Real-Time without Clocks?

Steve Furber
Department of Computer Science, The University of Manchester,
Oxford Road, Manchester M13 9PL England

27 March 1996

Abstract

For the last 20 years digital design has been based on the use of a central clock, but it hasn't always been so, and it may not always be so in the future. Where ultimate power-efficiency, performance or electromagnetic compatibility are required, the clock may be the wrong answer. Recent work within the OMI and elsewhere has pointed the way to a future where VLSI devices can more readily be designed without clocks, and the advantages of so-doing are becoming increasingly apparent.

Introduction

The spectacular progress in the performance and power-efficiency integrated electronics over recent years has put computing power into portable consumer products that mainframe designers could only dream about a couple of decades ago. As semiconductor manufacturing technology has improved by a factor of two every 18 months, designers and CAD tool suppliers have barely kept pace with the ever-increasing resources placed at their disposal. An important factor in staying with the pace has been the adoption of a simple design methodology based around a central clock. All the state changes on a chip are synchronized to the clock and the whole design can be viewed as a single, synchronous, finite state machine.

However, over the last decade or so, interest in alternative design methodologies has been growing steadily. These alternative approaches dispense with the central clock and are therefore termed *asynchronous*. Since the clock has been so successful for so long, why should there be a move to abandon it? To understand this, it is necessary to look at the way chip technology and the market for it are developing.

Chip technology

The major factor in advancing chip technology is the reduction in the size of a transistor from one technology generation to the next. As transistors shrink they get cheaper, faster and use less power. This win-win scenario has carried the industry forward for several decades and, although there are signs that we may be approaching the limit of this process, there is still some way to go. Within the next decade the technology will move to 0.1 micron feature sizes. At this scale a chip will offer 1,000,000,000 transistors and designs such as microprocessors will operate at over 1 GHz. Unfortunately, although transistors and logic gates get faster as the feature size reduces, long wires get slower, so clock skew gets increasingly difficult to manage. More and more power goes into generating the required clock precision, creating thermal problems and increasing the packaging costs.

Consumer electronics

While chips are becoming hotter and more powerful, the products they are built into are becoming smaller and more portable. Mobile telephones, CD players, lap-top computers and personal digital assistants point the way to a future where the machine goes with the user rather than the user going to the machine. Such portable products require as much computing power as can be made available, but they also require power-efficient components to maximize battery life and high electromagnetic compatibility to minimize electromagnetic pollution (both to minimize interference with other radio equipment and because the equipment itself will often use low-power radio communications).

Clearly there is a conflict between the needs of these products and the way high-performance clocked chips are developing. The solution? Get rid of the clock!

Self-timed design

Without the clock, some other mechanism is required to control state changes within the circuit. Self-timed circuits replace the central control of the clock with a devolved control system based on local ‘handshakes’ (see inset on ‘the workings of an asynchronous logic circuit’). Since the handshakes are local they do not suffer from the delays that clocks must suffer through passing down long, very narrow tracks, and they take correspondingly less power.

Local control means that cycle times can adjust to local conditions, whereas the clock cycle time must be set to suit the slowest logic under the slowest data and environmental conditions. Those parts of a self-timed circuit that are inactive need not cycle at all, so they consume no power (on CMOS). It also means that different parts of a circuit switch at different times, spreading the current peaks and reducing the radio emissions - much as an army breaks step when crossing a bridge to avoid exciting a resonance in the bridge structure, so the activity in an asynchronous circuit avoids the coherent switching of a clocked chip which results in so much radio output at harmonics of the clock frequency.

Self-timed design isn’t all good news. At the low level, designers have to re-learn the techniques of hazard-free logic design which they were taught at university but have long-since forgotten, since clocked circuits wait until all the glitches have settled before sampling the outputs. At a higher level, the design is no longer a single finite state machine but a concurrent composition of many independent state machines. As with all concurrent systems there are new problems to contend with, ensuring the system is free from deadlock, livelock and so on. The CAD tools which are so vital to today’s complex circuits are not well suited to asynchronous design either. Since clocks are so pervasive, some tools won’t tolerate logically redundant gates and will actually optimize out the gates you have carefully put in to make the circuit hazard free! Testing is also a potential problem area since the industry has a major investment in production test equipment which has a very synchronous view of the chips it is to test.

Self-timed experience

There are many examples of successful self-time designs around the world, including several within the OMI. The leading self-timed methodology has been developed at Philips who have a synthesis route based upon their ‘Tangram’ language. Within the OMI EXACT (Exploitation of Asynchronous Circuit Techniques) project they developed and fabricated an asynchronous error corrector for the digital compact cassette player which demonstrated a factor five power advantage over the best clocked design.

The University of Manchester worked with Philips in EXACT, and has also developed self-timed versions of the ARM microprocessor. In OMI-MAP (Microprocessor Architecture Project) we developed AMULET1, the world’s first fully asynchronous implementation of a commercial microprocessor architecture. AMULET1 first ran in 1994, and executes code produced by the standard ARM software development tools. It demonstrated the feasibility of developing complex self-timed circuits. AMULET2e was developed in the OMI-DE/ARM (Deeply Embedded ARM Macrocells) project, and is expected to demonstrate the advantages of the asynchronous approach for embedded systems. Both AMULET chips were developed using a conventional design flow based around Compass Design Automation tools, though AMULET2e also benefited greatly from accurate timing simulation using TimeMill from EPIC. The development effort and costs were comparable with those for a clocked processor.

Asynchronous real-time?

The clock in a conventional synchronous circuit doesn't only control state changes, it also provides a very accurate timing reference. The state change control can readily be replaced by local handshakes, but what about the timing reference, which may be particularly important in real-time applications? Also, the argument about typical versus worst-case delays is all very well when the issue is average throughput, but in a real-time system we are up against hard constraints and only worst-case considerations apply.

There is no doubt that a crystal oscillator provides a timing reference that is hard to beat for accuracy by other means. However, the timing reference rarely requires to be at the processor clock frequency. More often it is required at the data rate which is considerably lower. A low frequency crystal can awaken the processor which can complete its task in however long it takes and then go back to sleep, using just the energy required to process that sample.

Likewise, hard real-time constraints are usually single-sided. Provided the processor can perform the necessary tasks in less than the allotted time, all will be well. The clocked chip is *always* delivering according to worst case conditions, since that is where the clock must be set. The asynchronous chip is *usually* better (and can exploit this margin to save energy), but under worst-case conditions sinks to the performance of the clocked chip.

Conclusion

The end of the clocked chip is not yet nigh, but the writing is on the wall, at least for those applications which demand the performance that will become available with near-future process technologies that drive clocks beyond their limits. Today abandoning the clock represents a risk since there are weaknesses in design experience, tools support and production test methodologies. But the imperatives of power-efficiency, performance and electromagnetic compatibility are driving several companies (including some of the world's major electronics companies) to take a careful look at asynchronous design.

The workings of an asynchronous logic circuit

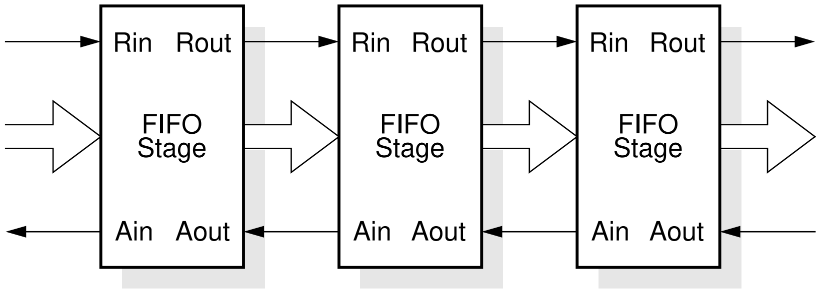

Figure 1 An asynchronous FIFO.

A simple asynchronous FIFO operates with a *Request* signal and an *Acknowledge* signal connecting consecutive stages. The *Request* signal says when new data is available from one stage and the *Acknowledge* signal says when it has been accepted by the next stage. A stage latches new data when its old data has already been accepted by the next stage (as signalled on *Aout*, the output *Acknowledge*) and the availability of new data is signalled by the previous stage (on *Rin*, the input *Request*); whichever of these is last defines the timing of the latch closure.

The simplest form of *Request* and *Acknowledge* signalling uses transitions rather than levels; the first data is signalled by a rising edge and the second by a falling edge. The synchronisation in each stage requires a gate which waits until both inputs have risen before its output rises, and likewise waits until both inputs have fallen before its output falls. This is a ‘Muller C’ gate, named after its creator who devised it in the 1960s. Asynchronous design is not new!

AMULET processor core organization

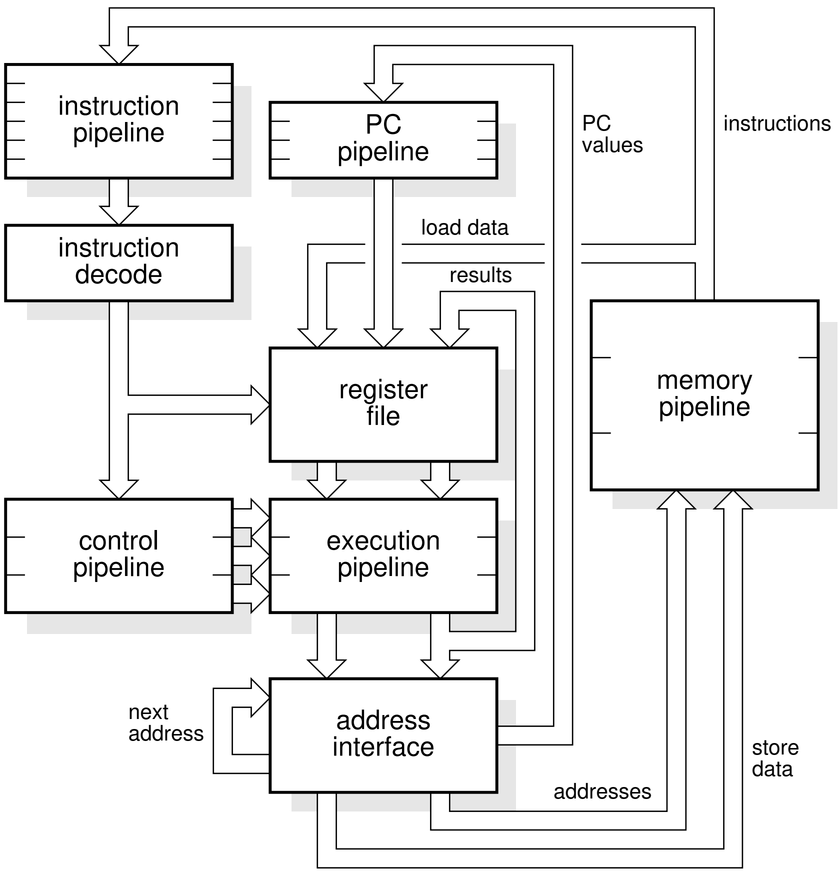

Figure 2   AMULET processor core internal organization.

The asynchronous AMULET processor cores can be viewed as sets of interacting self-timed pipelines with internal processing where needed. The *address interface* generates sequential instruction fetch requests; the *memory pipeline* supplies the instructions which flow into the *instruction pipeline* where they are held before being sent, in turn, to the *instruction decode* logic. The decoded instructions access their operands from the *register file* and flow down the *execution pipeline*; the *control pipeline* performs further instruction decoding and supplies control information to the *execution pipeline*.

When the instruction is a memory load or store, the *execution pipeline* computes the address which is interleaved with the instruction fetch stream in the *address interface* and the data steered back from the *memory pipeline* to the target register.

Control flow instructions (jumps and branches) in the ARM are program counter (pc) relative, so they collect the pc value from the *pc pipeline*, compute the target address in the *execution pipeline* and feed the result to the *address interface* to switch the instruction fetch stream to the new code address.

The AMULET2e embedded system chip

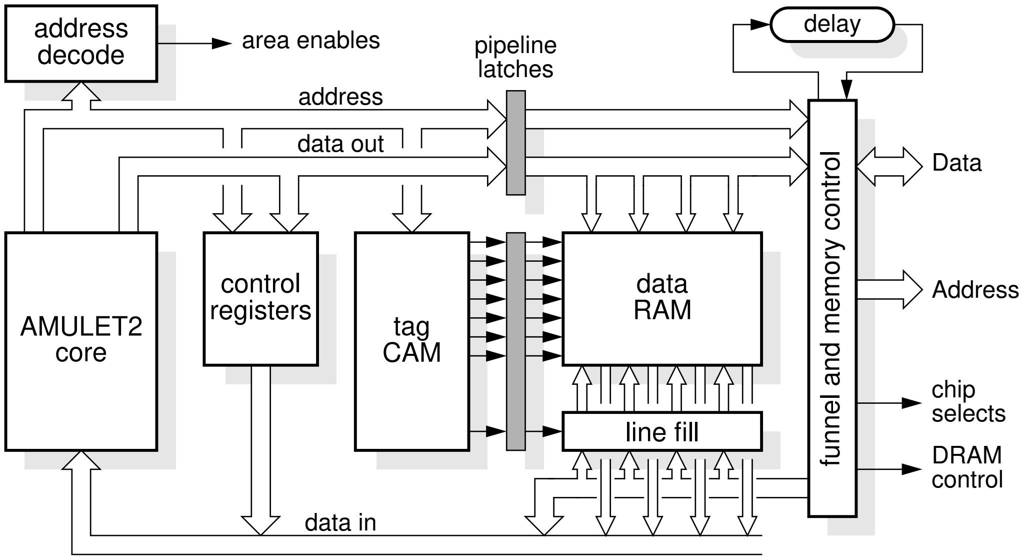

Figure 3 AMULET2e internal organization.

The AMULET2e self-timed embedded system chip incorporates an AMULET2 processor core, 4 Kilobytes of RAM which may be configured to operate either as a cache or as a fixed memory-mapped RAM and various control registers which configure the chip to operate with 8-, 16- or 32-bit off-chip devices with programmable timing characteristics.

The off-chip design environment is conventional: address and data buses connect to standard peripheral and memory devices, and chip select lines provide the timing information. DRAM control lines are also provided. All off-chip accesses are timed in multiples of the *reference delay*, a single off-chip delay line. On-chip access are completely self-timed. A 32 KHz clock crystal provides a real-time reference and controls the DRAM refresh period.

Author’s biography

S.B. Furber MA, PhD (Cantab.), CEng, MBCS.

Steve Furber is the ICL Professor of Computer Engineering in the Department of Computer Science at Manchester University. Prior to this appointment he was in charge of the hardware development group within the R&D department at Acorn Computers Ltd., and was a principal designer of the BBC Microcomputer and the Acorn RISC Machine (a 32-bit RISC microprocessor for low-cost applications), both of which earned Acorn Computers a Queen's Award for Technology. He also led the team which developed the hardware architecture and VLSI components for the Acorn Archimedes. Since moving to the University of Manchester in 1990 he has established a research group with interests in asynchronous logic design, and has secured funding from the CEC (ESPRIT) and the UK government (DTI/SERC) to develop asynchronous design methodologies within a range of industrial collaborative projects related to low-power CMOS and high-speed bipolar processor design and portable consumer applications. One result of this work, the AMULET1 asynchronous microprocessor, has been recognised with a 1995 British Computer Society Award.

The University of Manchester Department of Computer Science

The Department of Computer Science at the University of Manchester was founded in 1964 and is thus the oldest university computer science department in the United Kingdom. The Department was formed as a direct result of the success of the Manchester Mark 1 computer, the world's first stored program computer, developed by F.C. Williams and T. Kilburn in the late 1940s. Many famous pioneers of Computing were associated with those early days at Manchester including Alan Turing who worked in the Mathematics department from 1948 to 1954 and was Deputy Director of the Computer Laboratory around the time of the Mark 1. Subsequently, a number of experimental computers, notably Mercury (c. 1957), Atlas (c. 1962) and MU5 (c. 1972), were built by teams led by Professor Tom Kilburn. Probably the most important of these was Atlas whose 'One Level Storage System' was the forerunner of modern virtual memory.

Present research activities are spread across the broad range of specialist fields that constitute modern-day computer science. There are world-renowned research groups working on parallel computer systems architectures, design methods (especially formal methods) for hardware and software, the design of low power asynchronous VLSI circuits, the design and application of information systems, artificial intelligence and robotics. The Department has been awarded the highest grade in all of the three research selectivity rating exercises that have been conducted, by the UK University funding councils, since 1985. The recent teaching quality assessment exercise graded the teaching in the department as excellent, putting it among a very small number of UK Computer Science departments with the highest ratings in both the major activities of research and teaching.