Energy Efficient Functional Unit For A Compute-Intensive Asynchronous Parallel DSP

W. Suntiamorntut, L.E.M. Brackenbury
APT Group, Department of Computer Science, The University of Manchester,
Oxford Road, Manchester, M13 9PL, UK

Abstract

A functional unit (FU) that is well suited to high computational-intensive DSP systems, and combines several techniques to achieve high energy efficiency is presented. This FU consists of a datapath enabling parallel operations within it. A user configurable RAM memory specifies the operations required and gives the user and system programmer great flexibility in executing algorithms. Not only multiply accumulator operations can be performed by the FU, but other necessary instructions of a general DSP such as the hamming distance are also implemented. The operation flow is controlled using a self-timed circuit style to further reduce power and spread the circuit switching. Post-layout simulation of our full-custom FU implemented on 0.18μm operating at 1.8 V with an area 0.36 mm^2 shows a large energy improvement by a factor of 10 compared to the original design whilst performance results show that using four-way parallelism DSP yields a high throughout rate of equivalent to 800 MHz.

1. Introduction

As is well-known, the capabilities of computation in portable applications has been increasing exponentially. However, the intensive and continuous computing of hand-held computers and other portable devices are restricted by the source of power. There is no equivalent of Moore's Law in the case of battery technology and only a 20% improvement in capacity of battery technology is expected over the next 10 years [1]. Reiner Hartenstein reported a technology trend graph as shown in Figure1[2]. It is clear that the energy density of existing battery technologies are far from what is needed. Hence, an energy efficient design becomes vital.

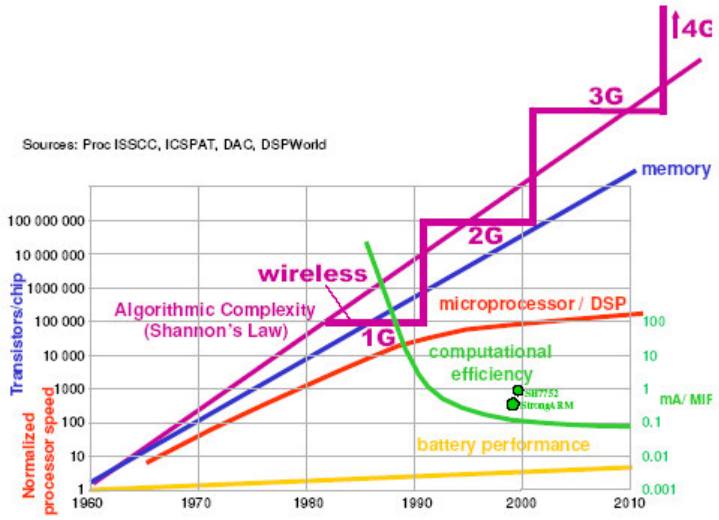

Figure1 Technology trends

Digital Signal Processors (DSPs) have been developed for wireless applications such as mobile handsets. Mostly, mobile phones are driven by cellular standards. In the first generation, many types of filter were expected to run on the DSPs. However, a modern mobile phone handles many more applications such as video decoding, data processing and speech recognition. Multiple standards are also needed in one device. Therefore, the trend for DSP architectures is parallelism; exploiting more than one processing unit gains high throughout. Whilst the area is increased by employing multiple computing units, the energy consumption is decreased by scaling down the supply voltage. However, there is a question as to whether large voltage scaling improves the energy efficiency since theoretically, the sub-threshold supply voltage leakage energy becomes dominant. For moderate voltage scaling, the logic depth, amount of switching and the workload factor of the circuit determine

the energy efficiency. In [3], it is reported that the operating voltage should be scaled to approximately 30% of the maximum supply voltage under typical workload requirements.

If the energy-delay product metric is used as the basis of comparison between designs, then the overall energy efficiency can be exponentially improved in a DSP with parallel FUs since each FU can be operated at lower throughput. This is because when the energy-delay product is taken into account, it means each component uses the smallest energy source per operation. When the speed has been slowed down, each component will dissipate less power. Meanwhile, the overall throughout of the system is still maintained because of the parallel architecture. Therefore, the system can gain an energy efficiency improvement merely by introducing parallelism.

An alternative approach to make an energy efficient system is to structure algorithms to fit the available hardware resource in a DSP. This usually has restrictions with the existing hardware architecture. Therefore, at this level, system may not gain a big improvement. When several energy efficient design techniques are combined, the system will give a large energy improvement when running on a powerful architecture.

Another requirement of a DSP is flexibility. In recent years, many research works [4-7] propose reconfigurable digital signal processors with designs implemented on Field Programmable Gate Array (FPGA) technology. However, the power dissipation is still relative high. New generation FPGA includes specific arithmetic units, accelerators, IP cores are these provides the speed and flexibility but still tend to be high power. The research challenge is to develop a design for both energy efficiency and flexibility. Portable systems need a processing unit that gives them the lowest energy consumption but at the same time must provide enough flexibility to the user and programmer. The functional unit presented here is an energy efficient and flexible component for an asynchronous parallel DSP performing computationally intense algorithms. A configuration memory has been attached to each energy efficient FU in our design for maximum. Internally, the design combines several energy efficient circuit/logic design techniques. The detail of our DSP architecture will be described briefly in the next section followed by the instruction set for our system and an example of a simple 2-D discrete cosine transform (DCT) to demonstrate the exploitation of parallelism in our DSP. Our FU architecture including arithmetic, logical unit and the configuration memory are described in section 3. The low power circuits used in the datapath are described in section 4. Simulation results from the full-custom layout are presented in section 5 demonstrating the improvements to energy achieved and also contains some concluding remarks.

2. Digital Signal Processor

A. Architecture Overview

Our FU has been designed and implemented for a parallel asynchronous DSP named CADRE [8]. CADRE was proposed to be a minimum power consumption DSP whilst meeting the performance requirements of next-generation cellular phones. However, the simulation results show that the power dissipation is still on the high side. In [8], the power dissipation of CADRE was analyzed and approximately 50% of the overall power consumption was found to be dissipated in the FU. Therefore, the new FU in this research will replace on the original design. The original philosophy of the architecture is described to help the reader understand our FU easily.

CADRE was implemented by exploiting four-way parallelism as this appears to be optimal for power reduction, Chandrakasan and Brodersen [9]. This is based on the premise that area can be traded for increased speed because silicon area is rapidly becoming less expensive. Most of the DSP activity can be characterized by frequent repetition of fixed instruction sequences. So, the instruction encoding which determines the selection and passage of data for each operation can be predetermined and stored in advance in a configurable memory which is located locally to each FU. These encodings can then be recalled with a compressed instruction. Because the configuration memories are RAMs, this allows reconfiguration at any point in execution. In addition, CADRE the encodings could be expanded within the FUs. This dramatically reduces the size and amount of information that needs to be fetched from main memory. CADRE used a dual Harvard architecture that has one program memory and two separate data memories. In addition, a large on-chip register file of 256 16-bit words was included to avoid traffic and power dissipation in the main memories. In this way, the operands required by the FUs were provided directly from a register file. As with other DSPs, a 32-entry

instruction buffer was also included to handle loop instructions and reduce traffic to or from the program memory. Finally, all standard hardware components in CADRE were operated using self-timed techniques.

This top-level architecture has been adapted for the current research work because time is too limited to fully implement the CADRE design. However, we still keep the major advanced feature such as four-way parallelism to give high throughput. Meanwhile, a new FU has been designed with its configuration memory. The new system consists of four FUs connected together with a global bus and a pair of FUs are connected locally. The input data of each FU will be directly provided from on-chip RAM blocks with the output data being kept in another RAM block. The encoded top-level instruction is stored in a program memory. It contains controllable to enable the FU and accumulator writeback plus a 5-bit address which accesses the configuration memory associated with each FU; the functional unit instructions are stored in advance into each configuration memory of the FUs.

B. Instruction Set

The instruction set of the proposed FUs has two sets of instructions: computation and data movement and these can occur concurrently.

1) *Computational instructions*: These instructions contain arithmetic and logical operations; the arithmetic instructions include distance, normalization, shift, ADD, SUB, MPY and MAC (multiply and accumulate). The output destination of the operation can be 1 of 4 accumulator registers (AccA to AccD) inside the FU.
2) *Data Movement Instructions:* These instructions process the data movement in or out of the FU. These instructions include an accumulator register to accumulator register transfer within the FU itself, an accumulator register to an accumulator register in another FU, or a movement of data to the output RAM.

C. Example Assembly Codes

The two-dimensional discrete cosine transform (DCT) is a member of the family of sinusoidal transforms and is often used in image compression. An 8x8 2-D DCT has been implemented for compression algorithms such as JPEG. An algorithm for computing the 2-D DCT on a single 8x8 block in high-level C code is given below:

```
for (k = 0; k < 8; k++)
    for (l = 0; l < 8; l++)
    {
        sum = 0;
        for (i = 0; i < 8; i++)
            for (j = 0; j < 8; j++)
                sum += x[i][j] * c[i][k] * c[j][l];      // x[i][j] is 16 bits, c[i][k] and c[j][l] are
                                                         //      12 bits, and sum is 32 bits
        y[k][l] = (sum + 2^15) >> 16;                    // y[k][l] is 16 bits
    }
```

An example of the single 8x8 block of 2-D DCT forming the output y[0][0] optimally mapped to the four-way parallelism of the FU architecture is shown in Table1. The sampling data, coefficients, the input and output occupy 16 bits whilst the sum occupies 32-bits. The primary datapath for this FU is a 40-bit datapath. In the first two cycles, only multiplication is performed. Thereafter, the accumulators hold the valid data so a multiply-accumulate (MAC) operation is performed. When all MAC operations are complete, the four products of the 8 sums reside in accumulator registers A and B of each FU. The sum of loop j can be produced by combining them. Finally, all 8 products for loop i are then combined to produce y[0][0]. However, the final result needs to be rounded before writing the result back to RAM. This is repeated to apply the loops for k and l generating the output y(k,l). The way that operations are mapped to the hardware dramatically reduces the amount of switching activity within the multiplier because the value on the data bus within each FU is held constant for two successive instructions. In addition, the frequency of coefficient reading from memory is reduced by factor of two.

3. Energy Efficient Functional Unit Architecture

The challenge here is to meet the requirements of future portable applications, such as mobile phones. These devices have a very small power budget but need high performance with a complexity approaching that of a

desktop computer. The throughput has been achieved by adopting a parallel architecture allowing up to 4 instructions to be processed in parallel. In addition, parallel arithmetic logic is also used within a FU. The next challenge is therefore to scale the supply voltage downwards whilst maintaining the performance with the use of parallelism at different levels. Therefore, our FU can support supply voltage scaling without sacrificing the overall performance.

Table 1: Mapping 2-D DCT (8x8) onto 4-FUs to find y[0][0]

FU0	FU1	FU2	FU3
AccA= x[0][1]*C[1][0]	AccA= x[1][0]*C[0][0]	AccA= x[2][0]*C[0][0]	AccA= x[3][0]*C[0][0]
AccB= x[4][1]*C[1][0]	AccB= x[5][0]*C[0][0]	AccB= x[6][0]*C[0][0]	AccB= x[7][0]*C[0][0]
AccA= x[0][2]*C[2][0]+AccA	AccA= x[1][2]*C[2][0]+AccA	AccA= x[2][1]*C[1][0]+AccA	AccA= x[3][1]*C[1][0]+AccA
AccB= x[4][2]*C[2][0]+AccB	AccB= x[5][2]*C[2][0]+AccB	AccB= x[6][1]*C[1][0]+AccB	AccB= x[7][1]*C[1][0]+AccB
AccA= x[0][3]*C[3][0]+AccA	AccA= x[1][3]*C[3][0]+AccA	AccA= x[2][3]*C[3][0]+AccA	AccA= x[3][2]*C[2][0]+AccA
AccB= x[4][3]*C[3][0]+AccB	AccB= x[5][3]*C[3][0]+AccB	AccB= x[6][3]*C[3][0]+AccB	AccB= x[7][2]*C[2][0]+AccB
AccA= x[0][4]*C[4][0]+AccA	AccA= x[1][4]*C[4][0]+AccA	AccA= x[2][4]*C[4][0]+AccA	AccA= x[3][4]*C[4][0]+AccA
AccB= x[4][4]*C[4][0]+AccB	AccB= x[5][4]*C[4][0]+AccB	AccB= x[6][4]*C[4][0]+AccB	AccB= x[7][4]*C[4][0]+AccB
AccA= x[0][5]*C[5][0]+AccA	AccA= x[1][5]*C[5][0]+AccA	AccA= x[2][5]*C[5][0]+AccA	AccA= x[3][5]*C[5][0]+AccA
AccB= x[4][5]*C[5][0]+AccB	AccB= x[5][5]*C[5][0]+AccB	AccB= x[6][5]*C[5][0]+AccB	AccB= x[7][5]*C[5][0]+AccB
AccA= x[0][6]*C[6][0]+AccA	AccA= x[1][6]*C[6][0]+AccA	AccA= x[2][6]*C[6][0]+AccA	AccA= x[3][6]*C[6][0]+AccA
AccB= x[4][6]*C[6][0]+AccB	AccB= x[5][6]*C[6][0]+AccB	AccB= x[6][6]*C[6][0]+AccB	AccB= x[7][6]*C[6][0]+AccB
AccA= x[0][7]*C[7][0]+AccA	AccA= x[1][7]*C[7][0]+AccA	AccA= x[2][7]*C[7][0]+AccA	AccA= x[3][7]*C[7][0]+AccA
AccB= x[4][7]*C[7][0]+AccB	AccB= x[5][7]*C[7][0]+AccB	AccB= x[6][7]*C[7][0]+AccB	AccB= x[7][7]*C[7][0]+AccB
AccA= x[0][0]*C[0][0]+AccA	AccA= x[1][1]*C[1][0]+AccA	AccA= x[2][2]*C[2][0]+AccA	AccA= x[3][3]*C[3][0]+AccA
AccB= x[4][0]*C[0][0]+AccB	AccB= x[5][1]*C[1][0]+AccB	AccB= x[6][2]*C[2][0]+AccB	AccB= x[7][3]*C[3][0]+AccB
AccA= AccA*C[0][0]	AccA= AccA*C[1][0]	AccA= AccA*C[2][0]	AccA= AccA*C[3][0]
AccB= AccB*C[4][0]	AccB= AccB*C[5][0]	AccB= AccB*C[6][0]	AccB= AccB*C[7][0]
AccB= AccB+AccA	AccB=AccB+AccA	AccB= AccB+AccA	AccB=AccB+AccA
AccB=AccB+AccB(FU1)	NOP	AccB=AccB+AccB(FU3)	NOP
AccB=AccB+AccB(FU2)	NOP	NOP	NOP
Y[0][0]=AccBh	NOP	NOP	NOP

3.1 Configuration Memory

The regularity of typical DSP code allows multiple FUs to be employed without the power and area expense of dynamic scheduling hardware. For example, most modern DSPs, such as the TMS320C55x from Texas Instruments, use very long instruction words (VLIWs). In this case, program memory is fetched at the full rate demanded by the FUs and a lot of power is dissipated. Although cache would be a possible method to ameliorate this, there is an energy overhead in searching for a hit in cache memory. In our proposed FU architecture, the VLIW encoded instructions are stored in advance of executing each algorithm in a configuration memory located to each FU. This is different from others commercial DSPs such as the Phillips REAL DSP [10] or the Infineon CARMEL DSP [11] which have a single global configurable memory which is only used for special instructions.

In our design, the VLIW instructions for a particular algorithm are ‘cached’ by software and looked up using a short-form instruction. Up to 64 instructions are produced in a configuration memory this is more than sufficient for an anticipated algorithm; for example the DCT algorithm in table 1 requires only 6 encoded instructions in a configuration memory. Furthermore, the configuration memory instructions are completely user definable. The simple look-up avoids any tag overhead associated with a cache. The configuration memory makes the configuration of the FU flexible, enabling optimization by users. This flexibility has led to additional

hardware costs, particularly in the implementation of the configuration memory. These costs can be justified by the flexibility and performance gained by users. In particular, the use of configurable memory embedded into the design is unique compared with the other DSPs mentioned above. This feature also allows each FU to operate on an independent instruction stream if required.

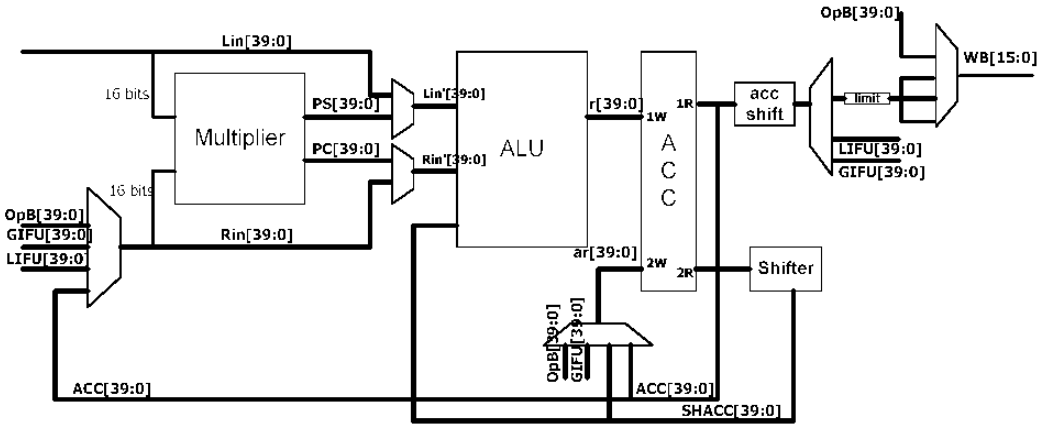

Figure 2 Functional unit datapath

3.2 Energy Efficient Arithmetic-Logical Unit

The FU datapath is shown in Figure 2. The number of available paths and input sources at different points mean that within the FU, many concurrent operations can occur within a single timeslot. For example, multiplication and addition with shifting can be performed in parallel with moving the contents of an accumulator result back to memory. Operations within the FU are termed major and minor. A major operation is usually performed in each timeslot and encompasses the arithmetic and logic operations. Data supplied to or from the FU is usually 16-bits. However written the FU the data is 40-bits wide as is usual in a DSP. Multiplication is 16x16 bits and produces a 40-bit partial sum (PS) and partial carry (PC) which are then forwarded to the adder in the ALU block for completion. Here, up to four operands may need to be added i.e. PS (or Lin) and PC (or Rin) together with an accumulator value (SHACC) which comes via the Shifter and a rounding constant (not shown). The ALU output is written to one of four accumulators AccA to AccD in the ACC.

Minor operations operate concurrently with a major operation. Here, in parallel with a major operation of addition and/or the multiplication, another Accumulator register can be written to either from another (shifted or unshifted) Accumulator register, or from the memory (OpB) or from the global bus GIFU. An Accumulator can also be written back to memory via the WB bus. To prevent unnecessary switching on the buses, transparent latches are inserted in front of multiplier, ALU and the second write port (2W) of the ACC.

The addition of the multiplier outputs PS and PC is performed by the adder in the ALU. This adder is therefore shared between the add/subtract and multiply/multiply-accumulate operations. This architecture is unusual in that most designs have a dedicated adder for the MAC operation and a separate adder provided for the addition. Having just one adder in the FU significantly reduces the amount of logic required in the datapath which in turn reduces power.

A. Asynchronous Circuit Design

The FU unit forms part of an asynchronous pipelined design. Asynchronous timing provides further power improvement as it eliminates clock generation, buffering and distribution. This asynchronous approach also gives a reduction of electromagnetic interference (EMI) as the switching of the logic is spread instead of being concentrated around the clock edge. The FUs are therefore based on the principle of micro-pipelines [12] where the data transfer between blocks uses local handshake signals rather than clocks. The principle is shown in Figure 3. The done signal allowing an output to propagate to the next micro-pipeline stage is generated either from the combinational logic for a data dependent operation or from a matched delay. Where the operation time is data dependent, as in the adder, the operation can be self-timed by using a *completion-detection* (CD) circuit. Many CD techniques are proposed [13-15]. The circuit in the FU has the additional requirement of

being low power. Therefore dual-rail coding having two wires per signal which always return to a '00' state, or duplicate logic producing the done signal which has the same delay as the combinational circuit is not suitable for such a low power application. Current-sensing CDs are also not suitable because they require an additional supply voltage and have higher quiescent current than synchronous circuits negating their advantage. Activity-Monitoring CD (AMCD) has an even higher energy-delay when there is a maximum ripple path. This is disadvantageous in a DSP since a maximum ripple path occurs in mamy arithmetic operations. Many designs use the *bounded delay* approach as it is the easiest CD method to implement. However, this is not a proper completion-detection method since the delay is fixed and needs to be longer than the maximum delay path. In [15], it is claimed that for combinational circuit having a critical path of less than ten gate-delays then a bounded delay approach works satisfactorily.

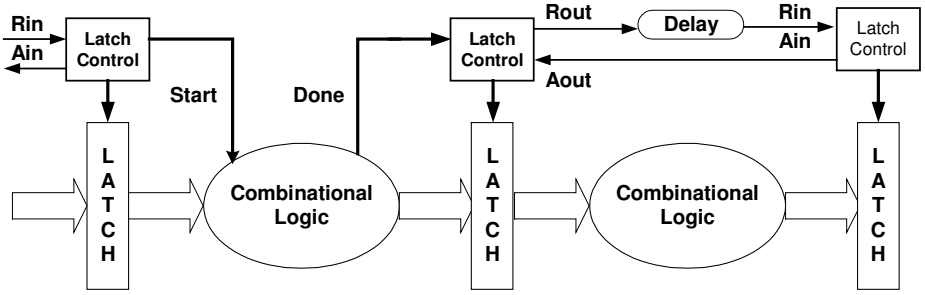

Figure 3 Asynchronous pipeline

Using different delay options for the same logic block has been proposed. For example, Nowick et al. proposed a method called '*speculative completion'* for use in a single rail asynchronous datapath [16]. This uses several different matched delays that allow each component to operate at a different speed. However, it involves extra circuit complexity, area and power dissipation. In our research work, we use our novel method to vary timing in an asynchronous control circuit by varying its supply voltage. It offers many advantages, such as allowing several different matched delays without incurring extra area on power overheads. Tuning the control delay to the datapath also improves performance and offers a flexibility not apparent in other asynchronous timing approaches. It therefore improves the likelihood of obtaining working asynchronous circuits at the first attempt and should lead to greater acceptability of asynchronous design techniques by easing the problem of designing timing and control.

B. **Multiplier Implementation**

The arithmetic unit has been implemented using two's complement rather than the sign and magnitude arithmetic in the initial design; this reduces design complexity, lowers dissipation and gives a better performance. The algorithms for GSM operations indicate a high proportion of multiplication operations. Generally, the multiplier consists of two main components: a partial product generation (PPG) and the addition of those partial products (PPs). The current 16x16 low-power multiplier is shown in Figure 4, has employed a modified Booth's algorithm [17] to reduce the number of PPs by dividing the multiplier bits into groups and selecting multiples of the multiplicand. Although larger groups of multiplier bits can give more PP reduction, a more complex selection table is required, so (as is usual) three bit groups are chosen to maintain speed while remaining low power.

The addition of the eight PPs requires a carry propagate adder (CPA) which has a long latency. To avoid this, a Wallace Tree [18] structure of 4-2 compressors is used which not only improves performance by reducing the latency but also reduces power by significantly reducing the amount of overall logic required. As a result of using tree structure topologies, the number of stages traversed by each input is approximately the same for all inputs. This leads to a *balanced delay tree* and results in less switching activity due to input skew. In addition, the eight PPs have been divided into two groups, PP1-4 and PP5-8 to produce partial carry and partial sum in parallel. Therefore, this multiply structure can achieve both performance and balance the delay in the tree structure. As the multiplier generates 40-bit outputs, all PPs have to be sign-extended. In order to minimize the logic within the tree structure, a pre-calculated sign-extension is applied. Furthermore, eight signed bits from the modified Booth's logic are forwarded to the adder; this reduces the depth of the Wallace tree logic required by one stage.

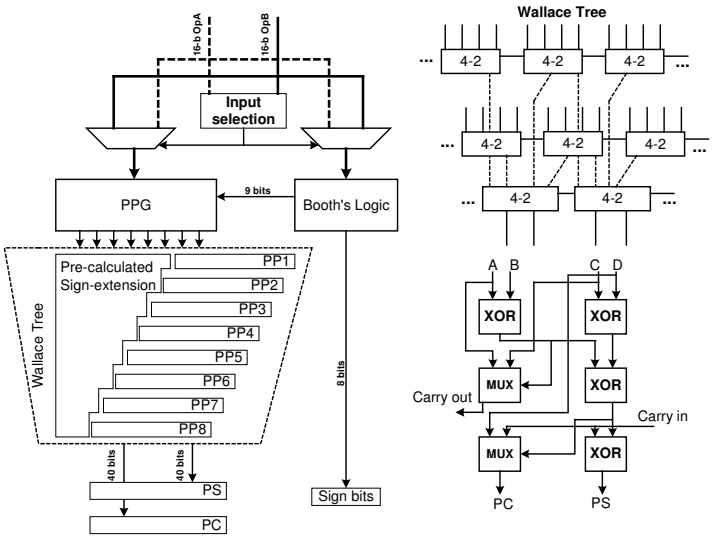

Figure 4 Multiplier Structure

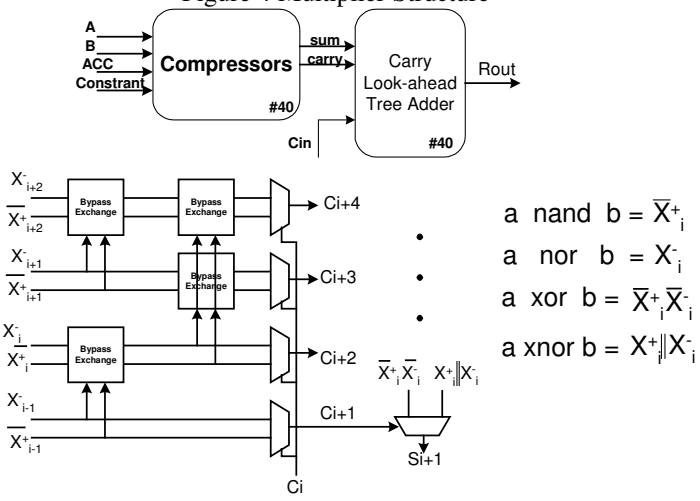

Figure 5 Structure of 4 input 40 bit adder and carry-look-ahead tree.

C. Adder & Associated Implementations

Operation analysis reveals that addition is the most frequent function performed in speech code. Thus, whilst the power dissipation of addition is approximately 1/5 that of multiplication; the very high proportion of additions means that it is essential to minimise the power required for this operation. In the FU, addition involves adding up to four variables as shown in Figure 5. The four inputs are first compressed to two by using 4-2 compressors. The two compressor outputs are then added in a carry-look-ahead tree. The carry-look-ahead equation, given in [19], has been modified when used in our design to enable an easy efficient mapping onto pass-transistor circuits. The carry is generated across blocks of four bits because this offers the best speed-power product. The adder in [19] was implemented using VHDL whilst a synthesis tool then used to generate the circuit and layout; therefore, its energy efficiency was relatively poor. In our design, the logic has been optimised and organised to map efficiently onto low energy pass transistor circuits and the circuits have then been laid out by hand to give an area efficient implementation. Thus the design achieved here is power efficient.

The normalized and Hamming distance instruction are normally included in the instruction set since they are useful for most DSP algorithms. In our design, we combine these functions into only one logic circuit. This not only reduces the number of logic gates but gives lower power dissipation than other designs. This novel logic also gives a performance improvement when compared with the conventional logic of normalized and distance designs. This is because carry-save techniques have been used in preference to the more usual carry ripple in the adder.

4. Energy Efficient Circuits and Layout Implementation

It is important to understand how energy is consumed in a circuit. An energy efficient design can be realised by either minimizing the energy consumption subject to a throughput constrain, or by maximizing the amount of computation for a given amount of energy. The optimal design can be made if the trade-off between the energy and delay can be met since it is possible to determine the lowest energy for a given level of performance. One approach in the system is to incorporate parallelism or pipelining. In addition, use of energy efficient circuits to implement the sub-components in the system result in good energy efficiency. Therefore, this section will discuss and show the energy-delay related to the output loads and transistor sizing. The multiplexer cell as shown in Figure6 implemented by pass-transmission gate is the main cell applied everywhere within the FU datapath. If S is low, the top CMOS pass gate is on passing B to the output, whilst if S is high, the bottom CMOS pass gate is on and A passes to the output.

Gate sizing can be used to trade-off energy and delay. The minimal energy delay point of a circuit is not only affected by its basic design but also by its output load capacitance. The logic gate capacitance and load capacitance including wiring both increase linearly with transistor size. Gate delay can be calculated as $t_d = \tau d$, where τ is a process-dependent constant, and d is a unitless delay of the gate. The unitless delay is determined as $d = h_{eff} + \rho$, where h_{eff} is the product of logical effort of the driving gate and electrical fan-out; a logical effort of 1 results from a minimum size inverter driving a similar inverter. The electrical fan out is the equivalent number of minimum size inputs being driven. The self-loading delay ρ is the product of logical effort (g) and the ratio of the equivalent driven gate width to the equivalent driving gate width, $\rho = gW_{par}/W_{in}$. Because there are no publications relating to the logic effort of a pass-transistor logic as yet, we regard the multiplexer as a logic gate. In our work to find the minimal point of energy delay of multiplexer, the transistor size and loading capacitance for the gate in figure 6 are varied as shown in Table3. The circuits were simulated on SPICE assuming a geometry of 0.18μm and an operating voltage of 1.8V.

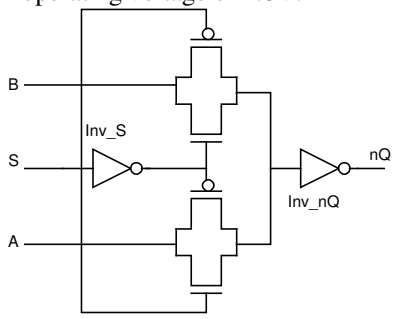

Figure 6 A pass-transmission gate multiplexer

Table 2: Energy delay product simulation results of pass-transmission gate multiplexer using transistor sizing with various load capacitances.

*0.18um geometry operating at 1.8V. CASE	Load Capacitance = 10 ff (equal to 4 Inverters)		Load Capacitance = 20 ff (equal to 8 Inverters)		Load Capacitance = 30 ff (equal to 12 Inverters)	
	Et (pJ) (worse case)	Et (pJ) (average case)	Et (pJ) (worse case)	Et (pJ) (average case)	Et (pJ) (worse case)	Et (pJ) (average case)
1	6.049	5.691	18.298	14.220	33.778	23.542
2	6.909	6.831	19.989	16.112	37.726	26.848
3	8.302	8.118	22.879	18.265	41.784	29.696
4	8.422	6.258	14.409	11.119	21.769	17.390
5	8.802	6.563	14.895	11.530	22.160	17.795
6	7.791	5.669	14.029	10.585	21.698	16.924
7	6.054	6.054	17.680	11.676	29.061	19.178
8	9.719	6.638	18.110	12.256	29.280	19.703
9	8.460	6.421	16.467	12.282	27.310	20.079
10	7.319	5.159	16.624	11.267	28.157	18.946

CASE	Pass Gates (-e06 m)		Inv_S (-e06 m)		Inv_nQ (-e06 m)	
	NMOS	PMOS	NMOS	PMOS	NMOS	PMOS
1	0.28	0.28	0.28	0.28	0.28	0.28
2	0.28	0.28	0.70	1.24	0.28	0.28
3	0.80	0.80	0.70	1.24	0.28	0.28
4	0.80	0.80	0.70	1.24	0.70	1.24
5	1.00	1.00	0.70	1.24	0.70	1.24
6	0.28	0.28	0.70	1.24	0.70	1.24
7	0.28	0.28	0.70	1.24	0.36	1.46
8	0.80	0.80	0.70	1.24	0.36	1.46
9	0.80	0.80	0.28	0.28	0.36	1.46
10	0.28	0.28	0.28	0.28	0.36	1.46

Table 2 shows the energy delay product resulting from the use of different transistor sizes in the multiplexer and different load capacitances. We have analyzed the energy delay product of the circuit with both average rise and fall times and worse case edge times. Clearly, transistor sizing can help the circuits to have a low energy delay product when load capacitance is increased. In practical, load capacitance is very important and has a large effect both the performance and power of the system, especially in a large design. In addition, using the minimal transistor size cannot achieve optimum energy efficiency. It can be concluded from these results, that the CMOS pass gate transistors that for minimum energy-delay should be minimum size, the inv_S should be a drive 1 gate (PMOS/NMOS width = 1.24e-06m/0.70e-06m) and be capable of driving the select signals on 2 or 3 multipliexer gates, and that the inv_nQ should be matched to the driven load (PMOS/NMOS width = 1.24e-06m/0.70e-06m). These figures do not include leakage power which is small for this process and provided the transistor sizing above is adopted.

Finally, a full custom layout is required to build an energy efficient system. This is especially important when the pass-transistor circuit topology is used since we can achieve less area and a non-predictable load capacitance. In addition, the physical capacitance and the length of wires can be minimised. Therefore, full custom design combined with transistor sizing will give a big energy saving compared to the automatic place and route tools which have developed to reduce the energy consumption and which yield only an 18% reduction[20].

5. Discussion and Conclusions

The results from remaining simulations on the full custom layout for the datapath shown in Figure2 indicate that our FU dissipates about 13.68mW@200MHz. Projecting this to four-way parallelism yields 800MHz with a power consumption of only 54.72mW. This indicates a factor of 10 improvement in the dissipation on the original FU design assuming a similar geometry and operating voltage.

Energy efficiency is a significant design constraint for battery powered DSPs requiring a coherent low energy technique applied at all levels and particularly at logic and circuit levels. The proposed FU architecture has been designed to be flexible, have low power and high throughput particularly in performing arithmetic operations involving the multiplier and adder. In addition, asynchronous data dependent operation and a tunable delay mechanism are incorporated to gain even better energy efficiency. At the circuit level, pass transistor logic is used which is low power without sacrificing performance. Transistor sizing has been applied to this pass transistor logic to identify the optimum trade-off between energy and delay.

In the future, the need for energy efficiency design should result in a move from low-level energy-efficient techniques to higher levels as the automatic tools which currently do not support coherent lower energy strategies.

Acknowledgement

This work was funded by EPSRC grant GR/S61270/01 and the authors are grateful for this support. The authors are also grateful to Jim Garside for discussion and valuable suggestions, and to Dave Clark, Jeff Pepper and Steve Temple for encouragement in this work.

References

[1] Sheng S., Chandrakasan A. and Brodersen R.W., "A portable multimedia terminal.", *IEEE Communications Magazine*, pp. 64-75, vol.30, no.12, December, 1992.
[2] Reiner H., "Reconfigurable Computing: A New Business Model and its Impact on SoC Design.", Keynote speeches in *EUROMICRO Symposium on Digital System Design, Architectures, Methods and Tools*, September, 2001.
[3] Bo Z., David B., Dennis S. and Krisztian F., "Theoretical and practical limits of dynamic voltage scaling.", *ACM/IEEE Design automation conference (DAC)*, under review, June, 2004.
[4] Li-Hsun C., Chen O.T.-C. and Ruey-Ling M., "A high-efficiency reconfigurable digital signal processor for multimedia computing.", *Proceedings of the 2003 International Symposium on Circuits and Systems*, vol.2, pp. 768-771, May, 2003.
[5] Sangjin H., Shu-Shin C. and Connaway C., "Variable-rate pipelined multiplier design for reconfigurable DSP applications.", *45th Midwest Symposium on Circuits and Systems*, vol.1, pp. 587-590, August, 2002.
[6] Martina M., Masera G., Piccinini G., Vacca F. and Zamboni M., "Reconfigurable DSP IP for multimedia applications.", *IEEE International Conference on Acoustics, Speech, and Signal Processing*, vol.4, pp. 4179, May, 2002.
[7] I. Verbauwhede, P. Schaumont, C. Piguest and B. Kienhuis, "Architectures and design techniques for energy efficient embedded DSP and multimedia processing.",
[8] M. Lewis and L. Brackenbury. CADRE: An Asynchronous Embedded DSP for Mobile Phone Applications. *Design Automation for Embedded Systems*, vol. 6, No. 4, pp. 451- 475, 2002.
[9]A. P. Chandrakasan and R. W. Brodersen. *Low Power Digital CMOS Design*. Kluwer Academic Publishers, 1995.
[10]P. Kievits, E. Lambers, C. Moerman and R. Woudsma, "R.E.A.L. DSP Technology for Telecom Basedband Processing", *Proceeding 9th International Conference on Signal Processing Applications and Technology*, Miller Freeman Inc., 1998.
[11]*Carmel DSP Core Technical Overview Handbook*, Infineon Technologies, 2000.
[12] I.E. Sutherland, "Micropipelines.", *Communications of the ACM*, vol.32, no.6, pp.720-738, June, 1980.
[13] M.E. Dean, T.E. Williams and D.L. Dill, "Efficient self-timing with level-encoded 2-phase dual-rail (LEDR)." *MIT Conference on Advanced Research in VLSI,* March 1991.
[14] V. Varshavsky, V. Marakhovsy and M. Tsukisaka, "Data-controlled delays in the asynchronous design," *Proceeding Of the 1996 IEEE International. Symposium Circuits and Systems (ISCAS'96),* Atlanta (USA), vol. 4, pp. 153-155, May 1996.
[15] E. Grass, Viv Bartlett and Izzet Kale, "Completion-Detection Techniques for Asynchronous Circuits." *IEICE Transaction Information and System,* vol.E80-D, no. 3, March 1997.
[16] S.M.Nowick, K.Y.Yun, P.A.Beerel and A.E.Dooply, "Speculative Completion for the Design of High-Performance Asynchronous Dynamic Adders.", *Proceedings of Async97*, pp. 210-223, April, 1997.
[17] A. D. Booth, "A Signed Binary Multiplication Technique.", *Quarter Journal Mech. Applicantion Math.*, vol. 4, pp. 236-240, 1951.
[18] C. S. Wallace, "A Suggestion for Fast Multipliers.", *IEEE Transactions Electronic Computer*, vol. EC-13. pp. 14-17, February, 1964.
[19] Keshab K. Parhi, "Low-Energy CSMT Carry Generators and Binary Adder.", *IEEE Transactions on Very Large Scale Integration(VLSI) Systems*, vol.7, no.4, pp.450-462, December, 1999.
[20] Chao K. and Wong D., "Low power considerations in floorplan design.", *International workshop on low power design*, Napa Valley, CA, pp.45-50, April, 1994.