Power Reduction for System Technology - PREST Project

Deliverable D1.2

Amulet Group
Department of Computer Science
University of Manchester
Oxford Rd.
Manchester
M13 9PL

Power Reduction for System Technology - PREST Project

Deliverable D1.2

Report on Asynchronous Design Techniques/Arithmetic Styles

Contents

Contents

1 Introduction

1.1 Motivation for low-power design

Low-power systems are becoming more and more commonplace. They have existed for many years in products such as digital watches and calculators, but the more recent surge in the popularity of portable audio products and mobile communications equipment has vastly increased the demand for energy-efficient designs.

As ever-growing numbers of transistors are packed into these low-power systems, to feed the constantly increasing demand for greater functionality, the power consumption of these devices is growing. This trend must be countered as battery technology has not been, and in all probability will not be, able to keep up with the demands placed upon it. This has resulted, in some cases, in successive generations of a product having poorer performance than their predecessors, in terms of battery life, rather than the improvements that consumers expect.

The problem exists in non-portable equipment too. As the power dissipated by a single packaged device has grown, the technology of IC packaging has struggled to keep up. Many silicon vendors are now concerned at their potential future inability to remove heat from devices, and the consequences for circuit performance and reliability that would result from such failure.

1.2 Low-power needs of digital mobile telephone handset designs

Mobile digital telephone handsets, the field of interest in this investigation, offer an excellent forum for the investigation of power reduction techniques. In particular, handsets must have the lowest possible energy consumption when they are in stand-by mode, but be capable of enormous processing power when required. They also offer an opportunity to introduce new techniques relatively easily as the product turnover is very high and product lifetimes very short (at least for a high performance electronic product).

A further advantage is that a leap in the technology of the GSM system has introduced a new requirement. The two recent standards for encoding the voice channel in a GSM system, Enhanced Full Rate (EFR) and Half Rate (HR), both require significantly more computational power than the currently predominant Full Rate (FR). An FR voice codec encodes the 8-bit, 8kHz sampled voice signal into a 9.6kbit/s stream, an HR codec does the same job but onto a stream only half as wide, at 4.8kbit/s, and an EFR codec uses the 9.6kbit/s channel but gives a better quality voice signal. Most handset manufacturers are in the process of introducing EFR capable handsets, but HR handsets are not widely available to consumers yet. Handsets with HR capability require a new generation of low-power DSPs giving ever-more performance whilst not permitting their energy requirements to rise.

The major computational task of the DSP in a GSM handset is that of compression of the speech signal from 64kbit/s to 9.6kbit/s (for FR and EFR) and to 4.8kbit/s (for HR). It must also manage the channel equalisation and echo cancelling and other tasks. The workload is therefore highly variable, as the major task, the coding and decoding of the voice channel, is only performed during voice calls. The DSP is required in some capacity however for almost all parts of the GSM system. Any improvement in the energy efficiency of the DSP will therefore have a marked impact on that of the handset as a whole.

Over recent years, stand-by times for GSM phones with a given capacity of battery have soared, while call times have stayed relatively static. This is due to a much greater improvement in the energy efficiency of the handsets in stand-by mode compared to in active mode. Any techniques to reduce the active power consumption of the DSP are particularly to be welcomed.

1.3 Structure of this document

Section 2 of this document discusses how and why power is dissipated in CMOS circuits, and outlines some general techniques for power reduction, including circuit techniques, clocking techniques and technology techniques. Section 3 gives an overview of some asynchronous design styles and discusses some of their advantages and disadvantages. Section 4 describes some of the arithmetic styles that have been considered and considers their applicability for low-energy computation. Finally, Section 5 gives our conclusions on the approach we should take to the design of a low-power DSP design.

2 Circuit design for power reduction

There are many methods for reducing the power consumption of a circuit, but it is important to remember that we are seeking *low-energy* computation, as well as low-power, i.e. a reduced amount of energy to perform a particular task. This is a requirement that is distinct from low-power as the power required for a task can be reduced by simply doing the task more slowly. The power issue is important from a packaging and integration standpoint, but battery life is not improved by low-power operation unless it is also low-energy.

2.1 Sources of energy dissipation in CMOS circuits

Energy dissipation in CMOS circuits can be split into three categories, each effectively independent of the other.

2.1.1 Leakage

Leakage currents arise due to the incomplete turn-off of transistors in their subthreshold region (i.e. where the gate-source voltage, V_{gs}, is less than the threshold voltage, V_t), and reverse-biassed diode leakages (for example between drain and bulk).

These currents are currently very small and, in most cases, negligible. As balances of V_t and the supply voltages change these are likely to increase, and will, in future become much more significant, perhaps even dominant in applications where a circuit is in an idle state for a large proportion of its time.

This idleness in itself is likely to increase as, with increasing numbers of transistors available on a die, the proportion of them that are active at any one time will decrease, increasing the mean proportion of time that a module is idle.

2.1.2 Short-circuit current

Short circuit currents arise due to the imperfect switching characteristics of MOS transistors, and typically make up about 10-20% of the total dynamic current. In a stacked pair of CMOS transistors, ideally one should turn on at exactly the same moment that the other turns off. However, in practice, there is a short period of time when they are both switched on, and a current is able to flow from V_{DD} to GND.

For a simple inverter configuration, this occurs during the time that $V_{tn}<V_{in}<(V_{DD}-|V_{tp}|)$ is true, where V_{tn} and V_{tp} are the NMOS and PMOS threshold voltages. Because of this gap where short circuit currents occur, it is important to minimise as much as possible the time the input voltage spends in this conducting zone.

2.1.3 Switching energy

Switching energy is the largest component of power consumption, and the one that is easiest to alter by means of appropriate design. All nodes in a circuit have capacitance, and each time a node changes its logical state, this capacitance must be either charged or discharged, dissipating energy. Each time the capacitance is charged or discharged the energy dissipated will be

$$e_{transition} = \frac{1}{2}C_{node}V_{DD}^2 \tag{1}$$

If the node has a mean frequency of switching of a_{node} then the mean power dissipated at that node will be:

$$P_{node(switching)} = \frac{1}{2}a_{node}C_{node}V_{DD}^2 \tag{2}$$

and the power dissipated in the whole circuit will be:

$$P_{switching} = \frac{1}{2}V_{DD}^2\sum a_{node}C_{node} \tag{3}$$

From this it can be seen that power can be reduced by reducing the switched capacitance, the frequency with which that capacitance is switched, or the voltage through which it is switched. These factors are discussed in Section 2.2 and Section 2.3.

2.2 Circuit-level energy reduction techniques

Circuit level energy reduction techniques can be broadly split into those that are concerned with reduction in the switched capacitance and those that are concerned with the output voltage swing.

2.2.1 Circuit topology

The manner in which a given circuit functionality is implemented can have a significant effect on power consumption. In symmetric gates such as a simple 3 input NAND gate (illustrated in Figure 1) the order in which the inputs arrive affects the power consumed.

Assuming all inputs are initially low. If inputs A and B arrive first (in either order), then because there is still a path to V_{DD} through the C-input PMOS transistor, the parasitics of all three NMOS transistors will be charged. However, if input B and C arrive first (again in any order), their parasitics cannot be charged as there is no path to V_{DD}, the A-input NMOS transistor being off. When the A input then transitions from $0 \rightarrow 1$, and the output thus $1 \rightarrow 0$, then less charge on the parasitics must be discharged to ground. The rule of thumb is that the input closest to the output of the gate should always be the last to switch. Obviously this rule only works where one input is consistently later changing than others, and where circuit function allows. This effect, because of the reduced charging/discharging, also speeds up switching of the output.

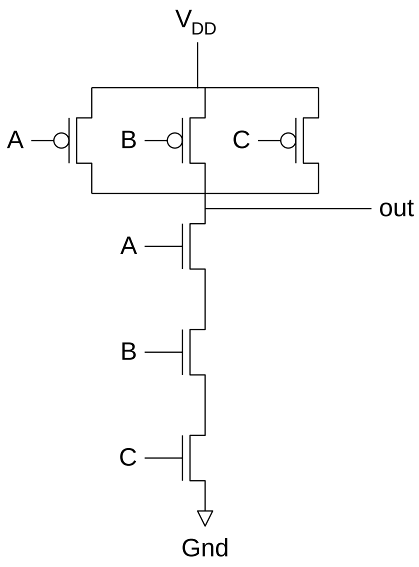


Figure 1: Implementation of 3-input NAND gate

A similar effect is also present in AND-OR-INVERT gates, and others with a similar topology. Considering the AOI gate shown in Figure 2, giving the function (AB+C). There will be an internal node capacitance in the PMOS stack at the point that is common to all three PMOS transistors. In Figure 2(a), if the output is high, the internal node will be charged, and when the output goes low, this node will want to discharge, which it can do through either the A or B PMOS, if they are on. In Figure 2(b), when the output goes low, the node could only discharge through C if it is on. This means that the node is twice as likely to discharge in the first case than the second if the probability of activity on each input is the same.

2.2.2 Reduction of power supply voltage

Reduction of the power supply voltage to a circuit appears the most attractive method of saving power as it has a quadratic effect on the power. However it also causes a linear slowdown in circuit *performance*, as the reduced voltage also reduces the current available for charging the capacitances, which remain unchanged, thereby increasing the charging time.

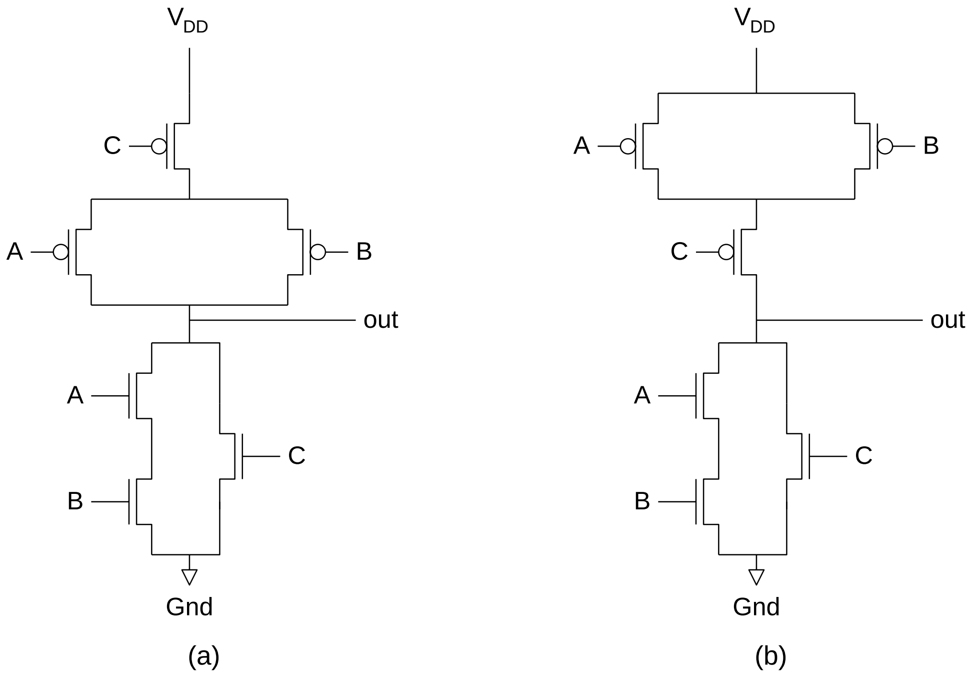


Figure 2: Alternative implementations of AND-OR-INVERT function

Schemes have been proposed that include multiple supply voltages, variable supplies and multiple variable supplies, often combined with critical-path analysis to ensure that maximum performance is available on critical paths, while minimum energy consumption is available on non-critical sections.

2.2.2.1 Multiple V_{DD}

Multiple V_{DD} schemes are a relatively simple way of overcoming the performance penalty associated with power supply reduction techniques. These simply involve using a different, closer pair of power supply rails for some sections of the circuit, usually those not on, or far from the critical path. There are two principal schemes, asymmetric and symmetric.

The asymmetric scheme, proposed by Toshiba, is based on the introduction of only a new, lower V_{DD}, with the lower rail, V_{SS}, remaining the same as for the normal parts. A single library of cells is characterised at two supply voltages and thus the choice of supply for a block may be left till late in the design stage. This has the advantage of simplicity, but requires conversion circuits between sections with different V_{DD}s. These conversion circuits, if used too frequently, remove the benefit from having the lower V_{DD}, and may also adversely affect performance. This technique may produce savings of between 30% and 50%.

The symmetric scheme employs both a new, lower V_{DD} and a new, higher V_{SS}[30]. This is slightly more complicated than the asymmetric scheme due to the need for an additional, internally generated supply voltage, but has the advantage that, as long as the difference between the inner and outer supply rails is less than V_t, no conversion circuits are necessary, so the modules may be mixed at will. Care must still be taken to ensure that the higher low and lower high logic levels of the modules with the closer supply rails do not cause excess leakage currents in modules with wider supply rails which they drive, due to incomplete turn-off of the transistors (see Section 2.1.1). This technique can produce savings of up to 50% compared to standard power schemes.

2.2.2.2 Variable V_{DD}

This technique involves altering the V_{DD} that is used by the entire device so that performance matches the current need. It can be controlled either by a signal from outside the circuit, or by software/firmware being run inside the circuit, but the V_{DD} is generated internally. All parts of the circuit have their V_{DD} scaled in the same way, so the critical path will not alter, merely become slower. This system causes problems with synchronous circuits as the clock must also be suitable scaled.

2.2.2.3 Multiple variable V_{DD}

This technique is a combination of the variable and multiple schemes, and involves having a scalable supply for some parts of the circuit, and a fixed one for other parts.

2.3 Activity reduction

Activity reduction may be broadly split into two parts. One, timing strategy, is covered in Section 2.5, the other, glitch reduction, is considered here.

There are two types of activity to consider, the first is that caused by glitches, and the second is that which is intentional and part of correct circuit operation. Both can, in particular circumstances, be reduced to a large degree. Glitches are made up of hazard transitions, glitches due to unstable, intermediate output states and those due to uncoordinated input changes.

In a datapath, glitches can have a particularly deleterious effect on energy efficiency as the number of nodes switched due to them may be very large. A common technique to reduce the effects of glitches is to introduce latches in the datapath. This can vastly reduce the glitch activity, but at a cost of performance and complexity of correctly generating the latch control signals. This in turn can be alleviated by pipelining the datapath, but this carries an increased latency cost. Another technique is to balance the delays through the datapath so that any module which sees two inputs will only experience changes to these inputs simultaneously. This technique while not only being extremely difficult to achieve in practice, also has a drawback in that it can be difficult to ensure that the power consumption is actually reduced, as the delay balancing structures will also consume power.

An illustration of the scale of the potential saving due to datapath glitch reduction was given in [15]. When simultaneous inputs were presented to the ALU of an ARM6, the power consumed by the ALU was reduced by 35% when compared to skewed inputs. If a latch was added after the ALU so that it only drove the output bus when data were ready, there was a 20% power saving.

2.4 Technology techniques

A method that is not particularly open to the designers of individual circuit components is that of using aspects of the fabrication technology to reduce power consumption. These techniques are mainly useful for reducing power not associated with switching, so are not particularly widely used yet, but with changing technology, and the resultant decrease in the proportion of power that is switching-related, they will probably achieve far wider use.

2.4.1 Varying V_t

The level at which V_t is set can have a marked effect on both the leakage and short-circuit currents. If the V_t is raised, then a given input level will mean that an off-state transistor lies further into the subthreshold region, reducing the leakage current in an exponential manner.

A raised V_t will also reduce short-circuit currents as there is a smaller window of input voltage that can cause both NMOS and PMOS transistors in a stack to be on. If V_t is raised to more than half of (V_{DD}-V_{SS}), then this window disappears entirely, and there are no short-circuit currents, although the output node is, for a brief period, undriven, so could go into an undefined state.

The drawback of a higher V_t is a reduced performance: as transistors are switched on (and off) later in an input transition, output nodes do not begin to be charged or discharged until longer after the input transition begins, meaning that the better power characteristics are offset by poorer speed.

2.4.1.1 Multiple V_t

A common way of using the benefit of high V_t without suffering the performance penalty is to have multiple V_t values. This technique is used in a similar manner to multiple supply voltages: high V_t transistors are used in non critical-path parts of the design[28]. This keeps the benefit of the lower leakage currents, but does not sacrifice much performance due to the non-critical nature of the affected transistors.

Another technique which is also a circuit technique and is usually used in conjunction with clock gating is to place a high-V_t transistor above the stack and control this with a separate, ‘off’ signal[27], as illustrated in Figure 3. Effectively this transistor acts as a current supply for the channels of the stack transistors, and when the module is not in use, the high-V_t transistor is turned off and its low leakage means low leakage in the stack.

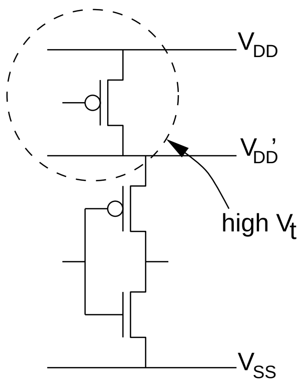


Figure 3: High V_t transistor to reduce idle state leakage currents

2.4.1.2 Variable V_{subs}

Under normal circumstances, the substrates of transistors are connected to the same place, and hence voltage level, as their source terminal. If we permit the substrate to be connected to a level *outside* the supply rail, the *effective* V_t is changed, and its value is given by:

$$V_{t(eff)} = V_{t(0)} + \gamma V_{subs}^{1/2} \tag{4}$$

where γ is a constant dependent on transistor parameters, lying normally between 0.3 and 0.7, with 0.5 usually being used in calculations.

Therefore if, when multiple V_{DD}s are used (see Section 2.2.2), the substrate is left connected to the widest supply rails, this has no effect on modules that use the widest supplies (as would be used in the critical path), but further reduces the leakage currents in those with a reduced V_{DD}[29]. This is illustrated in Figure 4.

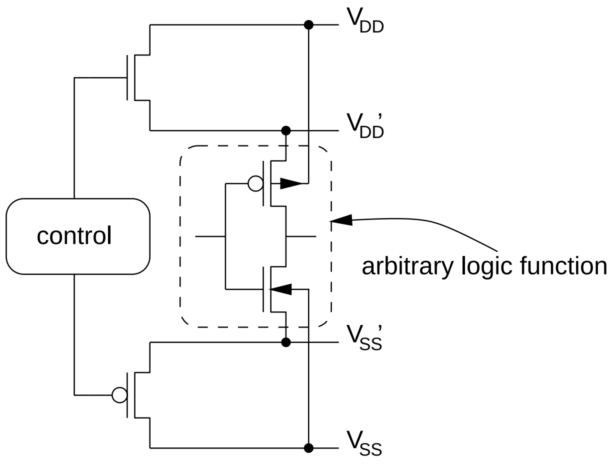


Figure 4: Dynamic variance of V_{subs}

2.4.2 Silicon-on-insulator fabrication

Silicon-on-insulator (SOI) technology is a fundamental change to the fabrication process[16]. Instead of the circuit being fabricated on a mass of silicon, there is a slim layer of silicon on top on a layer of SiO_2 insulator.

The presence of this insulator vastly reduces leakage currents, in practice to almost zero, meaning V_t (and hence V_{DD}) can be reduced to levels that are, in bulk CMOS processes, impractical, without incurring the leakage penalty.

Another, more important, advantage of SOI technology is that node capacitances are radically lower than in bulk CMOS technology, leading to reduced switching energy.

A final advantage of SOI that will, as more SOI processes become available, become more important is that there are fewer processing stages in SOI fabrication compared to bulk silicon.

2.5 Timing strategy

Because, in a synchronous design, the system clock reaches every part of the circuit, and operates continuously, it is continuously dissipating, and causing to be dissipated, large amounts of power. Altering a clocking strategy is potentially a very effective way of reducing power consumption.

2.5.1 Clock gating

Clock gating is a technique to control the distribution of the global clock and to drive it only to parts of a circuit that are in use, stopping distribution to portions of the circuit that are not currently doing anything useful. This can save an enormous amount of power as merely driving the clock line is often a very costly process. In addition, any datapath that is clocked may well consume significant amounts of power.

Clock gating works, conceptually, by simply feeding the clock signal through an AND gate with another signal that controls its distribution. All parts of the circuit that are fed from below the output of that gate are controlled, or ‘gated’ by that signal. In practice, the method of gating is likely to be much more complex than that.

There are several problems with this system, primary amongst them is that of generating the control signals. There is also the problem of the clock phase in which the wake-up occurs.

2.5.2 Multiple clock zones

The use of multiple clock zones enables a clock signal of a lower frequency than the circuit needs to be distributed around a circuit, and the full speed is regenerated locally using a phase-locked loop (PLL).

This mechanism has the benefit of removing the need to distribute a global, high-speed signal to every part of the device, but has the disadvantage that the PLL-based regeneration of the clock is both complex and fairly slow to restart after it has been halted, as the PLL system must be allowed to stabilise.

2.5.3 Asynchronous systems

Self-timed circuits have a number of key advantages over clocked circuits when design for low power is being considered. The defining feature of an asynchronous circuit is the absence of a global clock signal which must be distributed throughout the circuit. This clock signal causes unwanted switching activity at every node to which the clock is connected[1].

In high-speed synchronous circuits, the task of preventing clock skew when distributing the high frequency clock across the circuit requires considerable additional circuit overhead in terms of clock buffers and phase-locked loops and much skill, and effort from the designer. This is also a task which cannot be attempted until the design is almost complete as it will depend critically on placement and routing of the final design.

Most importantly, the need to force the clock signal around the circuit can cost an inordinately large amount of power - it is reported that in recent DEC Alpha devices 30% or more of the total power consumption (which is considerable in any case) is caused by the clock distribution system.

In an asynchronous system these problems are all absent. In asynchronous systems all timing considerations are localised and based on events generated by the circuits themselves. This means that such circuits are inherently modular: they create their own timing system. They are interconnected with handshake circuits and can operate at their *natural* rate. This makes design very modular and removes the need to design for worst-case scenarios.

Modules only dissipate power when they are doing something useful, and when there is nothing useful to do their activity ceases. This is obviously valuable where the workload of a module or the whole circuit is variable.

2.6 Conclusions

Many of the techniques mentioned in the previous sections may be unavailable to a designer simply on the basis of the amount of *extra* design work required to use them in large circuits.

In general, specific processes, such as dual V_t and SOI processes will not be available, and in any case, they would not materially affect the actual design of a circuit other than at the layout level, so from a circuit design point of view they may be considered a separate issue.

Reduction of the supply voltage can be a good source of power reduction, but only with an evolving technology landscape. As process sizes shrink, and as a consequence become faster, and have lower V_t, the performance loss due to the reduced supply is compensated for by the inherent speed increase. This method cannot, however, be used in practice without the process evolution. For a given design therefore, standard supplies are probably required.

Multiple supplies provide a possible compromise, as the performance loss is marginal provided the critical path elements are fully identified.

Circuit topology issues are a cost-free measure that can be implemented regardless of the use or absence of any other measures.

As may have been predicted, the most practical and probably the most significant way to reduce power consumption is by reducing the activity level of the circuit. There are two principal ways to achieve this: splitting the circuit with latches or blocking gates and using an asynchronous timing scheme. Blocking elements can be used at whatever scale is appropriate, including at the highest, system level.

In applications where there is a large variability in the processing load, asynchronous circuits offer an obvious and simple manner to reduce mean intentional activity, while at the same time easing the problem of removal of unintentional activity (glitches). Asynchronous design styles are explored in the next section.

3 Asynchronous design styles

Asynchronous circuits inherently cease their switching activity when no work needs to be done, and can go from idle (zero power) to full activity (maximum throughput) instantaneously. This means that where there is a significant time period in which the circuit must be able to react instantly to its environment, but is not actually doing anything, asynchronous techniques are a good candidate for power reduction.

This instant switch from idleness to maximum throughput is a part of the excellent implicit power management properties that asynchronous circuits have. As a module will cease activity any time it is not needed, for no design effort or performance overhead we have fine grained power management both in time and functional terms.

A further advantage of asynchronous circuits comes from their self-timed nature. In clocked systems, it is necessary to have the entire circuit operating at a speed governed by the slowest single circuit element, as all data transfers must be synchronised with the global clock. This means that circuits in the critical path often require considerable design effort and extra complexity to ensure that worst-case data can be dealt with in a particular time period and this may lead to the designer being forced to use a design which has poor power properties in order to shoehorn the operation into its allocated time slot.

Asynchronous systems manage the transfer of data at a local level, and operate in their *natural* time. This means that circuits can be designed to maintain a high throughput for typical data, and the pathological cases are simply given longer to complete. Because there is no need to force a circuit to complete in a given time, if required it may be modified to be more energy, if less time, efficient or vice versa, depending on the design requirements. This locality also reduces the number of long-distance nets in a circuit (most notably the clock), and thus the number of very high capacitance nodes that must be switched.

A factor that is being ever more widely recognised is the inherent portability of self-timed circuits. As they have a well defined interface that is not constrained in its timing requirements, parts of an asynchronous circuit may be easily reused in new designs, and in designs to be fabricated in a different technology. With the growth in the size of devices, such hardware reuse will become more important.

The final advantage of asynchronous systems, and one that is of crucial importance in the mobile telecommunication arena, is that as all parts of the circuit are operating at different speeds and times, rather than switching simultaneously in response to the global clock signal, the electromagnetic emissions are both more dispersed, and lower in total. Where there are distinct frequency peaks and harmonics in synchronous systems, these are much reduced in asynchronous systems. Where large subsystems run flat-out, such as a fully loaded execution pipeline, there are still detectable characteristic lumps in the spectrum, but these are many dBs lower than the levels needed to meet EMC requirements.

Asynchronous design has not been popular for several decades (certainly not in the CMOS era), but is currently receiving a much higher level of attention. Unfortunately, because of this past neglect, there are almost no CAD tools for asynchronous design, and none integrated into the commercial VLSI CAD tool suites available. There are a few tools however, and in several of the design techniques described below, conventional synchronous tools may be used for much of the design work.

In almost all cases, for the lower levels of design, layout and place-and-route, there is in practice no difference between synchronous and asynchronous circuits. It is in the higher levels, architectural design and synthesis, that the lack of tools is most obvious. This too is changing: LARD[24] is proving extremely useful for architectural experimentation in the Amulet3[25] project, as has the verification and synthesis tool Petrify[26].

There are many different asynchronous design styles that may be classified most simply by their timing assumption. These are discussed in Sections 3.1 - 3.4. This document could not possibly provide an exhaustive comment on asynchronous design styles, so if the reader desires additional information on the various asynchronous design methodologies and their pros and cons, [6] provides a good starting point.

Another important difference in asynchronous design is the manner in which data are communicated around a circuit. This aspect is discussed in Section 3.5 and Section 3.6

3.1 Huffman circuits

The timing assumption of Huffman circuits is essentially that of synchronous circuits: that the delays associated with all elements of the circuit and their interconnections are known, or at least that their upper bounds are known.

3.1.1 Fundamental mode Huffman circuits

In fundamental mode circuits, the design method is usually to synthesise the circuit, either automatically or manually from a flow table[7] which indicates state stability and transitions, and the outputs generated by such transitions. This is synthesised by methods similar to that used for state machines, with state reduction and state encoding, via a Karnaugh map.

It is assumed that an input change will cause the circuit to pass through a (possibly zero) number of unstable states and into a new stable state, within a known, finite time. In order to ensure operation free of output hazards, redundant cubes (from the *Karnaugh map* of the function) must be implemented. Unfortunately, this redundancy cannot ensure correct operation if more than one input is allowed to change simultaneously, so inputs may only change one at a time, and the circuit must be allowed to find its new, stable state before another input may change.

An important factor when considering implementation in this design style is that, being implemented typically as a sum of products form, the required size of AND and OR gates increases as the number of inputs to the circuit. In most technologies large fan ins are either unavailable or penalised by a large delay. It may therefore be necessary to decompose complex nets into simpler ones. It can be shown[8] that many transformations (such as rearrangement under associativity, distributivity and DeMorgan’s laws) do not introduce new hazards, although care must be taken not to readmit hazards by accidentally removing the redundant cubes that were originally introduced to eliminate hazards.

When designing sequential Huffman circuits, we must resort to methods that are effectively synchronous. The state is stored by signals fed back to the inputs via delay elements to ensure that they do not change at the circuit inputs until the circuit has settled into its new stable state. Because the constraint of only allowing a single input to change must still be upheld, only one state bit may be allowed to change at any one time (because state bits are inputs to the combinational circuit), and this may not be at the same time as any external inputs. This causes state encoding to become a complex task, which is often simplified by using one-hot (or other types of redundant) encoding on the state signals. A further restriction on sequential Huffman circuits is that because the feedback signals may change many times after the input change has occurred, it may be difficult to calculate the time that must be allowed for the circuit to settle.

3.1.2 Non-fundamental mode Huffman circuits

The fundamental mode, although rendering the design task relatively easy as it can be almost entirely automated by tools, does cause greatly extended cycle times. The single input change restriction can however be slackened a little, as it may, in practice allow multiple inputs to change, as long as the start and end input states are both covered by the *same* cube in a Karnaugh map. When this is permitted the circuits are said to be operating in non-fundamental mode.

3.1.3 Burst mode circuits

Burst mode circuits[20][21] are not strictly Huffman circuits, and operate in a significantly different manner to them, but they have certain conceptual similarities, so they are grouped with them. The assumption they use is that in any given state, only certain combinations of inputs (known as bursts) may change. The circuit only changes state when all the inputs in a burst have arrived, but the inputs themselves may arrive in any order.

During a state change some of the output variables may change (an output burst), and the circuit moves into a state where it awaits a new input burst. New inputs are allowed only when the new state has settled.

This mode effectively transfers the Huffman requirement of only a single input changing, to only the inputs in a single burst changing. An additional requirement however is that no state may have a valid input burst that is a subset of another valid input burst, in order that there can be no ambiguity as to whether a state change should occur.

An illustration of a burst mode circuit is given in Figure 5. The timing generation circuit encapsulates all of the *burst* nature of the circuit, and only generates a latch enable pulse when a complete input burst has been received. This activates the latches which then pass the new output state as an output burst. The two latch phases must be present to ensure that the changing state bits caused by the phase 1 latches opening are not propagated round to the inputs, introducing a race condition.

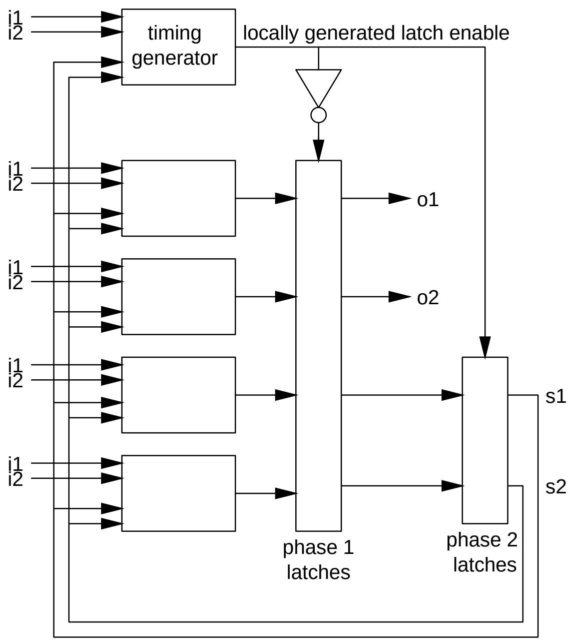


Figure 5: An example of a burst mode circuit

The claimed advantage of burst mode circuits is that by the introduction of the locally generated clock signal, the slowness that dogs fundamental mode circuits is avoided, standard state-machine techniques may be used in the design and many hazards are avoided as the state bits are not on an uninterrupted feedback path. However, outputs are generated directly from the inputs, so hazards must be explicitly avoided on those signals by the inclusion of the redundant cubes that are necessary for fundamental mode circuits.

One problem remains though, which is that as more than one state bit may change when the latches are enabled, there can be a hazard in the clock generation logic. This can be overcome, at a cost, by using a more complex clock generation technique.

A large, 300k transistor device[31] has been designed using this technique, principally with performance rather than power in mind.

3.1.4 Problems with Huffman circuits

The Huffman approach presents designers of large asynchronous systems with a number of problems. Firstly, since only one input (or in the case of burst mode circuits, only the members of a small group of inputs) may change at a time, they are very difficult to use as datapath elements, where many inputs commonly change simultaneously.

In addition the design methodologies for these circuits do not cope well with the task of combining smaller circuits, as in a hierarchical design, which it is common for designers to wish to do, and is in practice essential for large designs. A further problem is the need to add delays, and cater for worst case performance which can result in speed problems.

The final problem is that, having added redundant cubes to avoid hazards, we have removed the possibility of testing by detection of visible faults, as this fundamentally conflicts with the presence of redundant terms.

3.2 Delay insensitive circuits

The timing model for delay insensitive (DI) circuits is that both gates and wires have unbounded delays. Where a Huffman circuit assumes that after a given amount of time the circuits will have resolved its state, and a new input vector can be applied, these circuits cannot be assumed to have completed their transition at any time. As a result these circuits must explicitly indicate their completion.

The timing assumptions applicable to DI circuits force circuits to be designed in a manner that results in it being possible to mathematically prove or disprove their correctness without needing real delay data. This in turn means that performance of small parts of the circuit may be improved and then the new design used as a slot-in replacement for the old, without the need to again check the correctness of the circuit.

3.2.1 Synthesis of DI circuits

There are several methods available to synthesise DI circuits.

3.2.1.1 I-nets

I-Nets[10] are similar to Petri-Nets[14], and are transformed, via interface state graphs and state assignment processes to Karnaugh maps. There are two methods to synthesise these Karnaugh maps into real circuits.

In the first method the Karnaugh maps are synthesised into structures similar to those used for fundamental mode Huffman circuits (see Section 3.1.1). Now however the timing assumptions that were present there, that only one input may change in any particular time period, are not the same, and dynamic hazards may be introduced. No static hazards are introduced as the redundant cubes are retained.

The dynamic hazards are removed by the insertion of *inertial-delay elements*, which *swallow* a glitch up to a certain size.

The second method[10][22], known as Q-Modules, uses a structure similar to that of burst mode circuits (see Section 3.1.3), but instead of latching only when inputs arrive, it is constantly latching inputs and outputs with locally generated clocks, allowing enough time between clock edges for the combinational logic inside to settle (see Figure 6). This requires latches with synchronisers on their inputs, able to reliably latch asynchronous signals. As a synchroniser itself requires an unbounded amount of time to achieve a stable output, it must inform its clock control circuit when it has done so. Thus what is essentially a synchronous state machine has reliable DI operation.

Both these methods have flaws, the first because it tends to add a large amount of delay, and the second for the complexity of the latches and their control. Simple combinational circuits however tend to have small delays in the first style, so small circuits are more suited to this, and because the cost of the latches is fixed with respect to complexity and proportional to input numbers, the second style better suits complex circuits.

Once small modules have been implemented in one of these two styles, larger circuits constructed using them are much simpler to synthesise. However since predesigned blocks must be used, and these may not be altered without a great deal of care, there are some restrictions placed on the designers freedom.

3.2.1.2 Trace based circuits

Trace based implementation (so called because it is based on *trace theory*)[11][12], unifies the description of a circuit in terms of modules, and of those modules themselves. The circuit to be synthesised is specified in a special textual language that operates at a very low level.

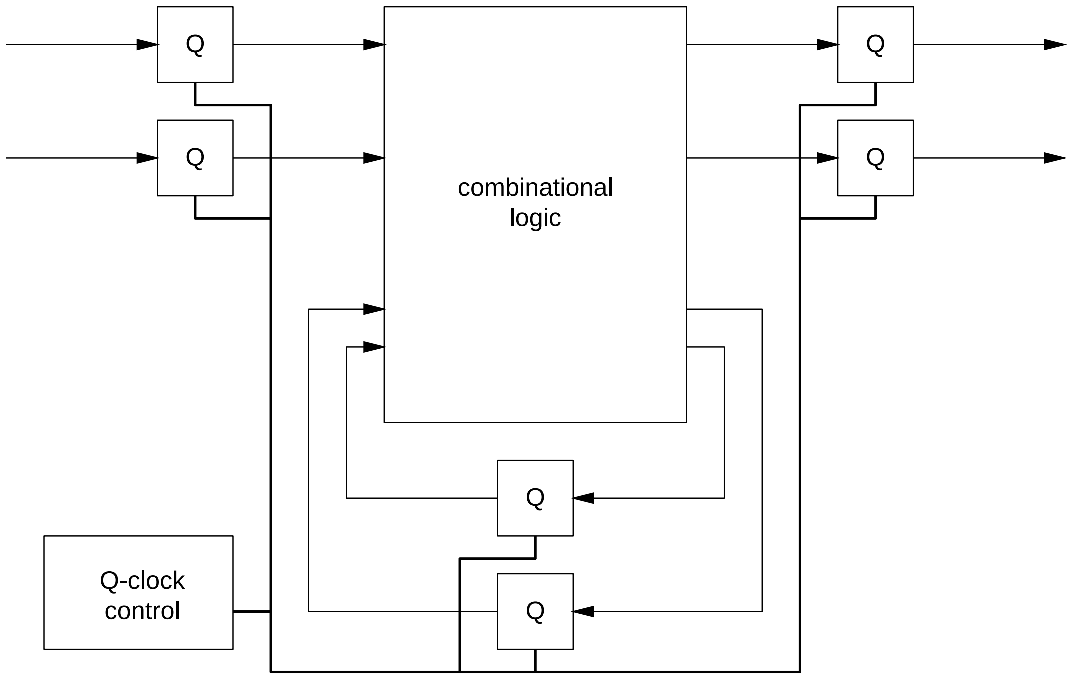


Figure 6: Q-Module delay insensitive circuit

The synthesis of this system relies on an almost direct translation between the input format and real modules. The more complex structures that are commonly found in asynchronous specifications are all describable in the input format of this system, hence the unification of the module description with the circuit description.

The input language is, however, difficult for a designer to understand and visualise, and the requirement that each individual signal transition be described is onerous for large circuits.

3.2.2 Advantages and disadvantages of DI circuits

The use of DI techniques, even when implemented using structures very similar to those used for Huffman circuits appears to provide a considerable performance benefit, and this is due to the fact that DI circuits, because of their timing assumption, must indicate completion. This results in nearer to average-case performance and better flow control than is possible with Huffman circuits. In other words, constraining the designer to build well-specified, correct circuits (i.e. good asynchronous circuits) has a performance advantage.

DI circuits are very difficult to build and have a tendency to have poor area characteristics, and as a result may also have very poor power characteristics due to the large number of transistors for a given function. The major problem with DI circuits is that their complexity increases much faster than their functionality as circuit size grows. This makes their synthesis difficult and their use largely impractical for larger circuits.

As far as the authors are aware, no large circuit has been designed using DI techniques.

3.3 Speed independent and quasi-delay insensitive circuits

Speed independent (SI) circuits use the delay model similar to that of DI circuits, but they assume that wiring delays, instead of being unbounded, are negligible. This assumption is perfectly valid where there is no question of connections being non-local, such as occurs within a small module or between physically adjacent modules.

Quasi-delay insensitive (QDI) circuits use a model somewhere in between SI and DI. Here, the wire delays are assumed to be unbounded, as in DI, but also that they obey the principle of the *isochronic fork*[13]. The isochronic fork principle is that while wire delays may be unbounded, the differences in time of receipt by multiple recipients are negligible.

The isochronic fork assumption means that as the wire delay is the same for all driven points on the wire, the delay can be seen as a single delay before the wire splits, followed by delay-free wires to the driven points. This delay can then, conceptually, be merged into the driving output leaving a negligible wire delay, which is the SI delay model. Thus these two models are, in practice, the same.

3.3.1 Synthesis of SI circuits

3.3.1.1 Signal transition graphs

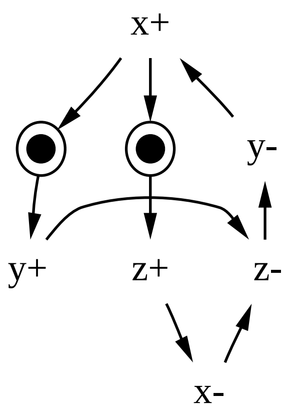
Signal transition graphs (STGs) are a means of specifying and synthesising asynchronous circuits that is currently very popular, for several reasons. The first reason is that, being a Petri-Net based method, they can be manipulated by well understood means and with mature tools. The second, more important reason is that their specification bears significant similarities to the way designers think about asynchronous circuits.

While STGs are a sub class of Petri-Nets, there are some important restrictions that must be obeyed: they may only be marked graph, free choice (or input choice) and non-input choice Petri-Nets[14].

An example of an STG and its specification is shown in Figure 7. The code shown in Figure 7(a) specifies that the STG has two inputs, x and y and one output, z. The circles containing filled blobs are known in Petri-Net parlance as places, and are only included in the diagram as a means of showing the initial state or *marking* of the STG. A place with exactly on input and one output will in practice behave as if it were not there, immediately passing any token from its input to its output.

```
.model example.g
.inputs x y
.outputs z
.graph
x+ y+ z+
y+ z-
z+ x-
x- z-
z- y-
y- x+
.marking {<x+,z+> <x+,y+>}
.end
```

(a)



(b)

Figure 7: STG and its specification

The *efficient* synthesis of circuits from STGs may require the STGs to have certain properties, for example: liveness, safeness, persistency, consistency of state assignment and uniqueness of state assignment[14][17]. The only one of these that must be satisfied for an implementation to be *possible* is that of consistency of state assignment.

These restrictions allow the use of synthesis techniques that are much more efficient than those that must be employed to synthesise I-Nets (see Section 3.2.1.1), as they avoid the problem of state-explosion.

3.3.1.2 Change diagrams

Change diagrams (CDs) are another Petri-Net-like format[18]. They are similar to STGs, but have additional types of arcs available, and relax some of the constraints on STGs that are required for efficient synthesis. There are two new arc types, weak arcs and disengagable arcs.

Weak arcs act as, in effect, an OR function in the graph. A place with weak arc inputs can fire when any of its inputs (if it has weak inputs, it may *only* have weak inputs) has a token, but when it does so it leaves a *debt* on the others which means that the next time they receive a token, it is swallowed by the debt, rather than causing a firing of the place.

Disengagable arcs are arcs that are disregarded after the first time they are fired. In other words, once they are fired, they are pruned from the graph. These arcs can therefore be used to set up initial conditions.

Disengagable arcs offer a great improvement over STGs, as initialisation when using STGs is problematic, but the relaxation of the constraints is also important. The requirement that all places to be potentially able to fire infinitely often (liveness) is substituted by the requirement only that places be able to fire at least once. A modification of the persistency requirement is also present.

Although CDs have useful features not present in STGs, they also have some crucial failings, such as being unable to model an XOR function! They are also unable to express input choice. A final problem is that there are currently no freely available tools for synthesising from them.

3.3.1.3 Communicating process compilation

Martin's communicating process compilation (CPC)[13] is a method for translating programs written in a form similar to communicating sequential processes (CSPs) into asynchronous circuits.

This method uses low-level (although more abstract than trace commands) source language constructs to describe the circuit according to the communications in it. These communications are decomposed into simpler structures until the lowest level of hierarchy is reached. These lowest-level structures are then decomposed into the four-phase handshaking protocols that they are equivalent to. These handshaking structures are then fed to production rules which produce a corresponding real circuit implementation.

3.3.1.4 Tangram and Balsa

Tangram[19] and Balsa[23] are both very similar to CPC, and even more similar to one another. They offer the ability to translate a textual description of a circuit, described by its *communication* rather than the transitions on its wires. The languages are efficient in their description of a circuit, and unlike CPC-based synthesis they may produce either DI or SI circuits. The ability to produce either DI or SI circuits is due to the use of an intermediate representation, which may be mapped onto either DI or SI backends. These two methods are included here rather than in the DI section as the backends that are most developed in both cases are the SI ones.

Tangram began as a research tool from Philips which was made available to the Amulet group during a collaborative project. After the project finished the company decided that it is a commercially useful tool and it is no longer available to those outside Philips. Balsa began as a reimplementation of Tangram, in response to Philips' decision, but has since moved beyond this and significantly expanded its feature set. Balsa is currently being used in the design of the Amulet3[25] microprocessor, and will soon be publicly available.

3.3.1.5 SI datapath

Synthesis of SI circuits is pointless without the ability to perform processing on the data. Differential cascode voltage switch logic (DCVSL) can be used for this. DCVSL is a dynamic logic style whose charging and discharging is controlled by four-phase `request` and `acknowledge` signals rather than a clock signal (see Figure 8). Blocks implemented in this style are interconnected in complex fashions using other, non-datapath SI circuits, but DCVSL blocks may be simply cascaded together with no glue logic where this achieves the desired functionality.

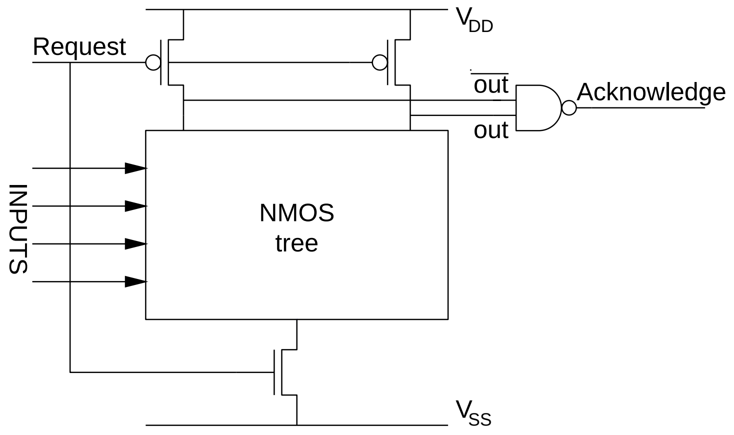


Figure 8: DCVSL logic

Methods exist to automatically generate optimal DCVSL implementations of datapath elements[7]. These blocks are implemented as a precharge/discharge circuit around an NMOS tree structure that has complementary outputs. These complementary outputs can be used to generate a completion signal, and because of their complementary nature, the completion signal cannot be generated before the block has completed its evaluation, giving to the block its SI characteristic.

Tangram and Balsa mentioned above can also be used for the synthesis of datapath, but they perform this task much less efficiently than the synthesis of control circuits. Balsa however can be instructed to use prebuilt elements supplied by the designer in the synthesis of datapath and this makes datapath synthesis much more efficient.

3.3.2 Advantages and disadvantages of SI and Q-DI circuits

SI circuits have an important advantage over DI circuits in that more complex basic blocks can be used with (multiply-)forked outputs, as the isochronic fork assumption makes a larger range of basic blocks into hazard-free structures.

The isochronic fork assumption is, however, sometimes difficult to satisfy. Wire lengths can differ, implementations of modules can differ, thresholds can differ, etc. In custom devices, most of these problems can be overcome by various means, but in an FPGA implementation for example, where placement and routing may be far from the designers control and wiring delays may dominate gate delays, this would be impossible.

Where datapath is implemented as DCVSL modules, there can be a significant power penalty as there may be an expansive NMOS tree that is discharged every time the block's output is evaluated, regardless of its previous state and its inputs. In some cases there can be a power benefit due to lower input capacitances on some types of logic function.

Tangram has been used very successfully by Philips, and two commercial products are available that have been designed using it. The first, an error corrector for use in a DCC system[32] was targeted at a low-power application, and achieved an order of magnitude power reduction due to the infrequent, bursty nature of the demand. The second, a microcontroller for a pager[33], was targeted at EMC concerns, and was the first microcontroller that could be run during receipt of a paging message. This allowed, for the first time, the paging protocol to be in software and therefore allowing the pager to operate on all standards without design modification.

3.4 Micropipelines

This category is in a section of its own as it combines the timing assumptions and design techniques of other methods. Sutherland's micropipelines[5] are a combination of bounded delay techniques, used in the datapath, and SI techniques, used in the control path.

The timing assumption used here is that for a small, isolated section of datapath, the delays are known, but that section can operate in an environment of unbounded delays. This is achieved by separating such sections of datapath with latches controlled by a network of handshaking components that control the datapath and perform neighbour synchronisation.

The timing constraints for this style are met by either having SI circuits in the datapath or conventional circuits and matched delays. Where SI circuits are used (the assumption of no net delays is reasonably robust here because of the locality of reference) they explicitly indicate the timing. Where matched delays are used, the matched delay elements perform the *bounding* of the delays. The control network then uses the timing information provided by the datapath elements to ensure correct operation.

3.4.1 Basic operation of a micropipeline

A micropipeline is like any other pipeline in that it has latch separated sections, however because of the asynchronous nature of the system, the latches must indicate when they have latched a data value. The simplest example of a micropipeline is a FIFO, as illustrated in Figure 9.

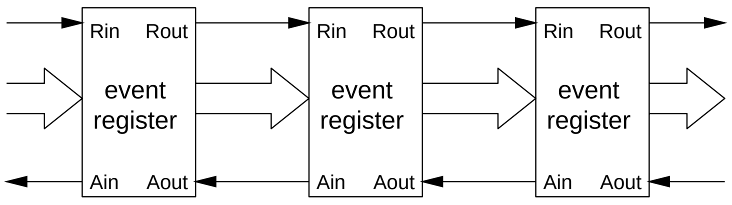


Figure 9: Micropipeline FIFO

The event registers latch a datum in response to a `request` event on Rin, but only if the next register downstream has indicated that it has itself latched in the last datum it was passed. The downstream register does this by putting an event on its Ain (connected to the Aout of the upstream latch). When the latching has taken place, the register tries to pass the datum onto its downstream neighbour by placing an event on Rout (its neighbour's Rin), and acknowledging to its predecessor that it has latched by an event on Ain.

The event register itself is illustrated in Figure 10. This shows how the various handshaking lines interact with the latch itself. Assuming that the next stage has latched the last datum passed to it, then the arrival of an event on the Rin signal will cause the *capture* port to receive an event which latches information into the latch. At some point later, when this latching is complete, this is indicated to both the upstream and downstream latches (via the Ain and Rout signals respectively) by an event on the *capture done* port. At some later point, when the downstream latch has latched the datum passed to it, it will send an event on the Aout signal to the *pass* port which will, some time after that, arrive at the *pass done* port, allowing the cycle to begin again.

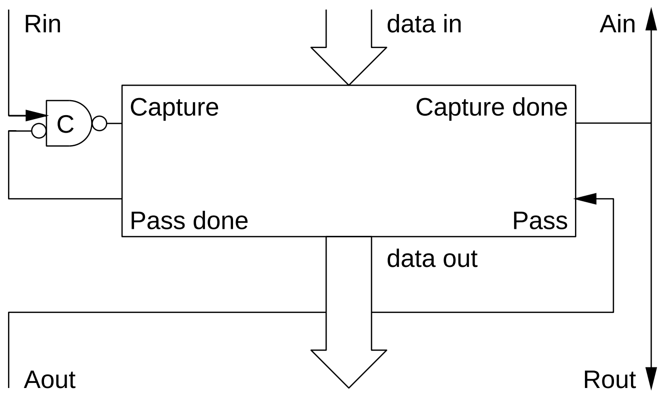


Figure 10: Micropipeline event register

If another event arrives on the Rin wire before the *pass done* event has arrived, then the C-element prevents it from being passed to the *capture* port until an event arrives at *pass done*. This prevents an upstream latch overwriting downstream information.

When processing is added to the pipeline, this is just placed in the data stream between event registers, as illustrated in Figure 11. Now however, the data will not make the journey between the output of one register and the input of the next in the near-instantaneous fashion it did before, so the `request` signal must be delayed by an amount greater then the delay through the processing logic.

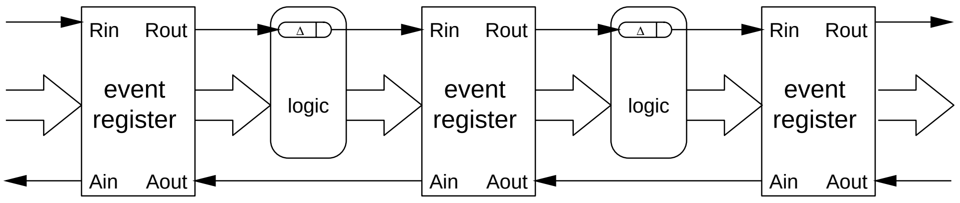


Figure 11: Micropipeline processing

3.4.2 Effect of latch control style on power consumption

A micropipeline design contains a large number of stages, therefore the control mechanism of the registers can have a great effect on the power consumption. If the registers are used in a normally open configuration, glitches can propagate along the pipeline until they arrive at a full stage. This means that if the pipeline experiences low occupancy, glitches may be particularly costly. If the latches are used in normally-closed mode, there is a speed penalty, as they must be opened to allow a datum to pass, but they are more energetically efficient, as glitches only propagate to the next latch.

A third type of latch controller has been proposed[9] that allows the latch mode to be dynamically switched between normally open and normally closed. This switching allows the latches to be operated in normally open mode when speed is desired, and in normally closed mode when energy efficiency is a greater priority. A serendipitous aspect of this switchability is that commonly when high performance is required, the pipeline will experience high occupancy, which will cause glitch propagation to be reduced in any case.

3.4.3 Advantages and disadvantages of micropipelines

Micropipelines have an inherent advantage over other self-timed styles which is that datapath elements may be easily designed, in a conventional style, by conventional means, and with conventional tools. Using matched delays it is possible (if not necessarily always useful) to simply take a block of conventional, synchronous style logic and wrap it in a micropipeline shell, add a matched delay, and it becomes an asynchronous component. This facility of module design means that it is very easy to design small blocks and in turn to use these blocks to build very large devices.

It is not always advantageous to use a synchronous block drawn from a library, as there is usually a much better way to implement the function when the designer is freed from the timing constraints of the synchronous world. For example, a synchronous design of adder must wait for the longest possible carry chain, even though this case will be extremely rare. In an asynchronous adder, the length of the carry chain may be determined to know the correct delay to associate with the operation, or a cessation of activity may be detected in the adder network to know when addition has finished. Both of these methods may have a much shorter delay under normal circumstances and a longer delay in the worst-case situation, but as the worst case is so infrequent, average performance is increased compared to using a synchronous design with a matched delay[37].

Using matched delays compromises some of the average case performance that can be expected of micropipeline systems as *some* parts of *some* circuits must, in *some circumstances* exhibit worst case delays. The occurrence of worst case delays is, however, not seen in all parts of a circuit at once or on every occasion that a section of the circuit is used.

The disadvantage is that the networks of handshaking components were in the past difficult to design, although much less so than is commonly thought. Furthermore, tools for the design and simulation of these networks are evolving rapidly. As these handshaking networks are implemented as speed independent networks, many of the tools used for their design are applicable in this case. Micropipelines is the approach that has been used for the design of the Amulet1 and Amulet2 processors, and is being used for the design of the Amulet3, which have been or are being designed here at Manchester University. Because of the lack of existing tools, tools have been crafted in-house: LARD was created to fill the need for an asynchronous architectural simulator, and BALSA to fill the need for an asynchronous synthesiser.

3.5 Data encoding

Because of their timing assumptions, different communication methods must be used in the different design styles described. In a synchronous or Huffman circuit, a wire can be assumed to have reached its final value at a given time, but when information about time is removed, a problem arises. A receiver cannot know if a wire’s value has stayed at, say, 0 because that is its correct value or because a transition to 1 has not yet arrived.

The receiver will, however, receive the transition at *some* time, and this can be used as a basis for the signalling. A sender will indicate the validity of data on one signal and the receiver will indicate the completion of processing of the data via another signal. This can be done either by dual rail coding, or by the bundled data protocol.

Dual rail coding involves encoding the signal on two wires, and which of the two wires undergoes a transition indicates the bit's value. This method is, effectively, one-hot coding for the two states of each bit. An example of this is shown in Figure 8, where the *out* and $\overline{out}$ signals are a dual rail coded version of the `out` value.

The bundled data protocol encodes the data on a conventional 1-bit per wire scheme, but uses an explicit request line to initiate. It is assumed (or ensured) that the delay on the explicit request line is greater than that on the data lines. In both cases an explicit acknowledgement line is needed. It can be seen from this that it is not possible in these types of circuits to encode a bit with a single wire.

If a single bit is to be transmitted, transition signalling requires three wires but retains true delay insensitivity while bundled data also requires three wires, but loses the strict DI characteristics (although it does remain SI). For large data widths however, transition signalling becomes impractical, as not only are there a large number of wires, but the arrival on all of them must be ascertained

3.6 2-phase versus 4-phase

These initiation and completion signals can operate with one of two protocols, either two-phase or four-phase.

In the two-phase protocol, in which an event occurs on a wire when there is *any* transition: a signal that changes 0→1→0 has had two transactions on the wire, and no differentiation is made between a rising edge and a falling edge. In Figure 12, two complete transaction cycles are shown, illustrating the fact that the sense of a change in `request` or `acknowledge` is ignored. The data must be valid before the sender initiates the `request` transition, and must remain valid until after the receiver indicates completion of receipt by sending a transition on the `acknowledge`.

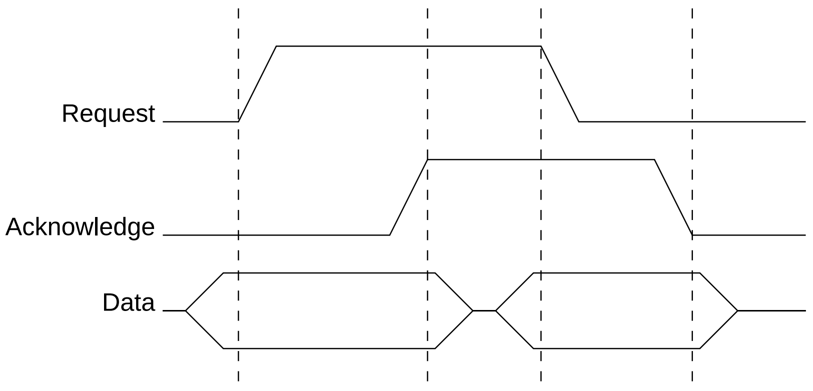


Figure 12: The 2-phase protocol

The four-phase protocol has a return-to-zero phase for each signal so the same sequence, 0➞1➞0, comprises only one transaction on a four-phase protocol signal. In Figure 13, we see the four possible 4-phase protocols that may be used to indicate data validity.

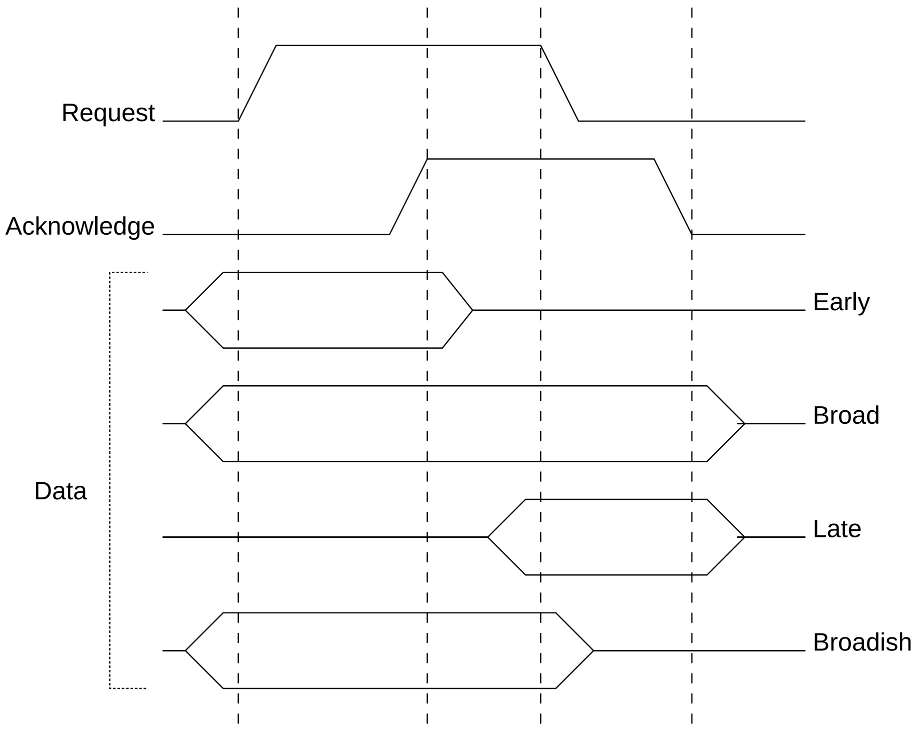


Figure 13: The 4-phase protocol

The broad protocol has the advantage that it simplifies datapath design as the data are valid as long as `request` is high, meaning that it can be used as a conventional enable signal. The broadish protocol also has this advantage, but the receiving latch must treat the protocol as early. The late protocol is, in practice, of no use, so can be ignored.

The advantage of four-phase circuits is that, having the return-to-zero phase, circuit elements may be level, rather than edge, sensitive, which simplifies them. Two-phase circuits, on the other hand have, at least in theory, a speed and power advantage due to having half the number of transitions in this control path. The advantage is not as great as might at first be thought, as the majority of the delay is usually in the datapath logic, which must be included whichever protocol is used, and 2-phase control elements are also generally slower than 4-phase ones.

3.7 Conclusions

The major factor in picking an asynchronous design style to use is that of the tools available to the designer. There are not currently any asynchronous tools available in EDA suites, although it would require relatively little effort to integrate, for example, Petrify as a synthesis tool synthesising STGs into netlists. This lack of tool support is the major problem which prevents greater adoption of asynchronous design.

Of the many asynchronous styles examined, only a few are practical for large-scale designs. Of these, only micropipelines seem to offer a practical engineering approach to an architecture with a large datapath component, where performance is also an issue. SI circuits are also useful, and will become more so as the sophistication of Tangram/Balsa increases.

Micropipelines however provide, for synchronous designers, an easy entry point into the asynchronous world. Large parts of the design can be done using only existing tools, and it requires the easiest conceptual leap from the synchronous domain. Those parts of the circuit that do require specifically asynchronous tools can, in general, be designed with freely available (and cost free) tools.

All micropipeline designs to date have had performance as their principal design objective, but future designs, where they are used in applications that have a highly variable workload, will exhibit excellent power characteristics also. The mantra *'asynchronous circuits are excellent at doing nothing'* must be kept in mind.

An illustration of this is the Amulet2 microprocessor. At peak performance, it can manage a computational performance approximately the same as the ARM6 microprocessor it was intended to be comparable with, consuming approximately the same amount of power. When idle however, it had a power consumption several orders of magnitude less than the ARM6.

4 Choice of Number Representation and Arithmetic Style for DSP

4.1 Introduction

The key feature which distinguishes a digital signal processor from a general purpose microprocessor is that a DSP tends to spend much of its time performing a repetitive function on a continuous stream of data. Typical algorithms performed by a DSP [34][35] are shown in Table 1

FIR Filtering	$y_n = \sum_{i=0}^{p-1} b_n x_{n-i}$	IIR Filtering	$y_n = \sum_{i=0}^{p-1} b_n x_{n-i} + \sum_{i=0}^{q-1} c_n y_{n-i}$
Convolution	$z_n = \sum_{k=-\infty}^{\infty} x_n y_{n-k}$	Fourier Transform	$X_k = \sum_{n=0}^{N-1} x_n e^{-j2\pi k\frac{n}{N}}$

TABLE 1. Typical DSP algorithms

More complex algorithms, such as data compression, encryption, statistical modelling etc. can often be represented by combinations of these or similar functions. As can be seen, the majority of these operations are formed from the summation of a series of multiplications. The basic operation in digital signal processing can therefore be thought of as being the multiply-accumulate (MAC) operation. It is usual to have a dedicated unit for performing this operation, with data and address calculations being performed by different functional units (as opposed to a general purpose microprocessor where data and address calculations are often performed by the same datapath). It is also common to have separate instruction and data buses (Harvard architecture) in order to maximise the throughput of data to and from the MAC unit. A typical DSP architecture, for the DSP56000 series of processors [34], is shown in Figure 14. This uses a dual Harvard architecture, an architecture common in modern signal processors, with a bus for program instructions and a separate bus for each operand of the multiply-accumulate

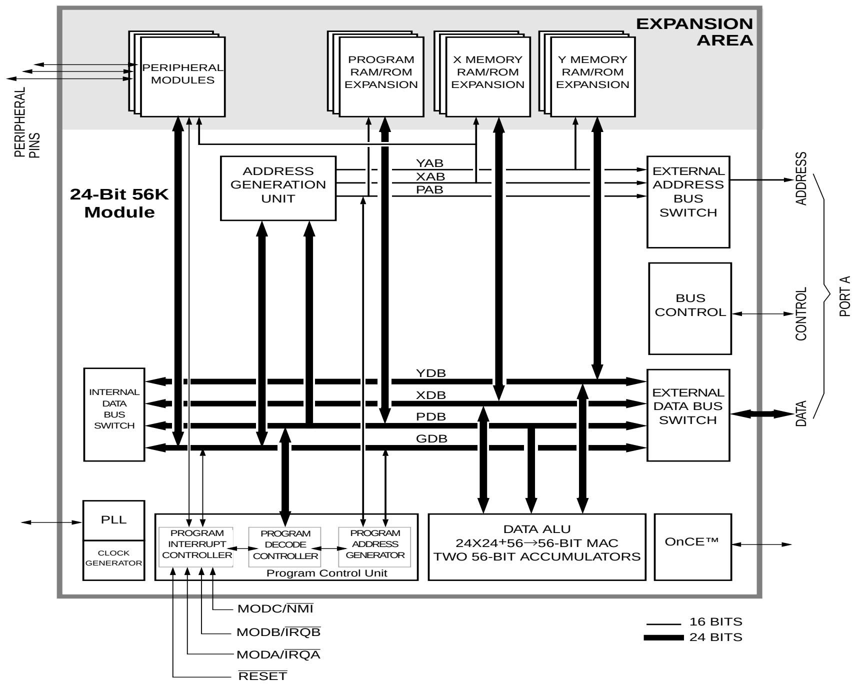


Figure 14: DSP56000 Block Diagram

4.2 Power Consumption in Digital Signal Processors

The fundamental operation of a digital signal processor can be seen as streaming data from memory, performing calculations (principally multiply-accumulate operations) on that data, and returning it to memory. The power consumption can therefore be broken down quite naturally into three component areas:

1. Moving data to and from memory.
2. Calculations performed on data in the MAC and ALU.
3. Instruction fetch and the control and address calculation units.

The cost of moving data to and from memory is often the dominant component of system power consumption [36]. This can be reduced to some extent by appropriately sized and located cache memories at the system design level. However, the minimum memory traffic that can be achieved is fixed by the algorithm being performed.

A DSP will usually be required to perform multiply-accumulate instructions a quickly as possible, within a single clock cycle in a synchronous system. To meet this requirement, it is necessary to use high speed array multipliers and fast adders. As the majority of DSP operations are likely to be multiply-accumulate operations, it is clear that the design of the arithmetic unit will have a large impact on the overall power consumption of the DSP core.

4.3 Choice of Number Representation

One of the first decisions to be made in the design of a digital signal processor system is how the numbers are to be represented. This impacts upon the power consumption of the system in a number of different ways. Firstly, the choice of number representation has an impact on the complexity of the arithmetic elements required to maintain a certain level of throughput. Secondly, the type of number representation used has an impact on the switching activity both on buses and within processing elements. Finally, the compactness of the encoding has an impact on the amount of memory required for storage and hence on the amount of power consumed in transferring data to and from memory. Only fixed-point number representations will be considered here, as floating-point system are generally more complex and hence will not be chosen for a low-power system. However, many of the points considered apply to the design of elements of a floating-point system as well.

4.3.1 2s Complement Representation

The number system most commonly employed for general purpose fixed-point digital signal processing is the *2s complement* numbering scheme. This representation has the form:

$$Z = -b_m 2^m + \sum_{i=0}^{m-1} b_i 2^i \qquad -2^m \le Z < 2^m - 1 \tag{5}$$

Its primary drawback is the large numbers of redundant ones required to represent small negative numbers. This means that for a digitised signal with small fluctuations about zero, there will be high switching activity in the sign extension bits.

4.3.2 Sign and Magnitude Representation

An alternative representation, that avoids the switching of the sign extension bits, is the *sign-magnitude* numbering scheme. This representation has the form:

$$Z = (-1)^{b_m}\left(\sum_{i=0}^{m-1} b_i 2^i\right) \qquad -2^m + 1 \le Z < 2^m - 1 \tag{6}$$

The drawback of this numbering system is that, in order to add numbers with different signs, it is necessary to either convert both numbers to 2s complement format and use a conventional adder, or use a dedicated subtracter circuit. If the former option is chosen, the extra transitions generated will reduce the benefit of switching to sign-magnitude representation. The latter option has a penalty in area. Chandrakasan and Brodersen [2] concluded that sign-magnitude representation is best used in designs where a large capacitive load is being driven such as external memory buses, etc. In this case the power overhead of converting to and from 2s complement representation is negligible compared to the power saving from the reduced switching activity on the bus. They also proposed a structure for reducing switching activity in sign-magnitude accumulators, where large numbers of accumulate operations are performed before the final value is read from the accumulators. Two separate accumulators are used, with positive values being fed to one accumulator and negative values being fed to another. The negative value is then subtracted from the positive value every N cycles (where N is large) to obtain the total accumulated result. This technique need not be restricted to sign-magnitude systems however: a similar benefit would be obtained in 2s complement systems as it would prevent the sign extension bits from switching frequently. However, it was noted that the extra control circuitry needed for this technique can increase the overall power consumption when the input signal does not exhibit rapid changes in sign. Also the extra circuitry required may be prohibitive in area for systems with a long accumuland word length.

4.3.3 Variable Bus Width

An alternative method for reducing the switching activity due to sign-extension in 2s complement signals was proposed by Nielsen and Sparsø[4]. The application in question, a FIR filter bank for a hearing aid, exhibited low-level input signals for long periods of time. Power was reduced by dividing the datapath into two eight-bit segments and only enabling the least-significant eight bits during periods of low input magnitude. This also reduced power consumption by allowing the multiplier and adder circuitry to be partially deactivated.

4.3.4 Delta Encoding

A method used for reducing the number of bits transmitted along a bus is the technique known as *delta encoding*. In this scheme, a (large) value is only transmitted once, and then only the changes to that value are passed. This method is often used as a means of reducing the amount of information transmitted along a communications channel with limited bandwidth, but it does not seem particularly attractive from a low-power viewpoint. At the receiving end it is necessary to perform an addition or subtraction for every value received, and in order to perform comparisons it is necessary to add the original value to the accumulated value.

4.3.5 N-of-M Encoding

Another method used to reduce the number of transitions on a bus is N-hot or N-of-M encoding. In this scheme, a value is represented by a high value on N lines out of M. This gives the possibility of encoding $M!/(M-N)!$ values. This can be very efficient for small values of M, but to represent large numbers requires M to become prohibitively large. Circuits to perform arithmetic operations in anything other than the simplest coding schemes such as one-hot coding become too complex, and so it would be necessary to convert to 2s complement form. This is also the case where the values need to be stored in memory. The case where most benefit could be obtained from the potential reduction in switching activity is where signals must be driven off-chip. Unfortunately, this is the case where N-of-M encoding is least suitable due to the restricted number of pins available.

4.3.6 Addition Considerations

The greatest challenge in addition for 2s complement and sign magnitude binary numbers is the fact that the value of the most significant bits of the result potentially depend on the least significant bits of the input. The worst case delay for a simple ripple-carry adder is proportional to the number of bits in the circuit. This performance is generally too slow for use with digital signal processing, although the typical delay is somewhat less than this, and so an asynchronous circuit can take advantage of completion detection to give better average performance[37]. To provide better performance, a number of circuits have been developed to reduce the addition time. The basis for these designs is still either calculating carry-generate and carry-propagate signals[38][39][40][41] or by conditional-sum type adders where the sum is split into blocks for which results are calculated in parallel for both the cases of carry and no carry into the block[42][43]. This adds considerably to the power consumption of the circuit. A novel carry encoding scheme[44] has been developed within the Amulet group (and has been patented) which gives a very high performance and makes the group adders redundant. An implementation in a 0.35 micron triple metal process gave a worst-case delay of about 1.8ns for a 32 bit adder. Its simplicity and high speed means that this style of adder would be extremely suited to a high speed digital signal processor.

4.3.7 Residue Number Systems

Using different number representations can eliminate the problem of carry propagation. One such class of representation are *residue number systems*[45]. In these systems, an integer Z is represented by its remainder modulo a number of different bases, which are selected so as to be relatively prime to one another. For example, the number fifteen would be represented (using bases 3, 5 and 7) as (0,0,1) while eight would be represented as (2,3,1). The properties of number represented in this manner are such that addition, subtraction or multiplication can be performed simply by applying the operation to each component and taking the result modulo the base value. For example, adding fifteen and eight would give the result (2,3,2). The expected result, twenty-three, is indeed represented by (2,3,2).

The greatest problem with residue number systems is that it is impossible to directly compare two numbers, or to tell whether a number is positive or negative. This also implies that division cannot be directly be performed on numbers in residue format. Methods for division have been developed[46], but they involve approximation and conversion to an intermediate mixed-radix number system. Conversion of numbers from binary to residue representation requires division operations, which are non-trivial[47], and conversion from residue to binary representation generally requires some kind of ROM look-up table. These difficulties mean that residue number systems are not very well suited to use in a general purpose digital signal processor. They do offer considerable possibilities for special purpose circuits however, particularly where no zero point or comparison is needed. One such example is in a frequency synthesis circuit which uses one-hot coding of the residues to give a circuit which has a very low power-delay product[48]. A residue number system was appropriate here because the basic function of the circuit is to add a variable phase increment onto an accumulator at each phase, and then look up the appropriate sample from a wavetable at each clock cycle. No comparisons were necessary at any point in the algorithm.

4.3.8 Redundant Signed-Digit Representation

An alternative class of number systems that can eliminate long carry chains and which also do not require sign-extension bits are redundant signed digit representations as proposed by Avizienis[57]. Redundant number systems are defined by the following equation:

$$Z = \sum_{i=0}^{N} z_i r^i \qquad -r+1 \le z_i \le r-1 \tag{7}$$

The restrictions on the values for z_i are based on the requirement that there be a uniquely defined zero value. For radices greater than two, it is possible to add two numbers together so that the output from a given pair of digits is dependent only on their values and the value of a transfer digit from the next lower order digit. Other redundant number systems such as carry-save or borrow-save can be shown to be special cases of this type of number system[57]. Another attractive aspect of signed-digit representations is that, due to the different possible representations of any value, it is possible to choose the representation with the minimum Hamming weight[57] (i.e. the representation with the greatest number of zero digits).

In order to reduce the amount of redundancy, it is common to use a *modified signed-digit* representation which is of radix two and where each digit is taken from the set of values (-1,0,1). This allows more compact encoding, at the expense of an increase in the possible carry propagation under addition to two places. This numbering system can be encoded onto two wires, with one wire indicating a positive and the other a negative value, which means that a number can be negated simply by reversing the wires. The sign of a number is slightly more difficult to determine than for a 2s complement or sign-magnitude representation, as it is necessary to examine all of the digits and find out the sign of the most significant place. An example of the addition:

(10-11)+(0-101)=7+(-3)=4

is demonstrated in Figure 15.

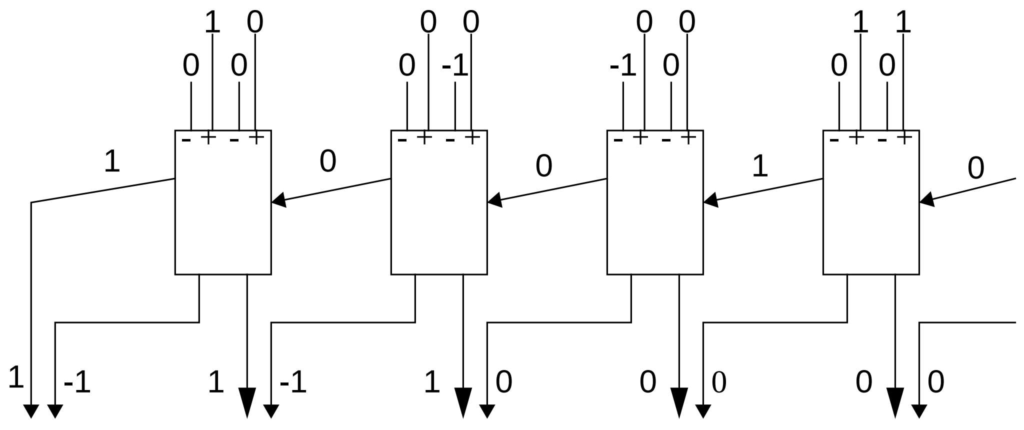


Figure 15: Addition in Modified Signed-Digit Representation

Conversion from sign-magnitude representation to this modified signed-digit representation is trivial: either the positive or negative lines are set the same as the input binary number. For 2s complement, it is necessary to set the most significant bit negative, and then set the remaining bits positive. For example, -7 is represented in 4-bit binary as 1001 which would become (-1 0 0 1) in modified signed-digit form. Conversion back from the modified signed-digit representation requires that the value on the negative lines be subtracted from the positive lines.

4.3.9 Multiplication Strategies

Multiplication can be thought of as a succession of shifts and additions, and it is common to use redundant number representations in order to avoid carries and reduce the number of operations required. Booth encoding[49] reduces by half the number of partial products that must be summed, by representing the multiplier as a radix-4 signed digit number as follows:

$$Z = -2^{n-1}a_n + \sum_{i=0}^{n-2} 2^i a_i = \sum_{i=0}^{n/2} 2^{2i}(a_{2i-1} + a_{2i} - 2a_{2i+1}) = \sum_{i=0}^{n/2} 2^{2i} k_i \qquad (8)$$

where k_i is taken from the set of values (-2, -1, 0, 1, 2). The coefficients for the partial products are made by shifting and/or inverting the multiplicand as required. To avoid carry propagation, it is common to use a redundant representation when summing the partial products. Wallace[50] proposed the use of full adders to reduce three rows of partial products into two outputs at each stage. A full adder can be thought of as a circuit which counts the number of high inputs and outputs the number in binary on the carry and sum outputs. A tree of full adders will give a 3:2 compression of partial products at each stage. The final partial sum and partial carry must then be resolved by a standard adder. A typical structure is shown in Figure 16.

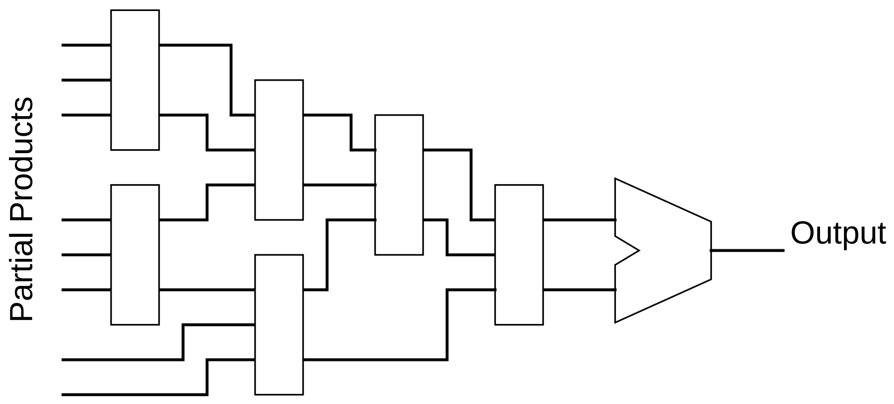


Figure 16: 8-bit Wallace Tree Multiplier Structure

Due to the increasing significance of interconnection delays in sub-micron VLSI technologies, it has been suggested that higher order compressors such as 4:2, 6:2, 9:2 or 27:5 compressors are preferred, as they allow a more regular layout[51], and reduce switching activity when compared to a Wallace tree made of full adders[51]. A 4:2 compression tree is shown in Figure 17.

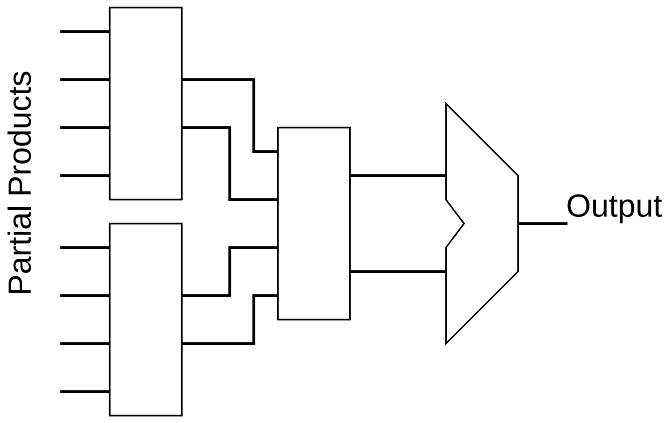


Figure 17: 8-bit 4:2 Compressor Tree Multiplier Structure

Analysis and simulation of a number of different multiplier architectures[52] found that glitches and wire delays mean that simple fast architectures such as Wallace tree or 4:2 compressor array multipliers are often lower power than slower architectures such as a linear array. In addition, there is a potential race condition between the multiplier and multiplicand in Booth encoding which can cause increased power consumption through glitching. Fried has proposed a Booth encoding scheme[54] which is race-free, at the expense of larger area for the partial product generators. Another undesirable aspect of Booth coding is the need to sign-extend the negative partial products, which increases both the switching activity and the number of compressors required to sum the partial products. This effect can be greatly reduced by using a modified sign-generate algorithm[44][51][53] which adds an adjustment value to generate the negative partial product.

An alternative structure for the compression tree is to use modified redundant signed-digit encoding[54][55]. Signed-digit 4:2 adders are used at each stage, which allows an efficient binary tree structure to be used, and also reduces the number of transitions required in generating the partial products using Booth's algorithm. To generate negative values for the partial products, the multiplicand is simply routed onto the negative input of the signed-digit adders, rather than requiring inversion or the use of the modified sign-generation algorithm. Where a 2s complement or sign-magnitude result is required, it is necessary to use a subtracter as the final stage of the multiplier. This type of structure is therefore particularly well suited to a system that uses sign-magnitude representation throughout. A 2s-complement representation would require that negative inputs be inverted and that the negative output be inverted before being sent to an adder circuit. A sign-magnitude system necessarily has circuitry for performing subtraction.

4.4 Investigation

In order to investigate the effects of different number representations, a simulated 64 tap FIR low pass filter operation was performed on a 5.4 second excerpt of sampled speech (“*Oh no, not cheese... can't stand the stuff. Not even Wensleydale?*”). The model used for simulation was based on the data ALU of the Motorola DSP56000 series, as shown in Figure 18. Transitions at each bit position, and total transitions, were measured for both of the multiplier inputs, the multiplier output, and the accumulator output. The number systems used were 2s complement, sign-magnitude and modified signed-digit representation. The speech data and the FIR filter coefficients both had a precision of 16 bits in 2s complement representation. The adding scheme used in the modified signed-digit model was based on that used by Takagi et al.[55].

Position	**2s Complement**	**Sign-Magnitude**	**Modified Signed-digit**
Data input	7.5	5.8	6.4
Coefficient input	8.3	5.7	6.5
Multiplier output	20.7	10.9	15.5
Accumulator output	14.8	11.5	17.0

TABLE 2. Average Numbers of Transitions per Operation

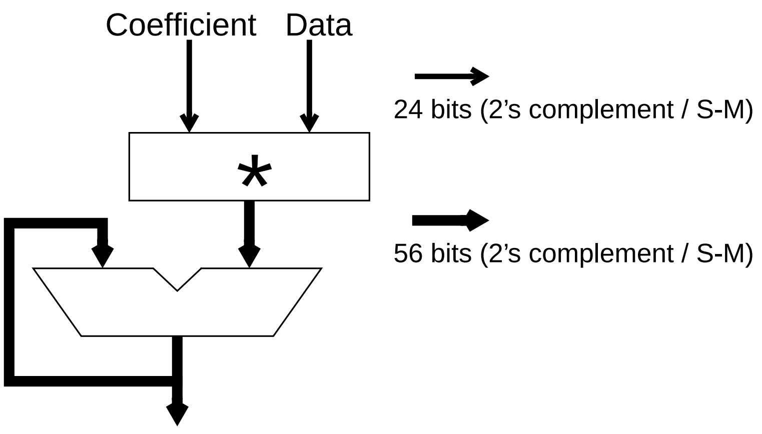

Coefficient
Data
*
24 bits (2's complement / S-M)
56 bits (2's complement / S-M)

Figure 18: Multiply-Accumulate Unit Model

The overall results for each bus in the circuit are shown in Table 2. Both inputs are presented simultaneously to the multiplier, so no spurious transitions are included. It can be seen that, at the multiplier inputs, 2s complement representation has more activity than either sign-magnitude or signed-digit representations. The reason for this can be seen clearly in Figure 19 and Figure 20 which show the transition probability per bit at the data and coefficient inputs (bit 1 is least significant and bit 24 is most significant).

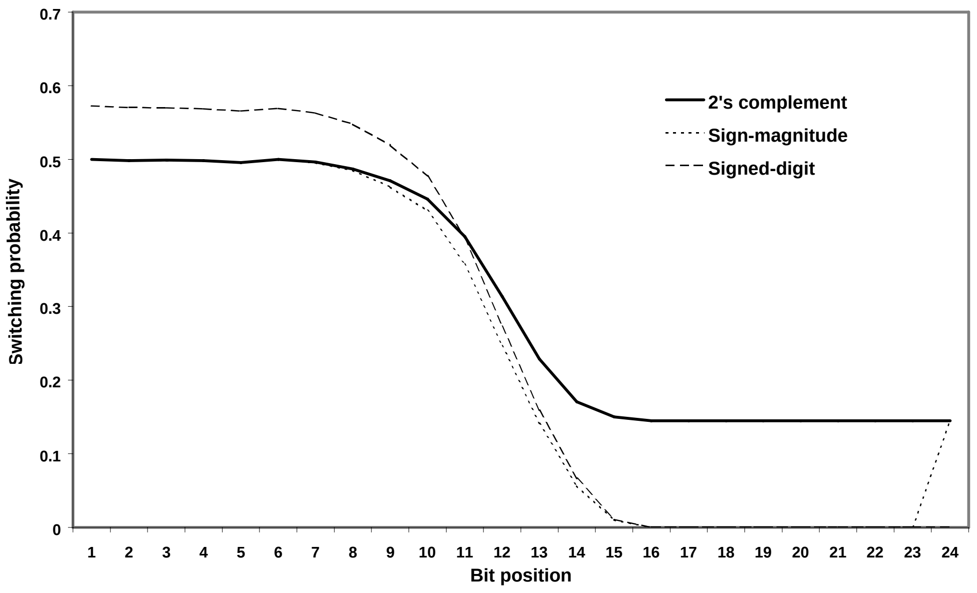


Figure 19: Bit Transition Probabilities for Data Input

The effects of transitions on the sign extension bits for 2s complement causes a noticeable increase in switching activity from around bit 7-9, and from bits 16-23 there are large numbers of redundant transitions as the data is only 16 bits wide. Signed-digit represen-

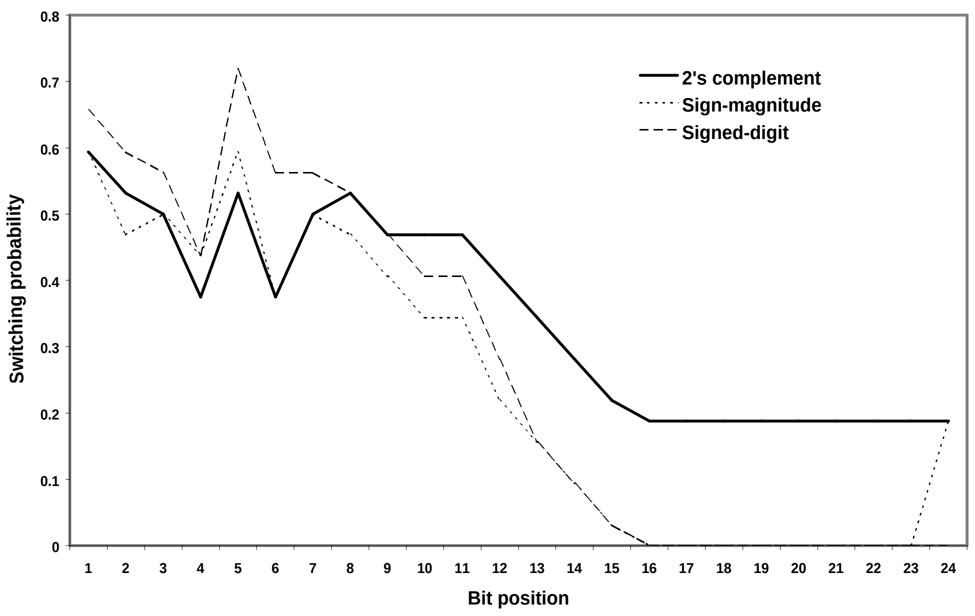


Figure 20: Bit Transition Probabilities for Coefficient Input

tation also has the advantage that there is no sign extension. However, for the bits that are occupied by data, there is a slight increase in switching activity. This is due to the fact that each bit is represented by twice as many bits as sign-magnitude or 2s complement.

Switching transitions at the multiplier output are shown in Figure 21. The extra difference between the 2s complement and signed magnitude representations are explained by the fact that there is a greater string of redundant sign-extension bits at the multiplier output. The increase in switching activity for the redundant signed-digit representation over the 2s complement and sign-magnitude representations is even greater in this case. This is because the multiplier output contains ones in both the positive and negative positions of the signed-digit word, while the input only has ones in either the positive or the negative position depending on the sign of the input.

Table 2 shows that there is a significant reduction in the difference in switching activity between the 2s complement and sign-magnitude representations at the accumulator output. The reason for this can be seen in Figure 22. The probability of transitions in the sign extension bits is reduced by approximately half, due to the smoothing effect of the accumulator.

4.5 Conclusions

The overall conclusion that can be drawn from these results is that sign-magnitude representation does offer significantly lower switching activity than 2s complement representation for data with these characteristics. There were no long periods of silence in the

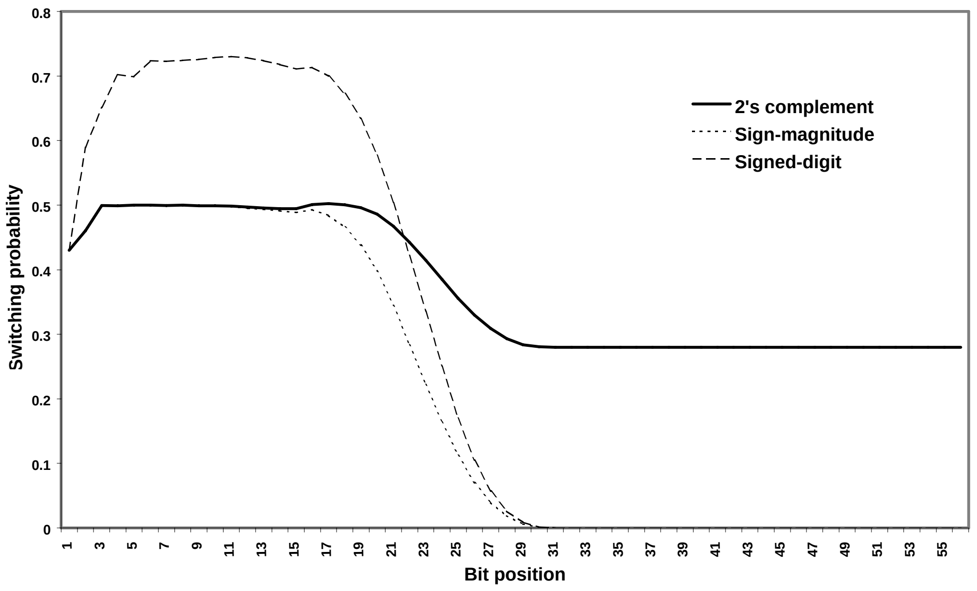


Figure 21: Bit Transition Probabilities for Multiplier Output

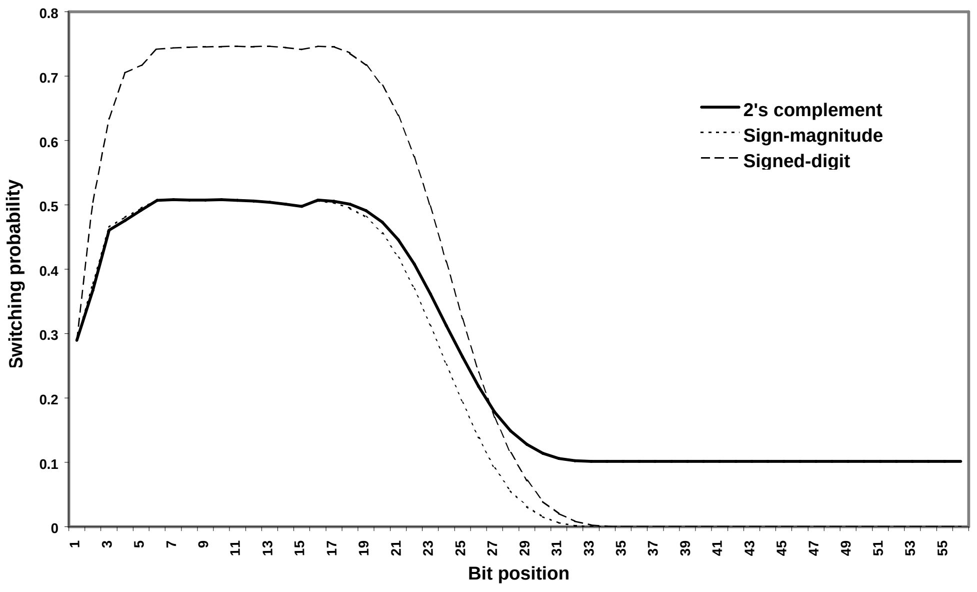


Figure 22: Bit Transition Probabilities for Accumulator Output

input data sample: if there were, the difference would have been even greater due to the large amount of switching activity generated in the sign-extension bits for 2s complement representation. However, no account was taken in the results of the necessity to perform subtraction operations when adding two signed-digit numbers with different signs. If the signed digit numbers are converted back to 2s complement in order to do this, then the only benefit from using the sign-magnitude scheme will be that gained from reduced switching activity on buses. The alternative to this is to include a dedicated subtracter, which is likely to be unacceptable due to area constraints. More detailed testing needs to be performed which takes into account the internal transitions required to perform the operations.

Modified signed-digit representation gives a net increase in switching activity when compared with 2s complement in this case, due to the introduced redundancy causing increased switching activity. When coupled with either the increased storage requirement or the need to convert to a non-redundant system by using a subtracter, this does not appear to be a good choice for an overall number system. Its main use would be as a number representation internal to the multiplier in a signed-digit system, where a subtracter is available at the output of the multiplier in order to convert to non-redundant form.

5 Conclusions

We believe that an asynchronous design offers significant advantages over a synchronous one in an area where the load is variable. This arises because of an asynchronous circuit’s ability to adapt easily to its workload, but it is necessary to use careful design to ensure that there is not excess activity in the device when workload is low.

Significant advantage can also be brought by the ability of an asynchronous circuit to implicitly exhibit very fine grain power management, and its ability to switch from zero power to maximum throughput on demand, which allows the detection of a variety of conditions where no useful work is being done by the chip.

Our experiences of designing large asynchronous processors cause us to believe that micropipelines and bundled data offer the most practical engineering solution to the question of which asynchronous style to use when large general purpose systems are to be designed. *Tangram* and *Balsa* offer a very useful and usable way to design asynchronous circuits but are currently less able to deal with datapath-based designs.

Applying asynchronous micropipeline techniques to DSP applications, especially in an operating environment such as that of mobile communication, should reveal similar benefits to those observed in microprocessor designs.

One important difference between microprocessor and DSP designs is the aspect of number representation. In a DSP device there may be some significant power advantage in opting for a sign and magnitude number system in the datapath rather than conventional 2s complement, even with variable bus widths. All of the other number schemes investigated prove to be inferior from a power perspective.

Our investigations are currently being extended to cover a more extensive DSP architecture than the multiply-accumulate system investigated thus far in order to establish whether the advantages so far observed are applicable across more general DSP structures.

6 Bibliography

[1] van Berkel, K., Rem, M.,*VLSI Programming of Asynchronous Circuits for Low Power*, Asynchronous Digital Circuit Design, Springer-Verlag, pp152-210, 1995.

[2] Chandrakasan, A. P., Brodersen, R. W.,*Minimizing Power Consumption in Digital CMOS Circuits*, Proceedings of the IEEE, vol. 83, no. 4, April 1995.

[3] Erdogan, A., Arslan, T., *Low-Power Multiplication Scheme for FIR Filter Implementation of Single Multiplier CMOS DSP Processors*, Electronics Letters, vol. 32, no.21, pp1959-1960, 1996.

[4] Nielsen, L., Sparsø, J., *A Low Power Datapath for a FIR Filter Bank*, Proc. International Workshop Symposium on Advanced Research in Asynchronous Circuits and Systems, IEEE Computer Society Press, March 1996.

[5] Sutherland, I. E., *Micropipelines*, Communications of the ACM, vol. 32, no. 6, pp720-738, June 1989.

[6] Hauck, S., *Asynchronous Design Methodologies: An Overview*, Proc. IEEE, vol. 83, no. 1, pp69-93, January 1995.

[7] Meng, T. H. Y., Broderson, R. W., Messerschmitt, D. G., *Automatic Synthesis of Asynchronous Circuits from High Level Specifications*, IEEE Trans. on CAD, vol. 8, no. 11, pp1185-1205, Nov. 1989.

[8] Unger, S. H., *Asynchronous Sequential Switching Circuits*, Wiley Interscience, New York, NY, 1969.

[9] Lewis, M., Garside, J., Brackenbury, L., *Latch Controller Operating Modes in Asynchronous Circuits*, Proc. 4th UK Asynchronous Forum, July 1998.

[10] Molnar, C.E., Fang, T.-P., Rosenberger, F. U., *Synthesis of Delay-Insensitive Modules*, Proc. 1985 Chapel Hill Conf. on Advanced Research on VLSI, pp263-278, 1990.

[11] Ebergen, J. C., *Translating Programs into Delay-Insensitive Circuits*, Centre for Mathematics and Computer Science, Amsterdam CWI Tract 56, 1989.

[12] Ebergen, J. C., *A Formal Approach to Designing Delay-Insensitive Circuits*, Distributed Computing, vol. 5, no. 3, pp107-119, July 1991.

[13] Martin. A. J., *Programming in VLSI: From Communicating Processes to Delay-Insensitive Circuits*, in UT Year of Programming Institute on Concurrent Programming, ed. C. A. R. Hoare, Addison Wesley, pp1-64, 1989.

[14] Murata, T., *Petri Nets: Properties, Analysis and Applications*, Proc. IEEE, vol. 77, no. 4, pp541-580, 1989.

[15] Segars, S., *Low Power Microprocessor Design*, MSc. thesis, University of Manchester, 1996.

[16] Colinge, J.-P., *The Development of CMOS/SIMOX Technology*, Microelectronic Engineering, vol. 28, pp423-430, 1995.

[17] Vanbekbergen, P., Catthoor, F., Goossens, G., De Man, H., *Time & Area Performant Synthesis of Asynchronous Control Circuits*, Proc. TAU'90, 1990.

[18] Kishinevsky, M. A., Kondratyev, A. Y., Taubin, A. R., Varshavsky, V. I., *On Self-Timed Behaviour Verification*, Proc. TAU'92, March 1992.

[19] vanBerkel, K., *Handshake circuits: an intermediary between communicating processes and VLSI*, Technische Universiteit Eindhoven, 1992.

[20] Davis, A., Coates, B., Stevens, K., *The Post Office Experience: Designing a Large Asynchronous Chip*, Proc. 26th Annual Hawaii Conf. on Systems Sciences, vol. 1, pp409-418, 1993.

[21] Nowick, S. M., Dill, D. L., *Automatic Synthesis of Locally-Clocked Asynchronous State-Machines*, Proc. ICCAD, pp318-321, 1991.

[22] Rosenberger, F. U., Molnar, C., E., Chaney, T. J., Fang, T.-P., *Q-Modules: Internally Clocked Delay-Insensitive Modules*, IEEE Trans. Computers, vol. 37, no. 9, pp1005-1018, 1998.

[23] Bardsley, A., Edwards, D., Compiling the language Balsa to delay insensitive hardware, Proc. IFIP TC10 WG10.5 Int. Conf. Computer Hardware Description Languages and their Applications, ed. Carlos Delgado Kloos and Eduard Cerny, pp89-91, April 1997.

[24] Endecott, P. B., Furber, S. B., *Modelling and Simulation of Asynchronous Systems using the LARD Hardware Description Language*, Proc. 12th European Simulation Multiconference, Manchester, June 1998, Society for Computer Simulation International, pp39-43. ISBN 1-56555-148-6.

[25] Furber, S. B., Garside, J. D., Gilbert, D. A., *AMULET3: A High-Performance Self-Timed ARM Microprocessor*, Proc. ICCD'98, 1998.

[26] Cortadella, J., Kishinevsky, M., Kondratyev, A., Lavagno, L., Yakovlev, A., *Petrify: a tool for manipulating concurrent specifications and synthesis of asynchronous controllers*, IEICE Trans. Information and Systems, vol. E80-D(3), pp315-325, 1997.

[27] Mutoh, S., et al, *1-V Power Supply High-Speed Digital Circuit Technology with Multithreshold-Voltage CMOS*, IEEE J. Solid-State Circuits, vol. 30, no. 8, pp847-854, 1995.

[28] Wei, L., Chen, Z., Johnson, M. C., Roy, K., De, V., *Design and Optimization of Low Voltage High Performance Dual Threshold CMOS Circuits*, ACM/IEEE Design Automation Conf., pp489-494, 1998.

[29] Oowaki, Y., et al, A Sub-0.1um Circuit Design with Substrate-over-Biasing, ISSCC Digest Technical Papers, pp88-89, 1998.

[30] Krishnamurthy, R. K., Carley, L. R., *Exploring the Design Space of Mixed Swing QuadRail for Low Power Digital Circuits,* IEEE Trans. VLSI Systems, vol. 5, pp383-400, December 1997.

[31] Davis, A., Coates, B., Stevens, K., *The Post Office Experience: Designing a Large Asynchronous Chip*, 26th Hawaii Int. Conference on System Science (HICSS 1993, pp409-418, January 1993.

[32] van Berkel, K., Burgess, R., Kessels, J., Peeters, A., Roncken, M., Schalij, F., *A Fully-Asynchronous Low-Power Error Corrector for the DCC Player*, IEEE J. Solid-State Circuits, vol. 29, no. 12, pp1429-1439, Dec. 1994.

[33] van Gageldonk, H., Baumann, D., van Berkel, K., Gloor, D., Peeters, A., Stegmann, G., *An Asynchronous Low-Power 80C51 Microcontroller*, Proc. International Workshop Symposium on Advanced Research in Asynchronous Circuits and Systems, IEEE Computer Society Press, pp96--107, 1998.

[34] Motorola Inc., *DSP56000 24-bit Digital Signal Processor Family Manual*.

[35] Higgins, Richard J. / Analog Devices, *Digital Signal Processing in VLSI*, Prentice-Hall, ISBN 0-13-212887-X.

[36] Nachtergaele, L., Catthoor, F., Kapoor, B., Janssens, S., DeManH., *Low-Power Data Transfer and Storage Exploration for H.263 Video Decoder System*.

[37] Garside, J. D., *A CMOS VLSI Implementation of an Asynchronous ALU*, Asynchronous Design Methodologies (ed. S. Furber, M. Edwards), IFIP Trans. vol. A28, Elsevier Science Publishers, 1993.

[38] Brent, R. P., Kung, H. T., *A Regular Layout for Parallel Adders*, IEEE Trans. on Computers, vol. 31, pp260-264, 1982.

[39] Oklodzija, V. G., Barnes, E. R., *On Implementing Addition in VLSI Technology*, Parallel Distributed Computing, vol. 5, pp716-728, 1998.

[40] Lynch, T., Swartzlander, E. E., *A Spanning Tree Carry Look-Ahead Adder*, IEEE Trans. on Computers, vol. 41, pp931-939, 1992.

[41] Quach, N. T., Flynn, M.J., *High-Speed Addition in CMOS*, IEEE Trans. on Computers, vol. 41, pp1612-1615, 1992.

[42] Sklansky, J., *Conditional-sum Addition Logic*, IRE Trans. on Electronic Computers, vol. 9, pp226-231, 1960.

[43] Bedrij, O. J., *Carry-select Adder*, IRE Trans. on Electronic Computers, vol. 9, pp226-231, 1962.

[44] Liu, J., *Arithmetic and Control Components for an Asynchronous System*, PhD Thesis, AMULET Group, University of Manchester, 1997.

[45] Szabó, N. S., Tanaka, R. I., *Residue Arithmetic and its Applications to Computer Technology*, McGraw-Hill Series in Information Processing and Computers, 1967.

[46] Hitz, M. A., Kaltofen, E., *Integer Division in Residue Number Systems*, IEEE Trans. on Computers vol. 4 no. 8, August 1995.

[47] Sivakumar, R., Dimopoulos, N. J., *VLSI Architectures for Calculating X mod* m, IEE Proceedings on Circuits, Devices, and Systems, vol. 142, no. 5, October 1995.

[48] Chren Jr, W. A., One-Hot Residue Coding for Low Delay-Power Product CMOS Design, IEEE Trans. on Circuits and Systems II: Analog and Digital Signal Processing, vol. 45, no. 3, March 1998.

[49] Booth, A. D., *A Signed Binary Multiplication Technique*, Computer Arithmetic-Benchmark Papers in Electrical Engineering and Computer Science vol. 21 (ed. E.E. Swartzlander), Dowden Hutchinson and Ross Inc. 1980, ISBN 0-87933-350-2.

[50] Wallace, C. S., *A Suggestion for a Fast Multiplier*, Computer Arithmetic- Benchmark Papers in Electrical Engineering and Computer Science vol. 21 (ed. E.E. Swartzlander), Dowden Hutchinson and Ross Inc. 1980, ISBN 0-87933-350-2.

[51] Moshnyaga, V. G., Tamaru, K., *Impact of Adding Schemes on Switching Activity of Digital Multipliers*, Power and Timing Modelling, Optimization and Simulation (PATMOS) 1995.

[52] Laurent, B., Saucier, G., *Performance / Power Tradeoffs in ASIC Multipliers*, Power and Timing Modelling, Optimization and Simulation (PATMOS) 1997.

[53] Fried, R., *Algorithms for Power Consumption Reduction and Speed Enhancement in High-Performance Parallel Multipliers*, Power and Timing Modelling, Optimization and Simulation (PATMOS) 1997.

[54] Balakrishnan, Burgess, *A Very-high-speed VLSI 2s-Complement Multiplier Using Signed Binary Digits*, IEE Proceedings-E, vol. 139, no. 1, January 1992.

[55] Takagi, N., Yajima, S., *High-Speed VLSI Multiplication Algorithm with a Redundant Binary Addition Tree*, IEEE Trans. on Computers, vol. C34, no. 9, September 1985.

[56] Farnsworth, C., Edwards, D. A., Sikand, S. S., *Utilising Dynamic Logic for Low Power Consumption in Asynchronous Circuits*, Proc. International Workshop Symposium on Advanced Research in Asynchronous Circuits and Systems, IEEE Computer Society Press, pp186-194, November 1994.

[57] Avizienis, A., *Signed-Digit Number Representations for Fast Parallel Arithmetic*, IRE Transactions on Electronic Computers, vol. 10,, pp. 389-400 September 1961.