S-38.220
Postgraduate Course on Signal Processing in Communications,
FALL-99

Pipelining and Parallel Processing

Carl Eklund
Nokia Research Center
P.O. Box 407
FIN-00045 Nokia Group
E-Mail: carl.eklund@nokia.com

October 13, 1999

Abstract

This paper presents the techniques of pipelining and and parallel processing. Both methods are commonly used for increasing performance in digital designs. Pipelining introduces latches on the data path thus reducing the critical path. This allows higher clock frequencies or sampling rates to be used in the circuit. In parallel processing logic units are duplicated and multiple outputs are computed in parallel. The level of parallelism directly increases the sampling rate. In addition to increasing performance both techniques can be used to reduce power dissipation.

Contents

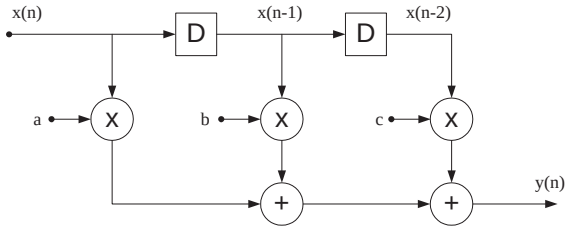

Figure 1: Direct form implementation of 3-tap FIR filter

1 Introduction

A three-tap finite impulse response (FIR) filter is given by

$$y(n) = ax(n) + bx(n-1) + cx(n-2). \tag{1}$$

The direct form block diagram of the filter is shown in figure 1. From the figure it can be seen that the required to process a sample is equal to the time of a multiplication (T_{M}) and two additions ($2T_{\mathrm{A}}$). This is the execution time of critical path. The critical path sets a condition

$$T_{sample} \geq T_M + 2T_A \tag{2}$$

for the sampling period and thus the maximum sampling frequency is limited to

$$f_{sample} \leq \frac{1}{T_M + 2T_A}. \tag{3}$$

If this condition cannot be met the direct form structure must be discarded.

The *effective critical path* can be reduced by introducing pipelining registers on the data path. The principle can be seen from figure 2. The execution time of the critical path for the structure in (a) is $2T_A$. In (b) the same structure is shown as a 2-staged pipelined structure. A latch hs been placed between the two adders thus halving the critical path. This allows operation at a higher sampling rate.

The throughput can also be increased with a completely different technique. In figure 2 (c) the hardware is duplicated so that two inputs can be processed simultaneously. This parallel processing structure doubles the sampling rate.

2 Pipelining

To consider pipelining we need to introduce two definitions.

Definition 1 (Cutset) *A cutset is a set of edges in a graph such that removing the edges makes the graph disjoint.*

Definition 2 (Feed-forward cutset) *A cutset is a feed-forward cutset if data move in the forward direction on all the edges of a cutset.*

Adding latches on a feed-forward cutset of a FIR filter leaves the functionality unchanged. In figure 4 a 2-level pipelined version of the three tap FIR filter is shown. The critical path has

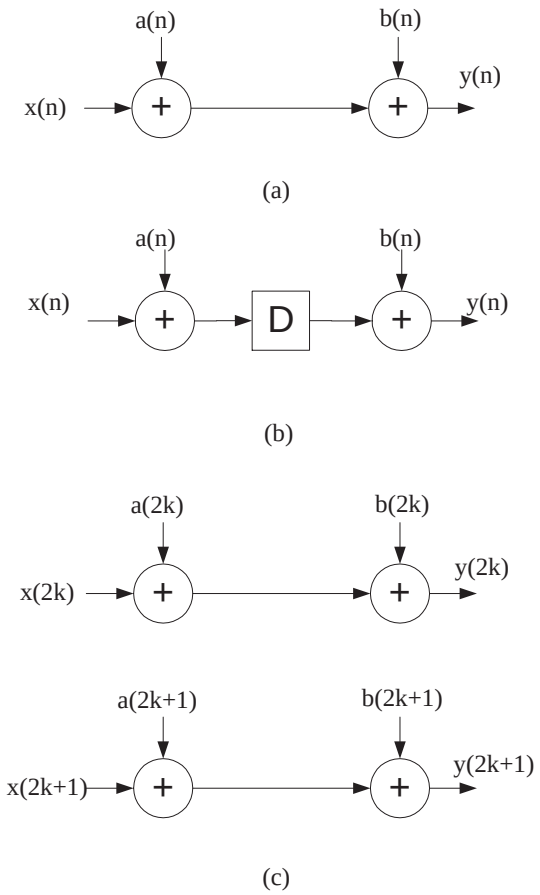

Figure 2: (a) A simple datapath. (b) Pipelined datapath. (c) Parallel datapath.

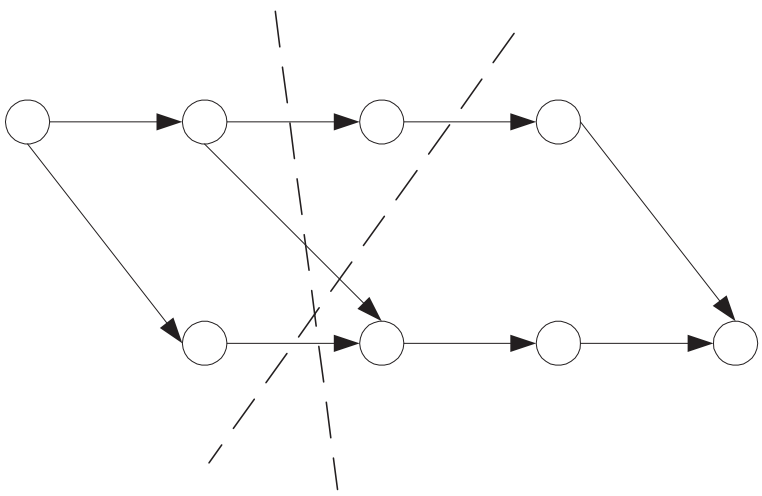

Figure 3: A graph with two cut-sets indicated by the dashed lines

Clock	Input	Node 1	Node 2	Node 3	Output
0	$x(0)$	$ax(0)+bx(-1)$			
1	$x(1)$	$ax(1)+bx(0)$	$ax(0)+bx(-1)$	$cx(-2)$	$y(0)$
2	$x(2)$	$ax(2)+bx(1)$	$ax(1)+bx(0)$	$cx(-1)$	$y(1)$
3	$x(3)$	$ax(3)+bx(2)$	$ax(2)+bx(1)$	$cx(0)$	$y(2)$

Table 1: Scedule of pipelined FIR filter in figure 4

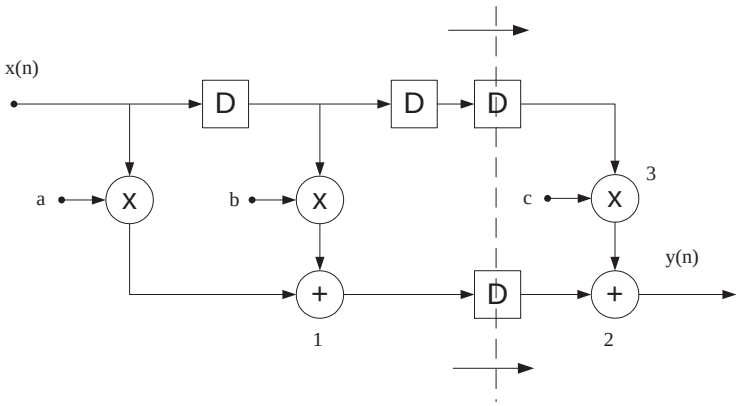

Figure 4: 2-level pipelined implementation of a 3-tap FIR filter. The dashed line shows the cut-set

been reduced from $T_M + 2T_A$ to $T_M + T_A$. As can be seen from table 1 the output is available only after two clock cycles as compared to one for the sequential implementation. In general the latency of M-level pipelined circuit is $M-1$ clock cycles more than that of the sequential circuit[1].

The throughput of the pipelined design is determined by the longest path between any 2 latches or between a latch and the in/output

$$R_{T,M} \sim \frac{1}{\max_i T_{D,i} + T_{D,latch}}, \tag{4}$$

where $T_{D,i}$ is the processing delay of stage i and $T_{D,latch}$ is the delay of the latch. In the improbable situation that all the pipeline stages are equal the throughput is given by

$$R_{T,M} = R_{T,1} \cdot M \frac{T_{D,i} + T_{D,latch}}{T_{D,i} + MT_{D,latch}}. \tag{5}$$

This relation is, however, instructive as it shows that the throughput for large M-no longer increases proportionally due to the delay of the pipeline latches.

Pipelining can be done with any granularity. Figure 5 shows how a 4-bit ripple adder can be pipelined. Note the delay elements on the input operands and outputs due to the cut-sets. The delays on the inputs assure that the carry bit and operands arrives simultaneously to the adder cells. This technique is called pre-skewing. The delaying of some of the output bits, called de-skewing, is necessary to assure simultaneous arrival of the sum bits. Note how the cut-set method automatically and elegantly evaluates the delays required[2].

In FIR filter design the practice of partitioning arithmetic function to sub-functions with pipeline latches is sometimes referred to as fine-grain pipelining.

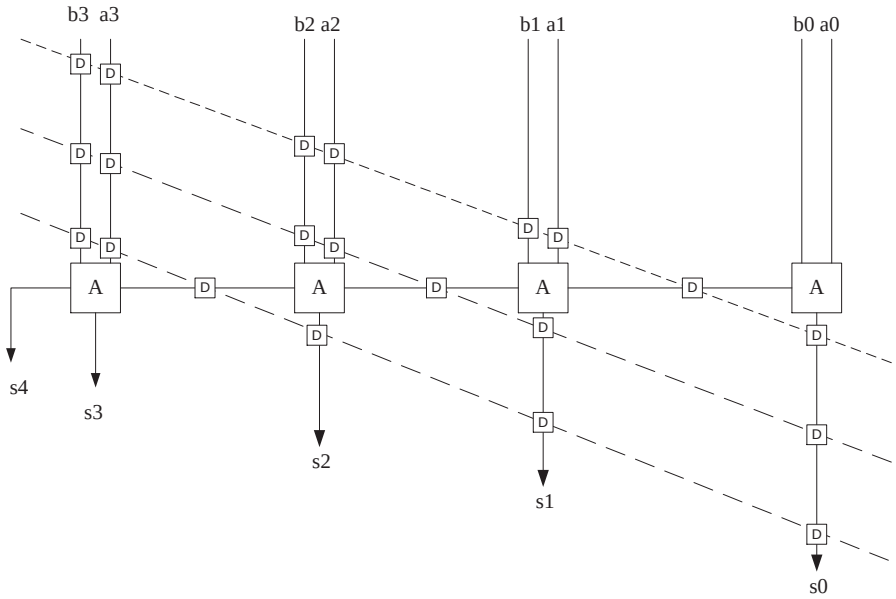

Figure 5: Pipelined ripple adder. The dashed lines show the cut-sets

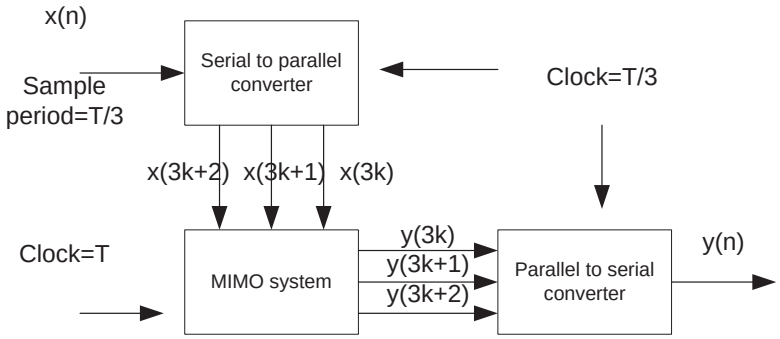

Figure 6: Parallel processing system with blocksize=3

In the previous discussion we have assumed feed-forward cut-sets. Many commonly used algorithms have feed-back loops and thus the cut-sets are not feed-forward. In the general case the rule states that positive delay elements are placed on the edges entering the set of cut-off nodes while an equal negative delay must be place on the edges leaving the set to keep functionality intact. As negative delay is impossible to implement, delay redistribution techniques beyond the scope of this paper must be employed in these cases[2]. An excellent discussion on this topic can be found in [3, 4].

3 Parallel processing

Any system that can be pipelined can also be processed in parallel. In pipelining independent computations are executed in an interleaved manner, while parallel processing achieves the same using duplicate hardware. Parallel processing systems are also referred to as block processing systems. The block size indicates the number of inputs processed simultaneously.

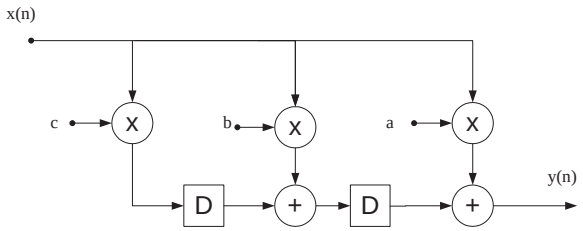

Figure 7: Data broadcast implementation of 3-tap FIR filter

A complete parallel processing system shown in figure 6 contains a serial to parallel converter (DMUX) the MIMO processing block and a parallel to serial converter(MUX). The data paths in the MIMO system either work with an offset of T_{clk}/M in a M-parallel system or the MUX and DMUX must be equipped with delay units allowing simultaneous processing.

The throughput of a M-parallel system is M times the throughput of the sequential system,

$$R_{T,M} = M \cdot R_{T,1}. \tag{6}$$

It should also be noted that for a parallel processing system $T_{clock} \neq T_{sample}$ whereas they are equal in a pipelined system[1, 2].

4 Combining pipelining and parallel processing

Parallel processing and pipelining can also be combined to increase throughput. Figure 7 shows the data broadcast structure of a 3-tap FIR filter. In figure 8 the 3-parallel implementation of the same filter is shown. The throughput of the parallel filter is three times that of the original filter. By introducing fine-grain pipeline registers in the multipliers we end up with the structure in figure 9. If the cutset can be placed so that the processing delays of the subcircuits are equal another factor two can be gained in the throughput[1].

5 Low power design

Pipelining and parallel processing are techniques to increase the sample speed. The same techniques can be used to lower the power consumption at a given speed.

The propagation delay in a CMOS circuit is given by

$$T_{pd} = \frac{C_{charge} V_0}{k\,(V_0 - V_t)} \tag{7}$$

where C_{charge} is the capacitance that is charged/discharged in a single clock cycle, i.e. the capacitance along the critical path. V_0 is the supply voltage and V_t the threshold voltage of the transistor. The constant k is technology dependent.

The power consumption can be estimated by

$$P = C_{total} V_0^2 f. \tag{8}$$

In the equation above C_{total} is the total capacitance of the circuit and f the clock frequency. It should be noted that these equation are based on crude approximations and that the issues

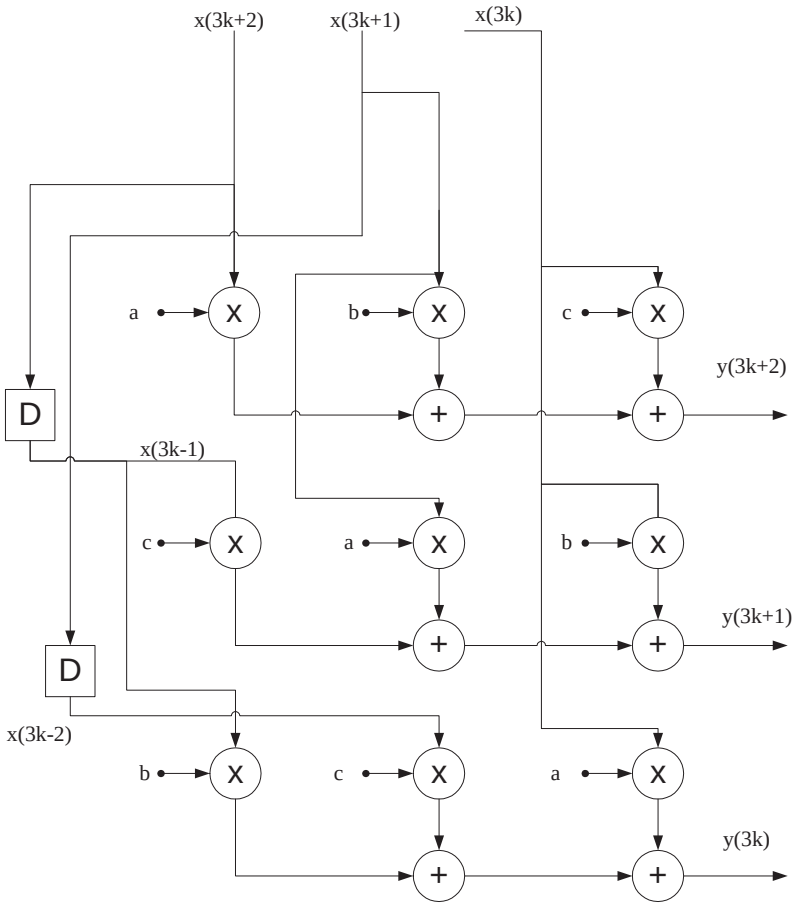

Figure 8: 3-parallel implementation of the filter in figure 7

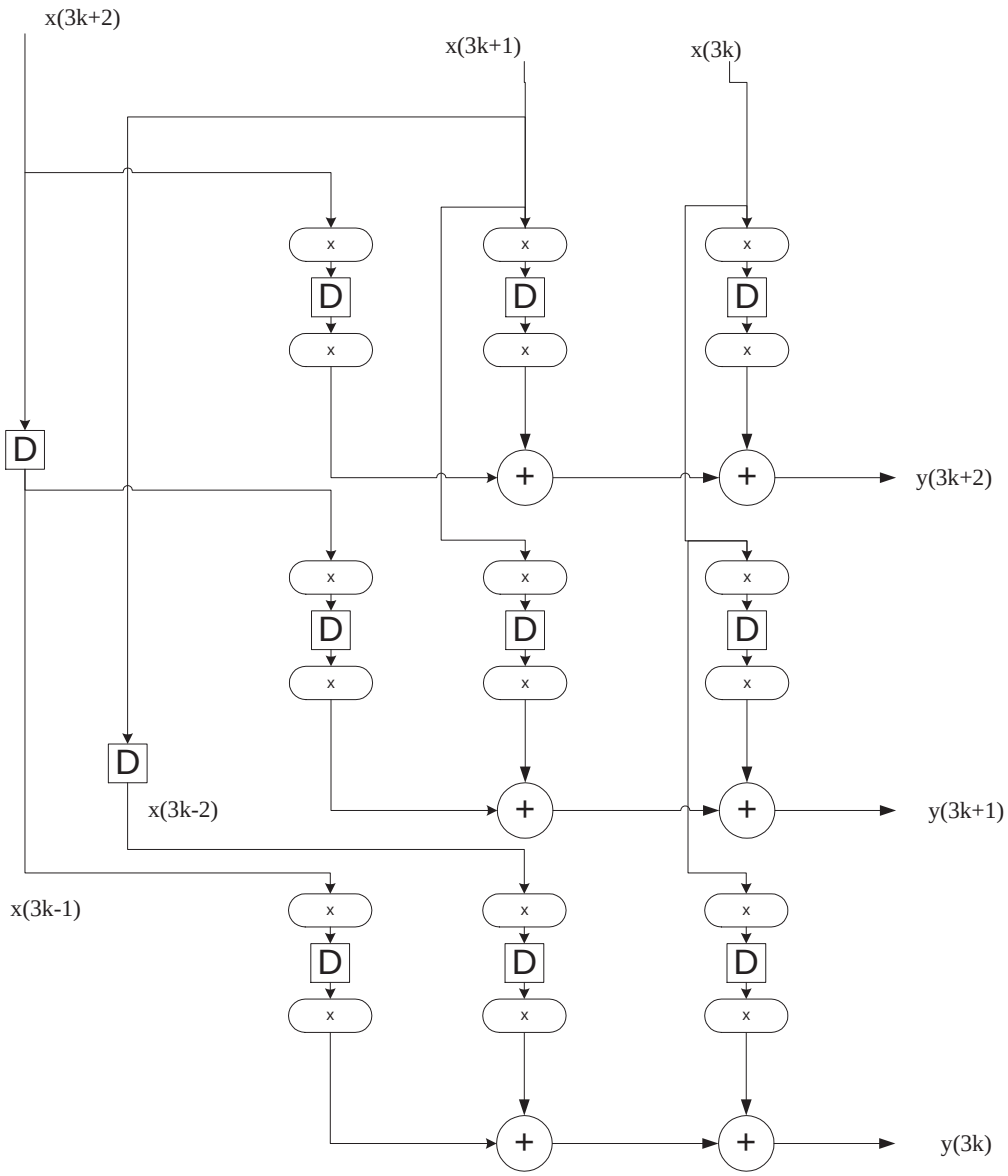

Figure 9: Finegrain pipelined structure of the filter in figure 8

are much more complex. The low power techniques aim to lower the supply voltage and thus reducing the power consumption. It should be remembered that the noise margin puts a lower limit on the supply voltage that can be achieved.

5.1 Power reduction through pipelining

Next the method of lowering power consumption by pipelining is examined using a FIR filter as an example. The technique is, however, also applicable in other cases. The power consumption of the non-pipelined FIR filter can be estimated using equation 8 to be

$$P_{seq} = C_{total} V_0^2 f. \tag{9}$$

The clock frequency f is determined by the processing delay of the filter. If M pipeline latches are introduced, the critical path is reduced to one Mth of the original and the capacitance to be charged/discharged per cycle is now C_{charge}/M. The introduction of the pipelining latches increases the capacitance C_{total} but as a first approximation this increase can be neglected. If we operate the pipelined circuit at the same frequency we note that since only fraction of the original capacitance C_{charge} is charged/discharged per cycle the supply voltage can be reduced to βV_0 where β is a positive constant less than 1. The power consumption of the pipelined filter is then reduced to

$$P_{pip} = C_{total} \left(\beta V_0\right)^2 f = \beta^2 P_{seq} \tag{10}$$

which is lower by a factor β^2 as compared to the original implementation. The clock period is usually set to equal the maximum propagation delay in a design. Noting that both filters run at the same frequency the factor β can be determined with the help of equation 7. Equating the propagation delays results in the equation

$$M \left(\beta V_0 - V_t\right)^2 = \beta \left(V_0 - V_t\right)^2 \tag{11}$$

from which β easily can be solved. The reduced power consumption of the pipelined filter can then be computed using 10[1].

The discussion above totally ignores the fact that probability of glitching is reduced in the pipelined implementation due to the smaller logical depth. Glitches are short termed charge/discharge effects that arise from non-uniform propagation times in networked combinatoric logic. Glitches can contribute significantly to the power consumption. In the case of a carry ripple adder the dissipation due to glitches can be as much as 22% of the total[5]. In general complex simulations are needed to evaluate the power consumption due to glitching[6].

5.2 Power reduction through parallel processing

Parallel processing can also be used to reduce the power consumption by allowing reduction of the supply voltage. In a L-parallel system the charging capacitance remains the same whereas the total capacitance is increased by a factor of L. The serial to parallel and parallel to serial converters required in a parallel processing system also add to capacitance and power consumption but are neglected in the following discussion.

In a L-parallel system the clock period can be increased to LT_{seq} without decreasing the sample rate. As more time is available to charge the capacitance C_{charge} the voltage can be

lowered to βV_0. The propagation delay in the parallel implementation maintaining the sample rate is

$$LT_{seq} = \frac{C_{charge}\beta V_0}{k\left(\beta V_0 - V_t\right)^2}. \tag{12}$$

Substituting (7) for T_{seq} the quadratic equation

$$L\left(\beta V_0 - V_t\right)^2 = \beta\left(V_0 - V_t\right)^2 \tag{13}$$

can be formed, from which β can be obtained. Once β is known the power consumption of the L-parallel system can be calculated as

$$\begin{aligned} P_{par} &= LC_{charge}\left(\beta V_0\right)^2 \frac{f}{L} \\ &= \beta^2 V_0^2 f \\ &= \beta^2 P_{seq}. \end{aligned} \tag{14}$$

As with the pipelined system the power consumption of the L-parallel system has been reduced by a factor of β^2 compared with the original system.

5.3 Combining pipelining and parallel processing

Pipelining and parallel processing can be combined in low power designs. The charging capacitance is lowered by pipelining and parallelism is introduced to allow lower clock speeds. The propagation delay of the parallel pipelined filter is given by

$$LT_{seq} = \frac{\left(C_{charge}/M\right)\beta V_0}{k\left(\beta V_0 - V_t\right)^2} = \frac{LC_{charge}V_0}{k\left(V_0 - V_t\right)^2}. \tag{15}$$

The quadratic equation

$$ML\left(\beta V_0 - V_t\right)^2 = \beta\left(V_0 - V_t\right)^2 \tag{16}$$

is obtained and again β can be solved.

6 Architecture efficiency

Architecture optimization aims at increasing the performance of a design. Throughput is the measure of performance, but measuring it is often problematic. For DSP applications an obvious choice is to measure the data input and result rates. Also computational power,R_C expressed in operations per unit of time is used instead of the throughput, R_T. When comparing computational power the underlying word width has to be considered. Comparing an 8-bit architecture to a 32-bit architecture is like comparing apples to oranges.

The clock period, T_{clk} is a measure for both performance and throughput. The performance in terms of computational rate is given by

$$R_C = \frac{n_{op}}{T_{clk}} \tag{17}$$

with n_{op} being the number of operations carried out in during the clock period. The throughput is can be expressed

$$R_T = \frac{n_s}{T_{clk}} \tag{18}$$

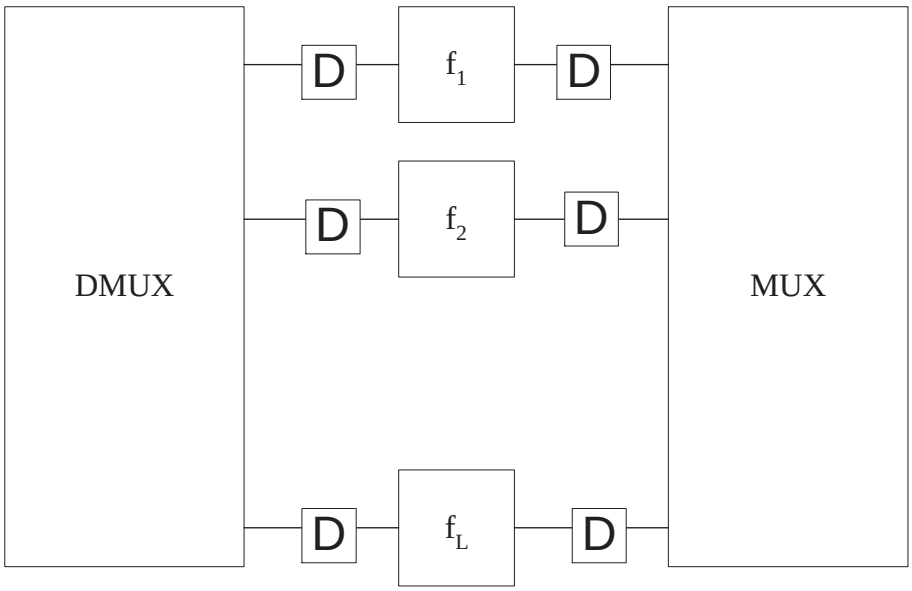

Figure 10: A parallel system with L-subfunctions

where n_s is the number of samples input or output in the time interval T_{clk}. This number is in samples per second and must be multiplied by the number of bits per sample to express the value as bits per second. Often the the computational rate and the throughput are proportional

$$R_C = \frac{n_{op}}{n_s} R_T \tag{19}$$

and the proportionality factor gives the operations per sample.

In integrated circuits the cost of a circuit is dependent of chip size. The size again is roughly proportional to the transistor count. The relationship between chip size and performance is often used to measure the efficiency of an architecture. The efficiency can thus be expressed as

$$\begin{aligned} \eta_T &= \frac{R_T}{A} \\ \eta_C &= \frac{R_C}{A}. \end{aligned} \tag{20}$$

If (19) holds optimization of η_T and η_C leads to the same solution. Combining equations 18,17 and 20 we get the commonly used AT product

$$\eta \sim \frac{1}{AT_{clk}}. \tag{21}$$

6.1 Efficiency of parallel architectures

Now consider the parallel implementation of the identical logic modules shown in figure 10. The efficiency will compared for various degrees of parallelism L. According to 6 the throughput increases in proportion to L. A parallel implementation also consumes additional chip

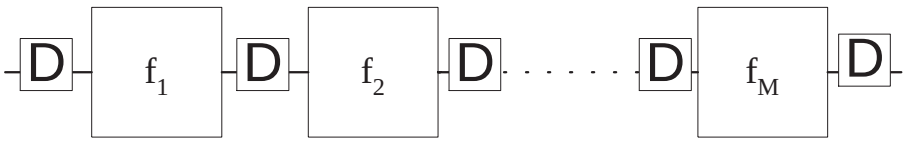

Figure 11: A pipelined system with M-stages

area A_a for data distribution and merging. Assuming this area is proportional to the degree of parallelism exceeding 1 the chip area is given by

$$\begin{aligned} A_L &= LA_1 + (L-1)A_a \\ &= A_1\left[L + (L-1)\frac{A_a}{A_1}\right]. \end{aligned} \tag{22}$$

Combining equations 6 and 22 it follows that the efficiency in terms of parallelism is

$$\eta_L = \eta_1 \frac{1}{1 + \left(1 - \frac{1}{L}\right)\frac{A_a}{A_1}} \tag{23}$$

From the equation it can be seen the efficiency is not improved through parallel processing if additional hardware is required, but rather worsened[2].

6.2 Efficiency of pipelined architectures

In a pipelined structure the pipeline registers affect the critical path and the delay as well as the chip size. If the logic is split into equal delay sub-functions the time dictating the throughput is

$$\begin{aligned} T_M &= \frac{T_1 - T_{reg}}{M} + T_{reg} \\ &= \frac{T1}{M}\left[1 + (M-1)\frac{T_{reg}}{T_1}\right]. \end{aligned} \tag{24}$$

The index now represents an implementation with one final register while index M is for a pipelined system with M pipeline registers, shown in figure 11. The additional pipeline register take up additional space on the die. The area is

$$\begin{aligned} A_M &= A_1 + (M-1)\,A_{reg} \\ &= A_1\left[1 + (M-1)\frac{A_{reg}}{A_1}\right]. \end{aligned} \tag{25}$$

Combining the equations the expression

$$\eta_M = \eta_1 \frac{M}{\left[1 - (M-1)\frac{A_{reg}}{A_1}\right]\left[1 - (M-1)\frac{T_{reg}}{T_1}\right]} \tag{26}$$

for the efficiency can be derived. From the result it can be seen that the efficiency increases rapidly as long as the contributions of the pipeline registers and delay are insignificant. The optimum value for M can be found to be

$$M_{opt} = \sqrt{\frac{\left(1 - \frac{A_{reg}}{A_1}\right)\left(1 - \frac{T_{reg}}{T_1}\right)}{\frac{A_{reg}T_{reg}}{A_1T_1}}}. \tag{27}$$

In general when delay and size of the logic block is significant compared to the contributions from the pipeline register the value of M_{opt} is clearly larger than 1[2].

7 Conclusions

The techniques of pipelining and parallel processing have been discussed. Which technique to employ in a specific design depends on factors such as functionality, chip area, power consumption and complexity of the control logic. Up to certain limit pipelinging provides significant performance gains with little increase in chip area. It also reduces glitching in the circuit. Throughput beyond that acheivable by pipelining can be attained by parallel architectures. For parallel architectures the throughput scales almost linearly with chip area.

References

[1] K.K. Parhi. *VLSI digital signal processing systems: Design and Implementation*, chapter 3. J. Wiley & Sons, 1999.

[2] P. Pirsch. *Arcitectures for digital signal processing*, chapter 4. J. Wiley & Sons, 1998.

[3] K.K. Parhi and D.G. Messerschmitt. Pipeline interleaving and parallelism in recursive digital filters – part I:pipelining using scattered look-ahead and decomposition. *IEEE Transactions on acoustics, speech and signal processing*, 37(7), July 1989.

[4] K.K. Parhi and D.G. Messerschmitt. Pipeline interleaving and parallelism in recursive digital filters – part II:pipelined incremental block filtering. *IEEE Transactions on acoustics, speech and signal processing*, 37(7), July 1989.

[5] A. Schlegel and T.G. Noll. The effects of glitches on the power dissipation of CMOScircuits. Internal report, EECS department RWTH Aachen, February 1997.

[6] A. Schlegel and T.G. Noll. Entwurfsmethoden zur verringerung der schaltaktivtät bei verlustoptimierten digitale CMOS-schaltungen. In *DSP Deutschland'95*, September 1995.