Transforming Architectural Models Into High Performance Concurrent Implementations

L.E.M. Brackenbury, S.B. Furber and R. Kelly
Department of Computer Science, Manchester University,
Oxford Road, Manchester M13 9PL

Abstract

In this paper, the advantages of an asynchronous approach for efficient, concurrent operation are presented. Different approaches to implementing asynchronous designs are summarised leading to the selection of bundled-data transition signalling. Some key differences between CMOS and differential bipolar design are discussed with reference to their impact upon the microprocessor design. Finally, the performance potential of this approach is given, indicating the benefits of this work.

Introduction

Current high performance chip designs are devoting a significant silicon area and high power to generate and meet the speed of the global clock required in a synchronously timed system. Asynchronous techniques, whereby units on an integrated circuit operate at their natural speed, when ready, offer a route to significant design simplification, smaller area and lower power. This in turn should result in increased processor throughput.

TAM-ARM is a collaborative project between ARM Ltd., GEC Plessey Semiconductors (GPS) and Manchester University. The goal is to model and implement a high performance asynchronous design of the 32-bit ARM RISC microprocessor architecture using the architectural asynchronous techniques developed at Manchester University. The implementational tools to support this design approach are being developed by ARM Ltd. and the design will be fabricated on GPS's new high-speed differential bipolar process.

Early Asynchronous Systems

Although some early computers used asynchronous techniques, whereby parts operated when particular states were detected, this approach quickly fell out of favour. The reasons for this are not hard to see. Firstly, the approach to control tended to be ad hoc with each designer adopting their own informal approach. Secondly, design tools provided inadequate support - making debugging and monitoring of the hardware difficult. This resulted in designs which were often only comprehendable to the original designer. Furthermore, designs relied upon the designer being able to predict all possible variations of states for control, leading to designs which were only as good as the designer. Not surprisingly, in many cases this led to a very large number of modifications to the control during commissioning. Since designs at this time were at the level of discrete or small scale integration, changes were possible.

Synchronous Systems

In view of this, it was hardly surprising that manufacturers almost universally adopted synchronous operation. Here, all parts operate in lockstep to a global clock signal. This had the benefit of introducing a formal design approach to implementing control. This framework enabled novice designers to attempt designs which were likely to work. Furthermore, more effective design tools could be implemented to ensure correctness. In addition, problems of testability were greatly improved by partitioning tasks into specific time slots.

The synchronous approach has been continually refined and developed over the past two decades. Design tools have also matched this development to the point where designs can be automatically generated from a high level behavioural description. Hand-crafted manual design need only be applied to critical sections and designs tend to be as good as the design tools!

Limitations of Synchronous Design

For these reasons, nearly all digital design is synchronous (and is likely to remain so). However, it is now becoming clear that considerable power (of the order of 20 to 30 Watts) is associated with the complex, high-performance, synchronous chips which are currently emerging, and a significant percentage of this power (up to 30%) is linked to the clock. Furthermore, this cost penalty is likely to rise disproportionately in the future as progressively higher clock speeds are applied.

The use of a high-speed global clock requires that *all* parts operate at this speed. Not only does this introduce additional complexity into a design, for example pipelining, in order that units conform to the timing criteria but also requires a powerful and carefully designed clock driver to drive its large capacitive load while maintaining clock skew within acceptable bounds. Thus a significant amount of chip area is associated either directly or indirectly with the use of such a clock. Power consumption is directly related to the amount of hardware. However, the situation is made worse here by the fact that state changes are associated with the clock edge and will often occur even if a unit is idle in a particular cycle; such transitions lead to unnecessary dissipation of power.

The Move To Asynchronous Design

Large and wasteful power dissipation plus the large silicon area associated with the use of a clock are responsible for renewed interest in asynchronous design techniques. Here, units operate at their natural speed when they are free to operate; there is no global clock. This approach will result in significant simplification of the hardware to process data. This in turn will lower power requirements which has great commercial potential, particularly for portable electronic equipment which is battery powered.

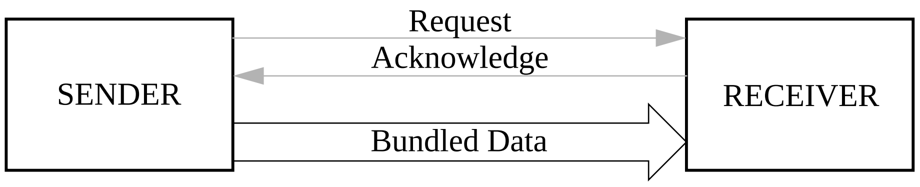
The control for an asynchronous design is by its nature more complex than that required for a synchronous approach. Furthermore, to be viable, current asynchronous designs need to adopt a more formal approach to control than was evident in the early machines. This is indeed happening although it is fair to say that no particular technique is as yet emerging as the definitive methodology for designing asynchronous control.

The main options by which asynchronous approaches can be differentiated are the choice between dual rail encoding or bundled data, plus the adoption of either transition or level sensitive signals. Dual rail encoding represents each signal by two wires, one representing the true and the other the inverse state; this allows each signal to carry both state and timing information. Bundled data, on the other hand, has one wire for each signal and (a bundle of) data is accompanied by an additional single timing wire indicating that *all* data is valid. In level sensitive logic, the two binary states are represented by a high and low voltage, whereas in transition signalling, the edges denote a change of state and the levels have no particular state attached to them.

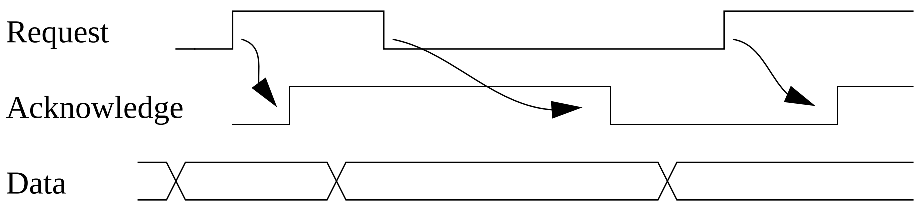
In general, transition control is more complex than level sensitive control because internal to the control logic, a positive or negative transition often needs to be converted to a level. However, level sensitive circuits need to be reset at the end of each operation before the next can begin. The additional transition is wasteful in time and energy. Dual rail encoding involves approximately twice the number of wires as a bundled data approach. However, it has the advantage that delay insensitive circuits (which operate correctly despite variable delays of any size for circuits and interconnections) can be designed. A bundled data approach tends to lead to speed insensitive circuits where the design is tolerant to arbitrary circuit delays but the interconnection delay is assumed to be bounded.

Micropipelines

In 1987, Sutherland presented an asynchronous approach referred to as Micropipelines [1] based on (level sensitive) bundled data accompanied by two transition control signals, figure 1a. A transition on the Request control line indicates to the Receiver that valid data levels are present on its input lines. This data is now held constant by the Sender until it receives a transition on the Acknowledge control line. This indicates that the Receiver has accepted the data and the Sender is now free to send another request, figure 1b.



a) Signals



b) Typical Timing

Figure 1. A Micropipeline Interface

Only about six basic logical structures are required to control the flow of data in transition signalling, including an arbiter. While these are fairly complex, they are well defined and once designed can be treated as standard library elements to form any other required control function. Because bundled data, transition control signalling appears to offer the best power, performance and area for asynchronous design, it has been selected as the approach with which to investigate asynchronous architectures.

The Asynchronous ARM

The synchronous ARM is a widely-used, relatively simple (the initial design was about 25,000 transistors), low-cost, low-power 32-bit RISC architecture [2]. It can therefore be considered a design of merit by which an asynchronous design may be measured.

In moving to an asynchronous micropipelined ARM processor, the design has been partitioned into a number of units [3]. A unit not only operates independently of all other units using the bundled data interface convention but is itself partitioned into independent, asynchronous subunits. Figure 2 gives a top-level view of the processor units.

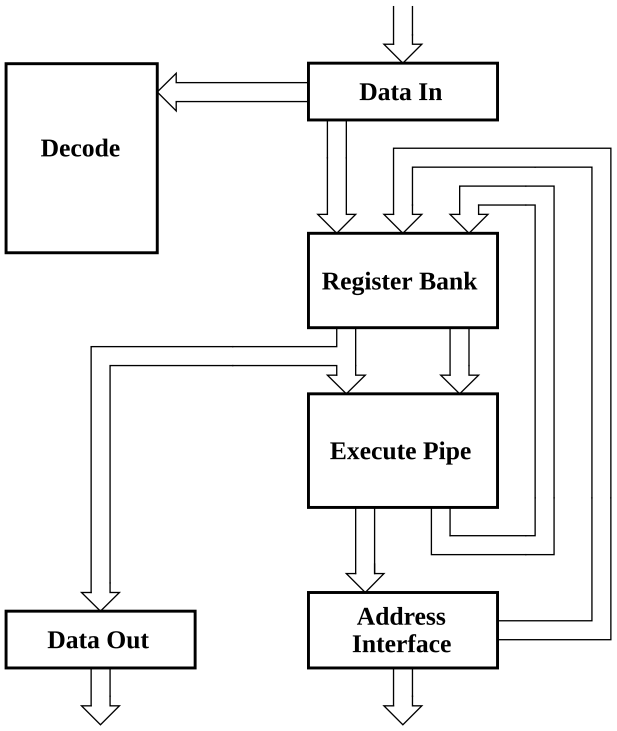


Figure 2. ARM Processor Overview

Data from memory is handled by the Data In Unit which passes instructions to the Decode Unit and operands to the Register Bank. The ARM uses a register-oriented instruction set and thus the heart of the processor is the Register Bank. Almost all instructions access at least one register (often two). Many then perform multiple operations (eg. shifting or multiplying and adding) on the source operand(s) in the Execute Pipe before writing back the result to a destination register. Data to be written to memory is sent to the Data Out Unit. Operand addresses are sent via the Address Interface which also computes instruction addresses and requests that they be prefetched; the route to the Register Bank is for new values of the Program Counter.

In order to keep units busy (and to prevent deadlocks), pipeline register stages, termed event registers, are used between and within units. These too use the bundled data interface and act as buffers so that data can be pre-loaded and queued for a unit. Where units or sub-units compete for a succeeding resource, an arbiter is used to allocate the resource. Register writeback occurs much later and asynchronously to the read-out [4]. The splitting of read and write activity in a single instruction and the need to keep the Register Bank busy to obtain good throughput means that several instructions are present within the processor, all operating concurrently but at different stages of execution. One consequence of such an asynchronous approach is that the number of prefetched and executing instructions within the processor at any time varies (although is bounded by the number of pipeline stages available).

Multi-Level Differential Logic

A high-performance design requires the support of advanced technology, and state-of-the-art high-speed differential bipolar logic developed and fabricated by GPS is to be used. This is a multi-level differential circuit with up to three levels accommodated within the 3 Volt supply. This leads to complex gate structures of up to three variables, such as shown in figure 3, which can be implemented in a single element.

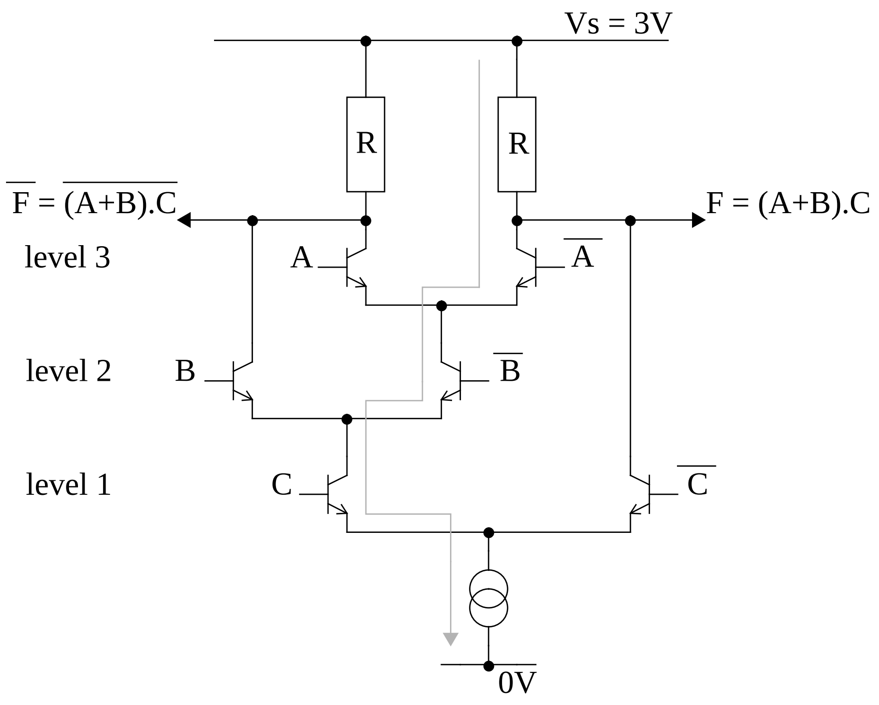


Figure 3. Example of Bipolar Multi-Level Differential Logic

Signals are represented by true and inverse phase, the difference determining the input signal state. Only *one* current conducting path is present between the true or inverse output and the current source, shown in figure 3 for C high with A and B low. This pulls the associated output low while the other output remains high, to form the differential output. In this logic, input voltages at each level are different requiring the use of (simple) translators to drop output levels to the two lower input levels. Circuit speed is primarily dependent on the current chosen with three options normally available to the designer, allowing a speed-power trade off to be made. The speed is also dependent upon the level at which inputs are applied, with inputs at lower levels having greater propagation delays due to the greater number of transistors which normally need to be switched to establish the new current path.

CMOS Versus Differential Bipolar Design

The features of the differential bipolar circuit mean that designs have not only to be efficient in the type of function implemented (which by its nature is normally at a higher level than usually found in a CMOS design), but must ensure that time critical design or the most frequent input changes occur at the highest input level (level 3). This additional ‘degree of freedom’ makes this type of bipolar design more complex than CMOS or other bipolar design.

For the new GPS process, prototype samples indicate that even at the lowest current source option of 90µAmps, the propagation delay through a simple level 3 buffer/inverter is typically 260psecs. This compares with a typical CMOS inverter delay of around 500psecs. However, whereas the loading effect on CMOS logic is very pronounced, the speed penalty from loading bipolar logic is relatively little. Furthermore, in CMOS the loading effect can introduce significant signal skew necessitating the adoption of a ‘safe timing strategy’ [5], where true and inverse control signals are independently generated to provide a non-overlap. Differential gates exhibit excellent fan out levels (at around 10 for a level 3 input and 20 for level 1 and 2 inputs), so that while buffering is still required for control signals on a 32-bit wide highway, this will be small compared with that needed for a CMOS design. Further speed is gained over a CMOS design by not having to precharge circuits, from not having to invert signals and from the very small voltage swing (160mV as opposed to 5V in CMOS).

Differential bipolar design is at a disadvantage to CMOS in terms of silicon area. This is to be expected as bipolar circuits are significantly larger than their CMOS equivalent. In addition, differential logic involves approximately twice the number of transistors and interconnections as a single-ended design. Furthermore, there are no ‘short cuts’ (such as a wired-or) or clever topological circuits (such as a barrel shifter) as are found in CMOS designs. The former is a result of using a differential design and the latter the result of there being no equivalent to the CMOS pass transistor.

Performance Potential

The factors listed above indicate that a performance improvement of at least three should be achievable over a CMOS implementation. To confirm this, a detailed design of the Arithmetic And Logic Unit (ALU) has been performed.

The ALU is the most important sub-unit in the Execute Pipe since all instructions pass through it during execution. Of these about 80% use the adder with logical operations being performed on the remaining 20% [6]. The time to perform addition is of course dependent on the time to propagate the carry. In a synchronously timed design, the ALU has to be designed so that worst case addition is completed within a timing cycle; this normally requires the incorporation of carry look-ahead logic. In asynchronous design, advantage can be taken of the data dependency to signal when the operation is complete. Here, each stage signals when its carry is valid and since this indication is given by the signal logic level, four-phase control is adopted internally. A worst-case ripple carry is rarely encountered. This leads to simpler logic overall compared with synchronous design and a performance gain for the majority of addition operations. Unfortunately, not many units can benefit from such data dependency.

The ALU has been optimally designed for the type of differential element available on the differential bipolar process and a block diagram of the data path is shown in figure 4. Inputs are preconditioned to be either zero, the operand or its inverse. Logic operations take a fixed time while addition awaits a ‘1’ level on the Valid signal from each stage. These are combined to indicate operation termination. The ALU output is then selected from either the logic or adder output.

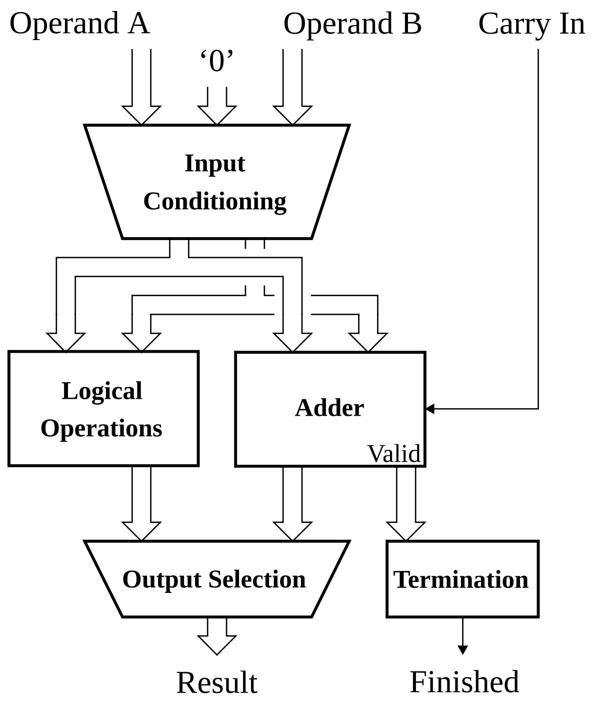


Figure 4. Data Path of the ALU

Each stage requires nine logic elements plus two (differential) level shifters. The most time-critical signals are the generation of the outgoing Carry and Valid signals from the incoming Carry and Valid Signals (if these cannot be generated directly from the operands). These are all handled at the highest input level to minimise delays. Verilog simulation at a current of 90µA per gate shows that logical operations take 4.05nsecs, while a worst case addition where the carry has to ripple from the least significant to the most significant bit will take 13.67nsecs. This includes the time for the accompanying control signals to operate.

This indicates an immediate factor of three improvement on an equivalent CMOS design. Higher current options and further design optimisation will yield further performance gains, although further design work and results from prototype samples are required to give a more accurate estimate of expected performance improvement.

Conclusions

The TAM-ARM project represents an exciting opportunity to transfer design skills for novel architectural approaches into a commercial environment. There is much work to be done to realise the potential of an asynchronous approach in terms of manufacturability, reliability, testability and availability of other compatible system components. However, a promising start has been made and results from this work are already contributing to commercial developments in the industrial partners.

References

1. I.E. Sutherland, ‘Micropipelines’, Communications of the ACM, Vol. 32, No. 6, June 1989. pp. 720-738.
2. S.B. Furber, ‘VLSI RISC Architecture and Organization’, Marcel-Dekker Inc., New York, 1989.
3. S.B. Furber, P. Day, J.D. Garside, N.C. Paver and J.V. Woods, ‘A Micropipelined ARM’, VLSI 93: International Conference on Very Large Scale Integration, Grenoble, France, September 1993, pp.5.4.1-5.4.10.
4. N. Paver, P. Day, S.B. Furber, J.D. Garside and J.V. Woods, ‘Register Locking in an Asynchronous Microprocessor’, ICCD ‘92: IEEE International Conference on Computer Design, October 1992, pp.351-355.
5. C. Mead and L. Conway, ‘Introduction to VLSI Systems’, Addison-Wesley, 1980.
6. D.V. Jaggar, ‘A Performance Study of the ACORN RISC Machine’, M.Sc. Thesis, University of Canterbury, 1990.