Reduction of Latency and Resource Usage in Bit-Level Pipelined Data Paths for FPGAs

P. Kollig B. M. Al-Hashimi
School of Engineering and Advanced Technology
Staffordshire University
Beaconside, Stafford ST18 0AD, U.K.
++44 1785 353367
peter.kollig@soton.sc.philips.com b.m.al-hashimi@staffs.ac.uk

ABSTRACT

Pipelining of data path structures increases the throughput rate at the expense of enlarged resource usage and latency unless architectures optimized towards specific applications are used. This paper describes a novel methodology for the design of generic bit-level pipelined data paths that have the low resource usage and latency of specifically tailored architectures but still allow the flexible trade-off between speed and resource requirements inherent in generic circuits. This is achieved through the elimination of all skew and alignment flip-flops from the data path whilst still maintaining the original pipelining scheme, hence allowing more compact structures with decreased circuit delays. The resulting low latency is beneficial in the realization of all recursive signal-processing applications and the reduced resource usage enables particularly the efficient FPGA realization of high performance signal processing functions. The design process is illustrated through the high level synthesis-based FPGA realization of a 9^{th}-order wave digital filter, demonstrating that high performance and efficient resource usage are possible. For example, the implementation of a wave digital filter with 10-bit signal word length and 6 bit coefficients using a Xilinx XC4013XL-1 device supports sample rates of 2.5 MHz**.**

Keywords

Bit-level pipelined, circuit latency, FPGA, recursive algorithms

1. INTRODUCTION

Traditionally, digital filters have been implemented as software solutions on dedicated digital signal processing (DSP) chips and are usually limited to audio frequency applications. To achieve the higher speed of operation required in many real-time DSP applications such as digital filter, FFT and DCT implementations an ASIC route is a viable and effective option. Despite its benefits, this route has some shortcomings including limited flexibility regarding design changes, layout costs and delivery time. Recently, growing gate densities and decreasing delay times of FPGAs coupled with powerful synthesis tools have provided a potential alternative to ASIC realizations. In such customized realization the critical operations involve multiplications and additions and to achieve the required performance, high throughput functional modules are needed. This demand is often satisfied by dividing the combinatorial logic of adder and multiplier modules into a number of pipeline stages with intervening registers or latches controlled by a global clock signal. Numerous examples of the resulting pipelined multipliers [1] and multiply-and-accumulators (MACs) [2], [3], [4] have been presented in the literature.

The developed pipelining schemes provide high throughput rates but result in high resource usage and large circuit latency due to the skewing and re-alignment of input and output data. Both problems have been addressed in many publications and are overcome with the design of architectures tailored towards a specific application. Since data formats at inputs and outputs of functional modules with the same pipelining scheme are compatible, the concerned operations are chained. This allows a considerable reduction of average latency because skewing and re-alignment of data is required only at the circuit input and output. It has been shown that this technique is particularly useful in the design of FPGA [5] and systolic array-based [6] FIR filters and in the design of multiply, square root and divide modules [7].

Although long delay times are generally acceptable in non-recursive DSP functions, recursive DSP applications such as IIR filters suffer from a reduction in sample rate because of the large latency introduced in the feedback loop. In an attempt to overcome this problem, scattered look-ahead and decomposition techniques [8] have been utilized. This technique avoids the latency problem by unfolding the recursive computation of the filter output value to the level of pipelining required. The main disadvantage of this approach is the increase in hardware complexity, which grows logarithmically with the level of pipelining. A technique without this shortcoming utilizes a most significant bit (MSB) first computation to feed the obtained result bits immediately back into the recursive computation of the filter output [9]. Thus the latency in the feedback path is reduced to the time required until the MSB is obtained, independent of the filter word length [10]. A shortcoming of this technique is the need for a signed binary number representation (SBNR) whose redundancy increases hardware requirements and necessitates conversion logic from two's complement representation to SBNR and vice versa. An alternative approach used in the alleviation of the latency

problem exploits the short coefficient property of wave digital filters to minimizing the critical path delay within the recursive part of the filter algorithm [11], [12].

All previously described approaches have addressed the problem of reducing resource usage and shortening circuit latency by chaining the functional modules in the data path, hence resulting in application-specific architectures. Tremendous efforts have been put into the design of these architectures, however their use is restricted to the targeted application. This contrasts to the flexibility of generic data path structures with their resource sharing capabilities and the possibility of trading circuit speed for design size on functional module level. This paper describes a new methodology for the design of bit-level pipelined data path structures which exploits the flexibility of generic data paths but reduces resource usage and circuit latency such that a performance comparable to application-specific architectures is achieved. The provided flexibility is best exploited with the application of high-level synthesis methodologies to generate efficient data paths in shortest possible development times. Section 2 discusses the problems inherent in the design of generic data paths and presents a novel design methodology. The implications on practical designs are discussed in section 3 through the design and FPGA realization of a 9th-order wave digital filter.

2. DESIGN METHODOLOGY

In the design of generic data path structures, a parallel input-output requirement is usually imposed to all building blocks [13], allowing the connection of all components to a data path as shown in Figure 1. The given structure consists of adder and multiplier modules for the execution of common operations in DSP functions, registers for the storage of intermediate results and an interconnection network. Note that one multiplier input is connected to a ROM that accommodates the constant coefficients used in many DSP applications. Pipelined functional modules used within this type of data path structure must comply with the parallel input-output requirements. Since pipelined functional modules process different bits of the input data in subsequent clock cycles, for example with least significant bit (LSB) first, a conversion into the bit-parallel format is needed. This is usually achieved with triangular arrays of skew and alignment flip-flops as shown in Figure 2 for the example of a pipelined parallel array multiplier, used as representative for all pipelined functional modules, ranging from simple pipelined adders [13] to complex square root modules [7].

Although the consistent use of bit-parallel timing formats allows the simple construction of data paths, the resulting structures suffer from two shortcomings. Firstly, the latency of functional modules increases the execution time of the DSP function and secondly the large number of skew and alignment flip-flops increases the design cost. The latency of an LSB-first pipelined functional module is determined with the number of pipeline stages. For example, the multiplier in Figure 2 has a latency of l_x+l_y clock cycles where l_x and l_y are the length of X and Y input vector respectively. The resource requirements of data skewing and alignment are determined with two triangular flip-flop arrays. The first array at the multiplier input consists of $\frac{1}{2}(l_y^2+l_y)$ input skew flip-flops and the second triangular array of output alignment flip-flops increases the total number of skew flip-flops to:

$$n_{skew}=\tfrac{1}{2}(l_y^2+l_y)+\tfrac{1}{2}\left((l_y+l_x)^2+(l_y+l_x)\right) \quad \textbf{(1)}$$

Assuming $l=l_x=l_y$ this simplifies to:

$$n_{skew}=\tfrac{5}{2}l^2+\tfrac{3}{2}l \quad \textbf{(2)}$$

which means that 265 skew flip-flops are needed for the realization of a 10 by 10-bit multiplier.

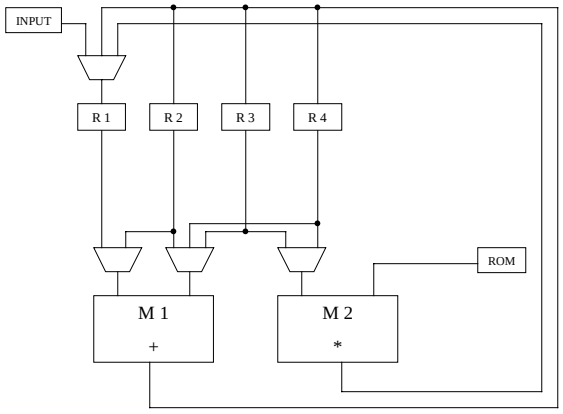


Figure 1 – Example of a generic data path for signal processing applications

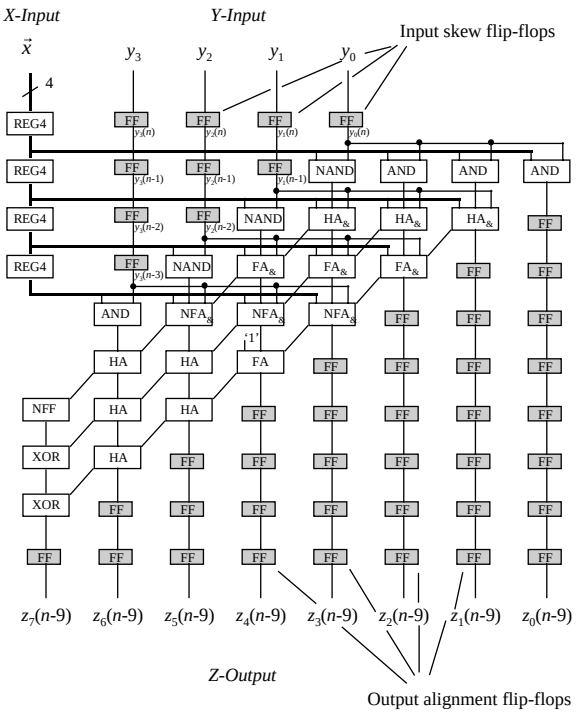


Figure 2 – Bit-level pipelined 4-bit two's complement parallel array multiplier

A solution to the latency and resource usage problem is available with the design of application-specific architectures, tailored towards specific DSP applications. In such architectures pipelined functional modules with identical interface timing are used, allowing the chaining of the data path components without the

need for the expensive and time-consuming skewing and alignment of data words as shown in Figure 3. This shows also that the triangular flip-flop arrays are only required at the inputs and outputs of the architecture to conform to the parallel input-output requirements.

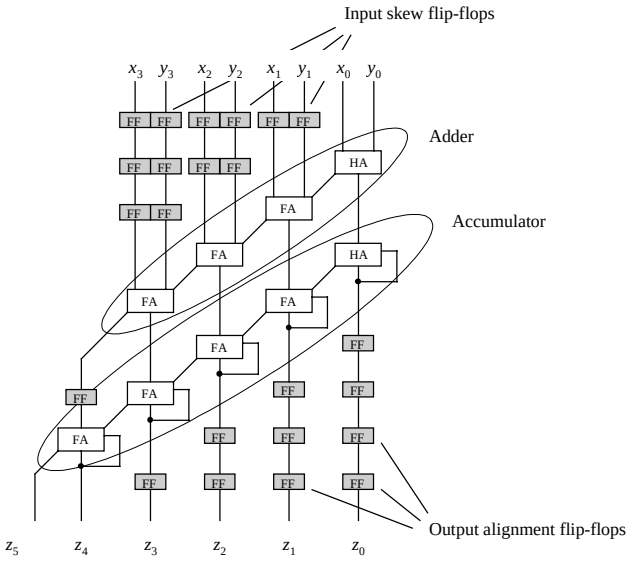


Figure 3 – Chaining of adder and accumulator modules

The simple yet powerful chaining of functional modules leads to efficient architectures tailored towards a particular application, however these architectures are not easily designed and do not provide a flexible trade-off between circuit size and speed. More efficient and flexible realizations of demanding DSP applications can be achieved with an alternative approach that eliminates the need for skew and alignment flip-flops in generic data path structures. This is shown in Figure 4 for the case of the pipelined 4-bit multiplier that performs two's complement multiplication based on the Baugh-Wooley algorithm [14]. The blocks AND, NAND and XOR represent registered primitive gates, blocks HA and FA are half adder and full adder cells respectively whilst the blocks labeled FF and NFF are flip-flops with normal and inverted outputs. Blocks $HA_{\&}$ and $FA_{\&}$ contain registered half and full adder cells with partial product term inputs.

Full resource utilization implies that the different pipeline stages process intermediate results of l_x+l_y different multiplications at any one time and the input bits y_0, y_1, … y_{l_y-1} represent bits of the multiplier operations n, n-1, … n-p+1 where $p=l_x+l_y$ represents the length of the pipeline. Conversely, the more significant bits of a particular data word n are provided in subsequent clock cycles, e.g. bit $y_0(n)$ at time t, $y_1(n)$ at t+1, … and $y_{l-1}(n)$ at t+l-1. Similarly, the result bit $z_0(n)$ is obtained at time t, bit $z_1(n)$ at t+1, etc as sketched in Figure 5*a*. To connect the output of such a pipelined functional modules in a generic data path (Figure 1) to a register, the individual result bits must be stored in subsequent clock cycles. This is achieved by splitting the p-bit register into p individual register cells, each with its own write enable signal we_m as shown in Figure 6. The timing of the individual write enable signals given in Figure 5*b* ensures that a data bit $z_m(n)$ is stored in the corresponding register cell at exactly the time t+m when it becomes available at the functional module output. For example, the write enable signal we_2 ensures that bit z_2 of the operation result is stored during the third clock cycle in its register location. Similarly, all p-bit wide multiplexers present in a data path are split into p 1-bit multiplexers and provided with individual control signals such that the functional modules obtain their input data with the required timing (Figure 6). As a result, a compact data path structure is obtained where fully pipelined functional modules are connected to registers and multiplexers without the need for skew and alignment flip-flops.

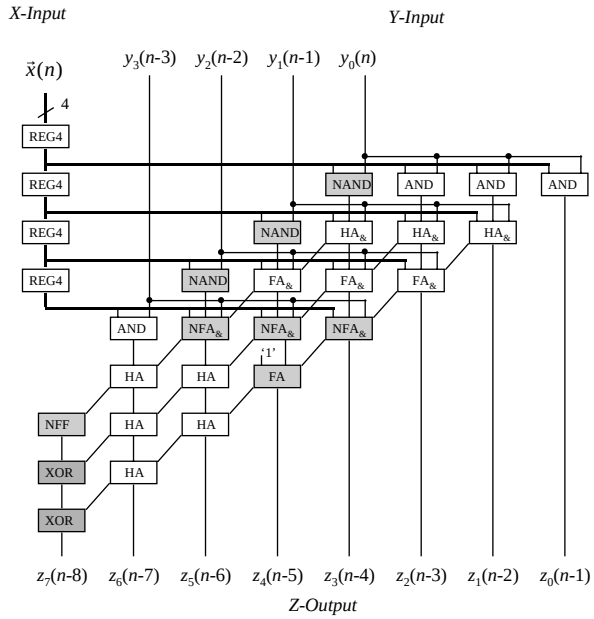


Figure 4 – Proposed bit-level pipelined two's complement parallel array multiplier

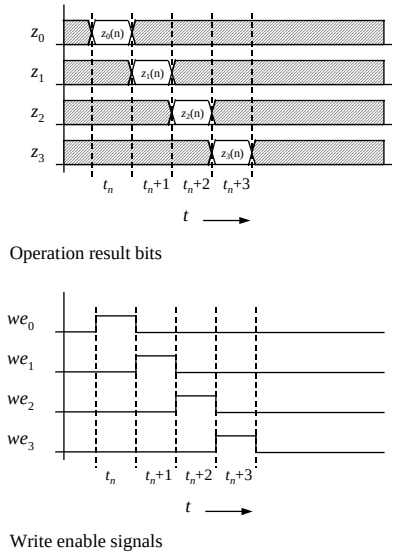


Figure 5 – Signal timing in bit-level pipelined data paths

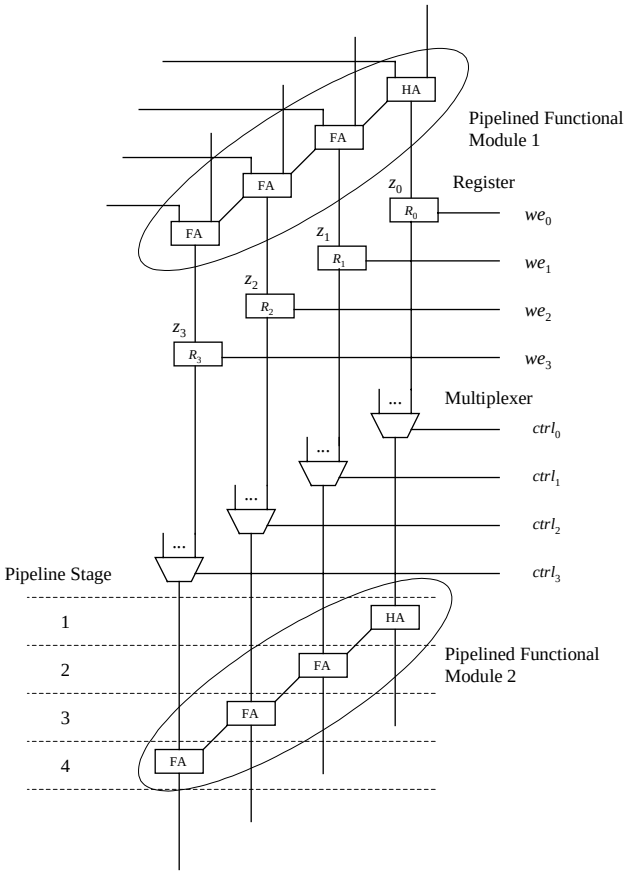


Figure 6 – Design of proposed pipelined data paths

2.1 Reduction of Circuit Latency

The construction of pipelined data paths without skew and alignment flip-flops results in optimized circuit delays. Figure 7*a* shows the timing within a data path based on LSB-first arithmetic and it can be seen that the least significant result bit of functional module 1 is stored in register R1 immediately after it becomes available whilst more significant bits are stored in subsequent cycles. Since the following operation processes the LSB first, it is possible to initiate this operation on functional module 2 even if the more significant result bits are not yet available. A comparison to a traditionally used data path structure is given in Figure 7*b* where the added skew and alignment flip-flops introduce a latency which increases the overall execution time substantially. For example, the bit-level pipelined multiplier of Figure 2 requires l_x+l_y clock cycles until the MSB is obtained and the following operation can be initiated whilst it takes only one clock cycle to obtain the LSB of the functional module in Figure 7*a*. This latency reduction is especially important in recursive signal processing algorithms where the achievable throughput rate is limited by the latency.

2.2 Reduction of Resource Usage

So far, it has been shown that the latency of a pipelined data path is reduced significantly by eliminating the need for skew and alignment flip-flops. A second advantage of this methodology is a notable saving on flip-flops, determined by:

$$s = \frac{n_{skew}}{n_{ff}} \cdot 100\% \qquad \textbf{(3)}$$

where n_{skew} is the number of skew flip-flops and n_{ff} is the total number of flip-flops. In the case of a traditional pipelined multiplier (Figure 2), it is apparent that a total of

$$n_{ff} = \tfrac{3}{2} l_x^2 - \tfrac{1}{2} l_x + 4 l_x l_y + l_y^2 \qquad \textbf{(4)}$$

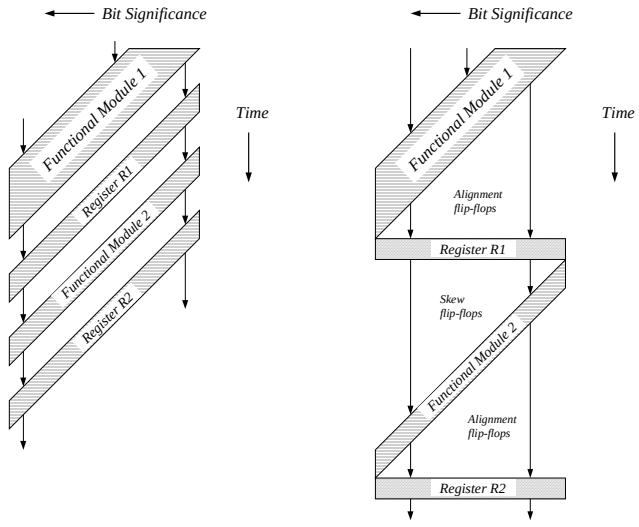
flip-flops is required to register the outputs of arithmetic blocks and for the skewing of input and output data. Assuming that $l=l_x=l_y$, Equation (4) simplifies to:

$$n_{ff} = \tfrac{13}{2} l^2 - \tfrac{1}{2} l \qquad \textbf{(5)}$$

and substituting Equations (2) and (5) into Equation (3) yields

$$s = \frac{5}{13} \cdot \frac{l^2 + \frac{3}{5} l}{l^2 - \frac{1}{13} l} \cdot 100\% \qquad \textbf{(6)}$$

This shows, for example, that a flip-flop saving of 42 % is achieved in the case of an 8-bit multiplier and 41 % for a 12-bit multiplier. Similar savings are projected for other pipelined functional modules and hence a good estimate of the savings in a complete data path structure is given with Equation (6). Clearly, this saving on flip-flop resources is achieved at the expense of an increase in control path complexity. Although the use of individual control signals for each register or multiplexer bit results in a vast amount of control signals, the increase in control path complexity is significantly less than the savings in the data path. This is because the exploitation of redundancies and don't care conditions amongst the control signals allows a substantial reduction of the number of unique control signals. The practical aspects of control path synthesis and the savings on design cost and latency are considered in the following design example.



(a) Proposed structure ***(b)*** Structure with bit-parallel timing

Figure 7 – Latency in bit-level pipelined data paths

3. DESIGN EXAMPLE

To investigate the implications of the proposed design methodology, the FPGA realization of a 9th-order wave digital filter is considered. The filter block diagram shown in Figure 8 consists of 1 first-order and 4 second-order sections, needing a total number of 10 multiplications, 19 additions and 9 subtractions for the calculation of the filter output value. The realization of this filter description based on the described data path structure allows a flexible trade-off between speed and resource requirements of a particular application. Generally, the mapping of filter description to data path structure is a complex and time-consuming task. One solution to this problem is the use of behavioral level compilers that allow the automatic mapping of algorithms into highly optimized circuit structures [15]. Here the high-level synthesis tool ARGEN [15], [16] is used to generate detailed timing and structural information. The start time as generated by ARGEN and the execution delay until the LSB of all filter operations is obtained is shown in Figure 9 assuming bit-level pipelined functional modules described in the previous section. The obtained data path structure optimized for Xilinx XC4000E FPGAs is shown in Figure 10 and it can be seen that 2 adders, 1 multiplier and 16 registers are used.

3.1 Latency of the Filter Realizations

Table 1 shows a comparison between the latency of the data path based on the proposed methodology and two alternative designs, assuming FPGAs of the Xilinx XC4000E family. The given latency represents the number of clock cycles until the LSB of the operation result is obtained, allowing the start of the next recursive algorithm execution. While this latency determines the filter throughput rate, an additional delay is apparent until the MSB of the filter output is obtained. Design alternative I uses a pipelined multiplier with skew flip-flops and a combinatorial adder based on the Xilinx fast carry logic. Design alternative II makes use of skew flip-flops within all functional modules. It is assumed that the pipelined multiplier requires 9 clock cycles to generate the first result bit and the parallel adder requires 3 clock cycles. This includes the data transfer from source register to functional module and back to the destination register. Multiplier and adder with skew and alignment flip-flops have an execution time of 18 clock cycles. Table 1 shows that the data path based on the proposed methodology has the lowest latency whilst design II has the highest. Although design II may not be the most appropriate solution in the considered filter realization, it still gives a good indication of the latency increase in designs with many pipelined operations in the critical path as is the case with commonly used compression and decoding algorithms. In the considered example, the proposed methodology leads to 35 % to 70 % faster designs when compared to the two design alternatives. Similar results in terms of latency reduction have been achieved in the realization of other signal processing functions, including a 32-point fast discrete cosine transform.

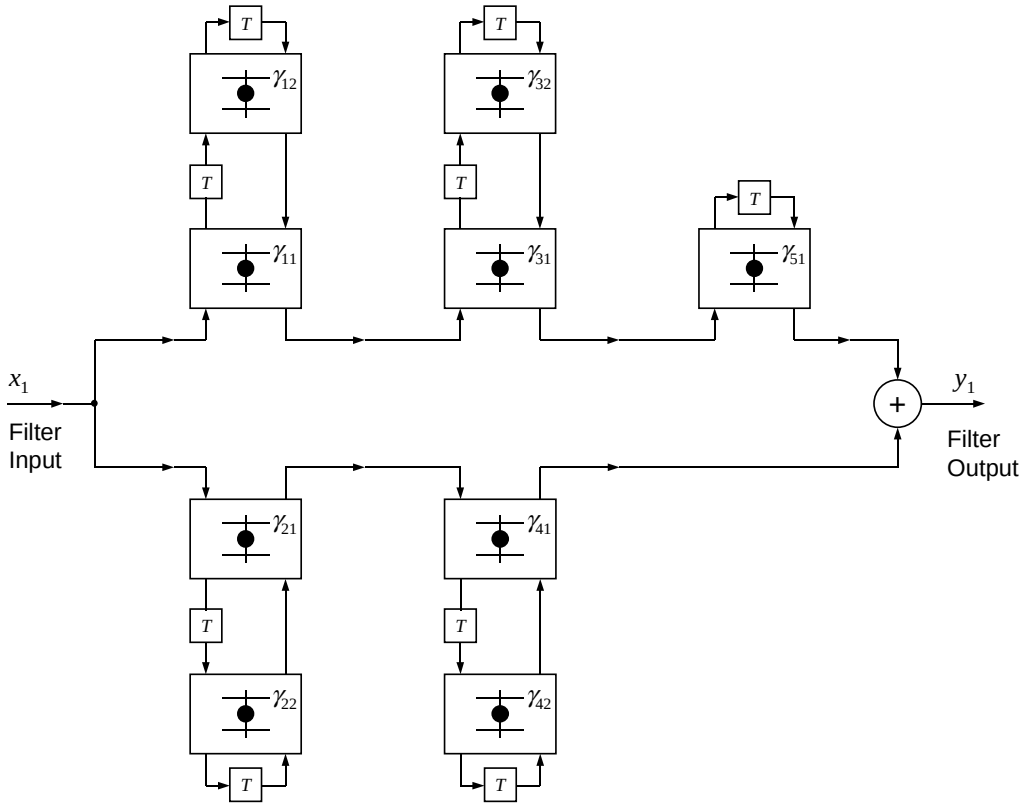


Figure 8 – 9th-order wave digital filter

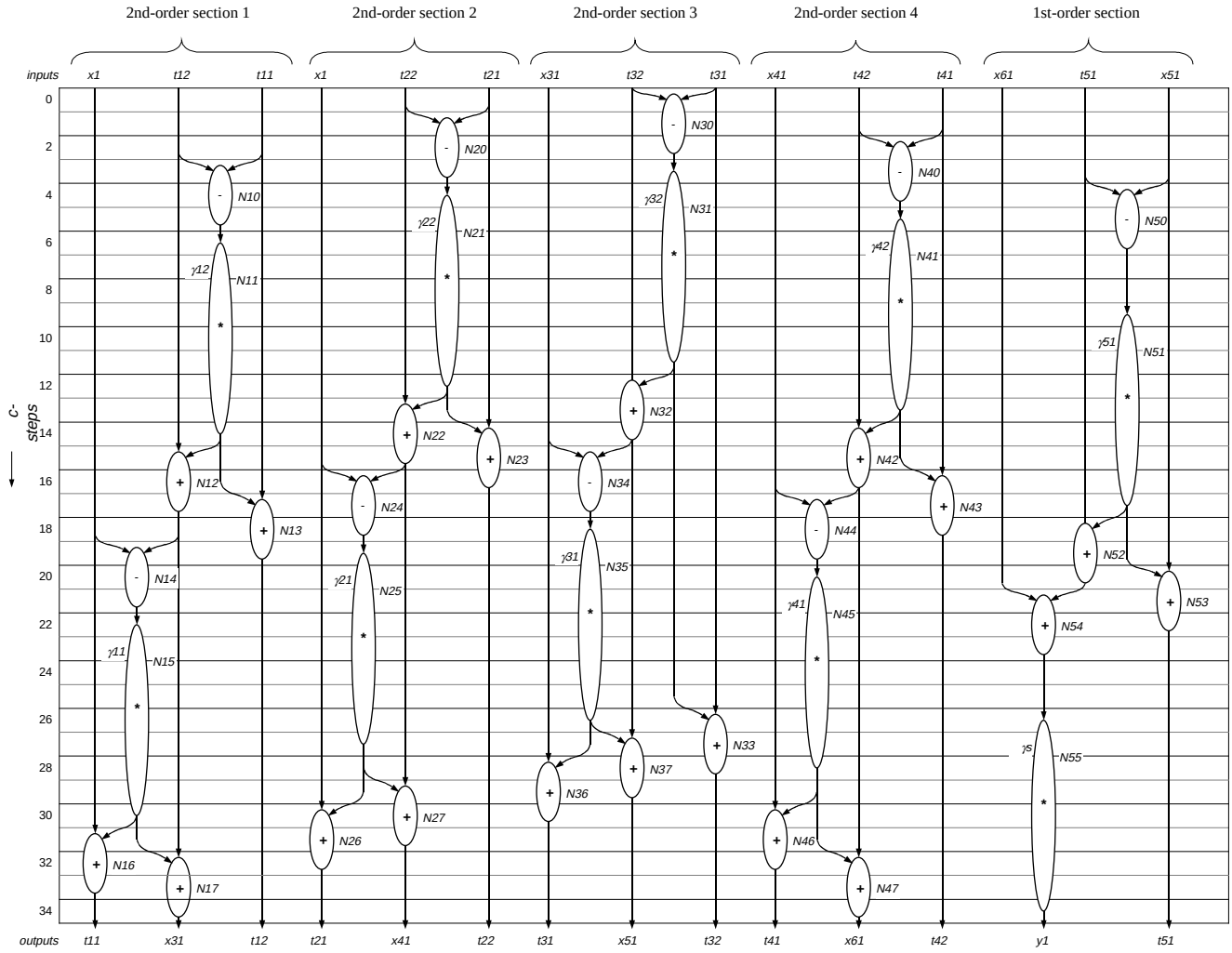


Figure 9 – Schedule of the 9th-order wave digital filter

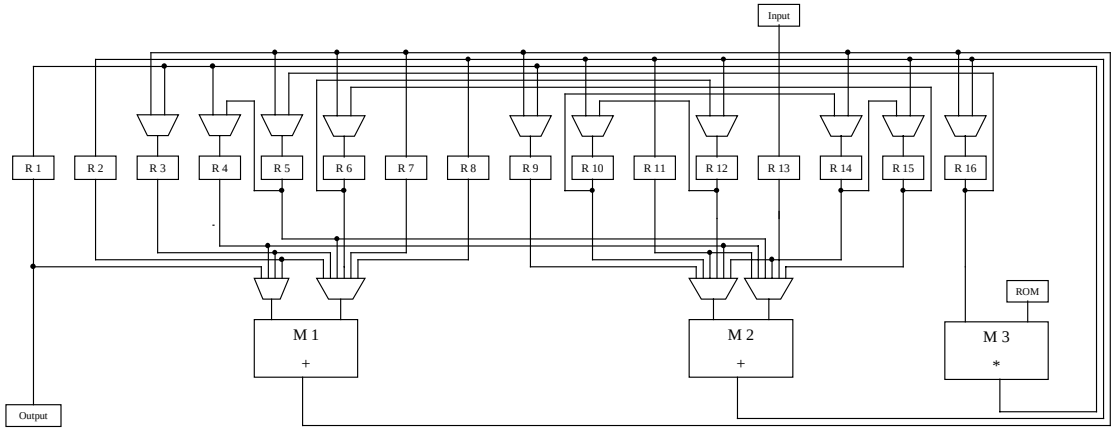


Figure 10 – Data path of 9th-order wave digital filter realisation

To allow a comparison of the proposed design methodology to previously published solutions to the latency problem, a wave digital filter two-port adapter is considered. Here the delay of an adder is assumed to be 1 clock cycle and the multiplier delay is l_x clock cycles where l_x is the coefficient length. Table 2 sets the two-port equation execution times for the proposed method, fully pipelined architectures [17], MSB first approaches [18] and systolic architectures [12] in contrast. Clearly, it can be seen that the latency achieved with the proposed method compares favorably to the other approaches. Note that the MSB first approach is independent of the coefficient length, which means that it provides better solutions in applications with long coefficients. However, this is not of practical relevance in wave digital filter realizations because commonly used coefficient word lengths are usually less than 8 bit.

Structure	Adder Delay	Multiplier Delay	Overall Latency
Proposed	3	9	35
Alternative I [1]	3	18	53
Alternative II [2]	18	18	113

[1] Multiplier with skew and alignment flip-flops, parallel adder.
[2] All functional modules with skew and alignment flip-flops.

Table 1 – Comparison of functional module delay and overall latency of three different data path structures

Architecture	Latency for coefficient length l_x	Latency for $l_x = 6$
Proposed	l_x +2	8
Pipelined [17]	l_x +2	8
MSB first [18]	9	9
Systolic [12]	2 l_x -1	11

Table 2 – Comparison of the wave digital filter two-port equation execution time

3.2 Resource Usage

The resource usage of the 9th-order wave digital filter realization based on the proposed structure and the two alternative designs is compared in Table 3 assuming devices of the Xilinx XC4000E family. This shows the size of data path, control path and complete design in terms of configurable logic blocks (CLBs) and it can be seen that the proposed methodology leads to the realization with the lowest cost. Since individual control signals are used for each register or multiplexer bit, the proposed design methodology requires 624 control signals for the considered data path. This amount of control signals can be reduced substantially by exploiting don't care conditions and redundancies amongst the signals. For example, the control signals of the multiplexers connected to registers R4, R5, R6, R12, R14 and R15 can be generated from a single source. As a result of such efforts, the control path size increases only moderately from 38 CLBs in a bit-parallel data path to 96 CLBs.

A significant cost saving is achieved in the data path area with the proposed design methodology (Table 3). Compared to design alternatives I and II, 191 and 1007 skew and alignment flip-flops can be saved. This saving is especially important in the case of FPGA realizations where skew flip-flops leave the associated combinatorial logic unused and hence lead to poor resource utilization. Overall it can be seen that the decrease in data path cost overweighs the complexity increase in the control path, enabling savings between 10 % and 50 % when using the proposed design methodology.

Although only FPGA realizations have been considered in this paper, the design methodology is equally applicable to ASIC technology. However, the reduction of circuit size is more beneficial in FPGA based realizations because of the limited resources and the inflexible grouping of flip-flops and combinatorial logic into CLBs.

Structure	Data path size (CLB)	Control path size (CLB)	Total size (CLB)
Proposed	335	96	476
Alternative I [2]	430	38	513
Alternative II [3]	838	38	876

[1] Hardware requirements in terms of CLBs.
[2] Multiplier with skew and alignment flip-flops.
[3] All functional modules with skew and alignment flip-flops.

Table 3 – Comparison of the design size of three different data path structures

3.3 Discussion

FPGA implementation using a Xilinx XC4013E has shown that operational speeds of more than 87.5 MHz are possible. Based on a schedule length of 35 clock cycles (see Figure 9), a filter sample rate of 2.5 MHz is achieved. Higher speeds of operation were not achieved because of the high FPGA utilization and the wiring complexity of the design.

As already indicated earlier, the described design method allows to trade area savings for an increase in control path complexity. In generic data paths (Figure 1) there is a large amount of control signals required for the switching of multiplexers and storing of operation results. It has been found that the amount of control overhead depends largely on the number of registers present in the design because control sharing amongst register control signals is not easily possible. This is quite different in the case of multiplexers where inactive periods result in don't care situations that simplify the control sharing of multiplexers substantially. During the construction of a data path it is thus important to find a trade-off between cost of functional modules, registers, multiplexers and control path. Such a complex trade-off can only be evaluated by means of high-level synthesis tools. At this point, it should be noted that an alternative approach for the reduction of the control path complexity if possible if various pipeline stages are grouped into a single stage. In this case, throughput rate can be traded for control path complexity.

The demonstrated filter implementation does not utilize the fast carry chains or CLB RAM found on Xilinx FPGAs. This is however not a limitation of the used design style but shows rather the general applicability of the design methodology. In the case that the fast carry logic of Xilinx FPGAs is used, it is possible to consider carry bits within the pipeline stage where they are

generated and don't have to be distributed to the next pipeline stage. The speed of this design realization depends then on the speed of the fast carry logic and the length of the carry chain. This contrasts to the described approach where the performance is determined by the CLB speed and wiring delay.

For two reasons the registers of the design have been implemented in CLB flip-flops instead of utilizing the CLB RAM capability. Firstly, the use of the RAM structures is only efficient in the case that common control signals are used for a large number of bits. Secondly, the described design methodology allows the grouping of the register flip-flops together with the register input multiplexers into a CLB, hence maintaining optimal CLB optimization.

4. CONCLUSIONS

This paper has described a new design methodology for bit-level pipelined data paths, providing a solution to the problems of high resource usage and limited flexibility inherent in previously reported approaches. Elimination of all skew and alignment flip-flops in the data path reduces the resource usage notably and more important, decreases the circuit latency to an extent that was previously only achievable with specifically tailored architectures. Although the proposed method is applicable to both, ASIC and FPGA technology, it is particularly effective in FPGA based design realizations due to the significant reduction in resource usage. This allows the realization of high performance signal processing functions on FPGAs. The implications of the proposed design methodology on the resulting data path structure have been illustrated through the high level synthesis-based FPGA realization of a 9^{th}-order wave digital filter and it has been shown that high throughput rates are achievable on medium-sized FPGA devices. For example, the described filter with 10-bit signal word length and 6 bit coefficients allows sample rates of 2.5 MHz using a highly utilized a Xilinx XC4013XL-1 device.

5. REFERENCES

[1] Hatamian, M., Cash, G.L.: 'A 70 MHz 8-bit parallel pipelined multiplier in 2.5-µm CMOS,' *IEEE Journal of Solid-State Circuits*, 1986, **21** (4) pp. 505-513

[2] Lu, F., Samueli, H.: 'A 200-MHz CMOS pipelined multiplier-accumulator using a quasi-domino dynamic full-adder cell design,' *IEEE Journal of Solid-State Circuits*, 1993, **28** (2) pp. 123-132

[3] Jou, S.-J., Chen, C.-Y., Yang, E.-C., Su, C.-C.: 'A pipelined multiplier-accumulator using a high-speed, low-power static and dynamic full adder design,' *IEEE Journal of Solid-State Circuits*, 1997, **32** (1) pp. 114-118

[4] Hatamian, M., Cash, G.L.: 'Parallel bit-level pipelined VLSI designs for high-speed signal processing,' *Proceedings of the IEEE*, 1987, **75** (9) pp. 1192-1202

[5] Kollig, P., Al-Hashimi, B.M., Abbott, K.M.: 'FPGA implementation of high performance FIR filters', Proceedings of the ISCAS, Hong Kong, 1997, pp. 2240-2243

[6] Wyrzykowski, R., Ovramenko, S.: 'Flexible systolic architecture for VLSI FIR filters,' *IEE Proceedings – E*, 1992, **139** (2) pp. 170-172

[7] McQuillan, S.E., McCanny, J.V.: 'VLSI module for high-performance multiply, square root and divide,' *IEE Proceedings – E*, 1992, **139** (6) pp. 505-510

[8] Parhi, K.K., Messerschmitt, D.G.: '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*, 1989, **37** (7) pp. 1099-1117

[9] McQuillan, S.E., McCanny, J.V.: 'A systematic methodology for the design of high performance recursive digital filters,' *IEEE Transactions on Computers*, 1995, **44** (8) pp. 971-982

[10] Woods, R.F., McCanny, J.V.: 'Design of a high-performance IIR digital filter chip,' *IEE Proceedings – E*, 1992, **139** (3) pp. 195-202

[11] Summerfield, S., Wicks, T., Lawson, S.: 'Design and VLSI architecture and implementation of wave digital filters using short signed digit coefficients,' *IEE Proceedings – Circuits Devices Systems*, 1996, **143** (5) pp. 259-266

[12] Lawson, S., Summerfield, S.: 'The design of wave digital filters using fully pipelined bit-level systolic arrays,' *Journal of VLSI Signal Processing*, 1990, **2** (1) pp. 51-64

[13] Hwang, K.: 'Computer arithmetic: principles, architecture and design' (John Wiley & Sons, Inc., 1979).

[14] Baugh, C.R., Wooley, B.A.: 'A two's complement parallel array multiplication algorithm,' *IEEE Transactions on Computers*, 1973, **C-22** (12) pp. 1045-1047

[15] Kollig, P.: 'Algorithms for scheduling, allocation and binding in high-level synthesis.' Ph.D. Dissertation, School of Engineering and Advanced Technology, Staffordshire University, April 1998

[16] Kollig, P., Al-Hashimi, B.M.: 'A new approach to simultaneous scheduling, allocation and binding in high level synthesis,' *Electronics Letters*, 1997, **33** (18) pp. 1516-1518

[17] Harris-Dowsett, D., Wicks, T., Summerfield, S.: 'A pipelining method for high speed VLSI wave digital filters,' Proceedings of the 37rd Midwest Symposium on Circuits and Systems, Lafayette, USA, 1994, pp. 1091-1094

[18] Singh, R.J., McCanny, J.V.: 'High performance VLSI architectures for wave digital filtering,' *Journal of VLSI Signal Processing*, 1992, **4** (4) pp. 269-278