circuits with clocked subcircuits. It will become established as a viable alternative technology in many areas, and the technology of choice for some.

References

[1] van Berkel C H, Rem M and Saejis R W J J, "VLSI Programming". Proceedings of ICCD'88, 1988, pp 152-156.

[2] Sutherland I E, "Micropipelines", Communications of the ACM, vol 32, no 6, June 1989, pp 720-738.

[3] Martin A J et al, "Design of an Asynchronous Microprocessor", Advanced Research in VLSI 1989, pp 351-373.

[4] Paver N C, Day P, Furber S B, Garside J D and Woods J V, "Register Locking in an Asychronous Microprocessor", Proceedings of ICCD '92, October 1992, pp 351-355.

[5] Garside J D, "A CMOS Implementation of an Asynchronous ALU", Proc. of the IFIP Working Conference on Asynchronous Design Methodologies, Manchester, England, 1993.

Illustrations

MU5: A mainframe computer built at the University of Manchester between 1969 and 1974 which operated asynchronously.

AMULET1: A micropipelined (asynchronous) implementation of the ARM microprocessor designed at the University of Manchester between 1990 and 1993.