

AREA EFFICIENT ARCHITECTURE FOR WAVE DIGITAL ELLIPTIC FILTER USING HIGH-LEVEL TRANSFORMATIONS

Pradnya Pravin Zode^{1,*} and Rupali Pankaj Singh²

Received: July 29, 2023; Revised: April 30, 2024; Accepted: June 07, 2024

Abstract

Digital filters are important blocks of low-frequency signal processing systems, especially biomedical signals. They are the integral algorithms of digital signal processing (DSP) systems. Multiplier, adder, and delay are the major three parts of a digital filter out of which multiplier is the maximum power and area consuming part. When implemented in a custom or semi-custom reconfigurable digital logic device, complexity will be more with multipliers than adders. Again functional units like adders and multipliers increase with the higher order of the filter. A novel architecture for wave digital elliptic filters (WDEF) using folding and retiming, which are high-level transformation techniques, is proposed. The basic elements of the filter from a data flow graph (DFG) are properly scheduled to reduce the number of multipliers and adders. The designed folded filter architecture is synthesized and implemented on the Xilinx ML605 (XC6VLX240T1FFG1156 FPGA) embedded development kit. The criteria considered for optimization are power consumption and minimum hardware area. A comparison is done between the folded architecture and the conventional architecture which shows that there is a 92.31% reduction in the used number of adders and an 87.5% reduction in the used number of multipliers. The performance comparison shows that the designed architecture is more efficient with a single adder and multiplier unit. The synthesis result shows that there is a reduction of 85.39% for logic cells and 92.31% for DSP slices in terms of area utilization for folded and conventional structures. Again the power consumption is reduced by 49.25% for folded and conventional architectures.

Keywords: DSP architectures; folding; high-level transformation techniques; retiming

¹ Department of Electronics Engineering, Yeshwantrao Chavan College of Engineering, Nagpur, India. E-mail: pradnyazode@gmail.com

² Department of Electronics and Communication Engineering, SRM Institute of Science and Technology, Delhi NCR Campus, Ghaziabad, India. E-mail: rupal.rishi@gmail.com

* Corresponding author

DOI: <https://doi.org/10.55766/sujst-2024-04-e01955>

Suranaree J. Sci. Technol. 31(4):010312(1-12)

Introduction

Different digital signal processing (DSP) algorithms are infinitely iterated algorithms which are described by data flow graphs (DFG) that perform core computation (Sindorf and Gerez, 2000). Recent DSP algorithms are efficient and well-trained in that they can solve complex and compute-intensive problems with real-time constraints. Opt for the correct algorithm and architecture, their implementation approach, and synthesis procedure are very important for designing any dedicated architecture for DSP applications. Universally, signal processing has a rich contribution and its importance is apparent in various fields (Oppenheim and Schaffer, 2011). Nowadays DSP is used everywhere with very huge range of applications like mobile telephones, high definition television (HD TV), multimedia, digital cameras, biomedical instrumentations, global positioning system (GPS), digital radio, speech recognition, and many more (Proakis, 2001). The prominence of the DSP performances is growing fast with the latest very large-scale integration (VLSI) technology. DSP systems working on real-time data can be easily implemented with VLSI technology (Parhi, 2007).

The performance of the DSP systems can be boosted by high-level architecture transformation techniques. Different high-level transformation techniques require efficient implementation like parallel processing, pipelining, retiming, folding, unfolding, and interleaving (Arslan *et al.*, 1996). These high-level architecture transformation techniques are used to design any DSP algorithm with a trade-off between important parameters such as area, speed, and power (Parhi, 2007). Creating a DSP algorithm gets customized with high-level transformations, but the original functionality is also maintained (Parhi, 2012). Area, power consumption, and speed are very critical features for complex systems designed for the consumer electronic market. Area reduction parameters can be achieved by folding transformation. The performance criteria of hardware design can be easily managed by the retiming technique. Adders and multipliers are the important controlling elements of any DSP architecture. Folding transformation is used to find out analytically when the multiple algorithmic operations such as addition and multiplication operations can be time multiplexed to a single functional unit as pipelined multiplier and pipelined adder (Parhi, 2007). The retiming transformation technique is used for changing the delay element without affecting the circuit behavior (Parhi, 2007).

High-level transformation techniques like folding and retiming are explored for the

minimization of dynamic power consumption and area. DSP architecture needed optimization for VLSI implementation. Digital filters are a very important part of any DSP system. Efficient architecture can be designed using high-level transformations considering the area and power (Zode and Deshmukh, 2016). Wave digital elliptic filters (WDEF) are very suitable for improving performance for real-time systems. Area, power, and round-off noise are the main important parameters in the design of any real-time system. (Chung and Parhi, 2012). These filters are appropriate for low-power consumption applications like wireless and cellular systems. Again they can be implemented for higher-speed applications also (Chung and Parhi, 2012). The filter can be designed for radio frequency (RF) applications where a very fast transition between the frequency cycles is required. Communication (Zhu *et al.*, 2021) and biomedical engineering are prominent areas of application of elliptical filters. In several biomedical applications, it is needed to eliminate some part of frequency from the original signal but keep the other part of the frequency spectrum intact. Several noises affect the electrocardiography (ECG) signals which are generally sufficient to disturb small amplitude features of the ECG signal that are of medical use. In this case, this filter works properly by improving the signal-to-noise ratio (Bhattacharyya *et al.*, 2013), as the range of frequency for noise intersects with the required signal frequency range.

According to the latest development in VLSI technology, the DSP has become increasingly popular which demands high-speed realization of digital filters with less power consumption for biomedical use. Various real-time DSP applications need large-order digital filters. With the increase in the order of the filter, the functional units required for computations also increase. VLSI implementation of these large-order filters is a very tedious job as the complexity of implementation increases (Parhi, 2007). In this paper, a novel architecture is presented, as the folded architecture of 5th-order WDEF. The proposed WDEF can be used to process real-time digital signals in high-speed or low-power applications.

The rest of the paper is arranged as follows: In section 2 literature review regarding WDEF and high-level transformation techniques is given. In section 3 complete and detailed procedures for the designed folded architecture of the digital elliptic filter are presented. Section 4 demonstrates the simulation-based results and provides the

comparison of the original DFG & folded DFG, the 5th section discusses the complete design with the result and the last section summarizes the conclusion.

Literature Review

Wave Digital Elliptic Filter

WDEF is a modular and generalized digital filter. It is used to control the characteristics of the filters along with a unit for coefficient updation (Tan and Chia, 1998). WDEF can be very easily implemented in VLSI technology.

Exceptional stability under any nonlinear operating conditions because of overflow and round-off errors, word length with low coefficient, and high-quality dynamic range are the eminent advantages of WDEF (Tan and Chia, 1998). DFGs are very commonly used for the representation of DSP algorithms (de Groot *et al.*, 1992). A DFG is indicated by the pair nodes and edges as $D(N, E)$ where all the node $N_i \in N$ ($i = 1 \dots n$) represents a multiplication or additional operation that is to be executed, and the edge $e_{ij} \in E$ signifies communication within the nodes N_i and N_j (Abdel-Kader, 2008). Again, each edge of the DFG should not have any negative number of delays on it (Parhi, 2007). The DFG of any DSP algorithm is the integral element of a high-level transformation technique (Memik *et al.*, 2005). The DFG for 5th-order WDEF is presented in Figure 1 (Parhi, 2007). A total of 34 nodes are present in the DFG. Out of which 8 nodes will perform multiplication operations and the remaining 26 nodes will perform addition operations. The final folded architecture is also presented as a DFG which can be easily implemented in VLSI technology (Parhi, 2007).

Folding and Retiming

In DSP systems control circuits are analyzed using folding transformation techniques. Operations

required for various algorithms like addition and multiplication are multiplexed for area optimization or used as a single functional unit by pipeline method in adder and multiplier (Parhi, 2007). The retiming transformation technique is used to adjust the position of delay elements working as a register in hardware for filters without affecting the behavior and characteristics of the system (Leiserson and Saxe, 1991; Parhi, 2007). These transformation techniques were applied to the DFG shown in Figure 1. For the proper folded implementation of any DSP algorithm, it is important to map each operation of the DFG on the hardware unit and at a specific instant (Parhi, 1995). In 1992, Parhi and others (Parhi *et al.*, 1992) described in detail the example of the 2nd order infinite impulse response (IIR) filter structure presented by DFG using a systematic folding transformation technique to fold any DSP algorithm. The folded DFG contains only 1 adder and 1 multiplier. In 1998, Denk and Parhi (1998) proposed a novel multi-rate folding transformation tool for pipelined VLSI architectures and systematic synthesis of control circuits. These are implemented for multi-rate algorithms of third-order wavelet filters and three-level orthogonal discrete wavelet transform analysis filter banks. It was concluded that different filter bank structures can be designed using multirate folding. In 1992, Parhi (1992) used lifetime analysis and forward-backwards register allocation technique and showed the systematic synthesis of DSP data format converter architectures for hardware implementation with the minimum number of registers. But the actual silicon area required for hardware implementation consists of registers, adders, multipliers as well as their interconnects. The limitation of this paper was that they were not sure that by reducing the number of registers the actual silicon area will be going to reduce or not. Register minimization and conventional methods for implementing fast Fourier transform (FFT) folded architectures using folding

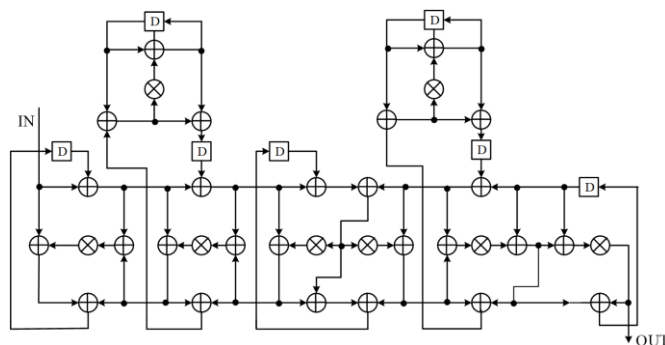


Figure 1. 5th-order Wave digital elliptic filter

transformation and register optimization techniques were presented (Ayinala *et al.*, 2012). They claimed that the power consumption was reduced by almost 50%. Only the disadvantage was that the acquired output samples were in a jumbled order. Folding and retiming applicable for iterative DFGs which contain many identical or similar substructures called hierarchical folding and hierarchical synthesis was presented by Parhi (2013). Using hierarchical synthesis it was concluded that the run time of DFG was reduced. In 2013, Karuppuswamy *et al.* (2013), applied a folding technique to the second-order digital gamma tone filter (GTF) (Parhi, 2013). They have declared that the silicon area is reduced as the number of multipliers is reduced from twenty to four and adders from twelve to four in the folded architecture. The use of high-level transformations, folding, and retiming, to protect DSP circuits with obfuscation was developed by Lao and Parhi, (2014) and Karuppuswamy *et al.* (2013). This literature presented a low-overhead solution to design DSP circuits that are obfuscated both structurally and functionally by utilizing high-level transformation techniques. In 2022, (Unnikrishnan and Parhi, 2022) designed multi-channel FFT architectures using folding and interleaving. In 2022, (Sharma and Rawat, 2020) implemented notch/peak filters using wave digital structure on field programmable gate arrays (FPGA) using the DSP tool of Xilinx system generator (XSG). From the above review analysis, it is concluded that folding and retiming high-level transformation techniques must be used to design high-speed, low-area, and power-optimized DSP algorithms. A controlled error-driven preset-input least mean square adaptive filter (EPLMS-AF) is designed by (Liu *et al.*, 2022) to improve the tracking accuracy (Tan and He, 2022). Founded a new adaptive filtering algorithm by applying the variable-step LMS algorithm altered by the logarithmic function via the multiscale wavelet transform. In 2024, Sasikala *et al.* (2024) designed an adaptive filter using a novel adder using 4×1 MUX, 2×1 MUX, and basic gates (Ferrer *et al.*, 2024). A type of the LMS algorithm that, related to the affine projection algorithm, starts from N data vectors of the reference signal but sorts out them by using only a scalar factor that functions as a convergence step is designed.

Methods: Systematic Folding of 5th-order Wave Digital Elliptic Filter

The folding transformation technique projected in (Parhi, 2007) must be applied on a DFG of a DSP algorithm so that the folded structure can be

synthesized. The Folding transformation technique is an organized process used for designing controller units for hardware that can perform various arithmetic actions at different times by a single hardware operation (Parhi, 2007). DFG is the graphical representation of DSP algorithms (Parhi, 2007).

DFG is a link between the algorithmic explanation and the actual hardware structural design of a DSP algorithm. It will be easy to plan any DSP algorithm to the latest and specific hardware implementation. The number of actions to be done by a single functional unit is described by folding factor F, if $F = 7$ then 7 multiplication operations are to be performed by a single multiplier. The order in which these F multiplication operations will be performed within a period of F cycles is definite with a folding set (Parhi, 2007). Folding set must have maximum F entries out of which some may be null operations. The arithmetic operation in the i^{th} location inside the folding set will be performed by a single adder and multiplier. Where, i range after 0 to F-1, is performed by the arithmetic component through the period i. Folding order is denoted by i, i.e., it is the period of occurrence, where the specific node is expected to complete the operation in hardware. Different folding sets give different folded architectures. Therefore, the proper selection of folding sets is very important for properly operating the folded structure. Only appropriate folding order gives the minimum area and power architecture than others. The systematic folding technique is applied on the 5th-order WDEF shown in Figure 1.

It is considered that the adder is one-stage pipelined which means one register or delay will be there along with the adder in the final folded DFG which will store the result of the addition operation and the multiplier is two-stage pipelined which means two registers or delays will be there along with the multiplier in final folded DFG which will store the result of the multiplication operation.

The critical path for any DFG is the lengthiest track in the DFG with no latch or delay element present in the path (Parhi, 2007). Therefore, the calculation time required for one iteration of the DFG is minimal. Critical path (Parhi, 2007) is the constraint on the sampling time of any DSP algorithm. The sampling rate or sampling frequency below the critical path computation time cannot be applied to any DFG. The DFG of 5th-order WDEF with node numbers for folding transformation is shown in Figure 2. The critical path calculated for the DFG shown in Figure 2 is specified by a dashed line and is given by Equation (1).

$$t_{\text{critical}} = 3T_M + 12T_A \quad (1)$$

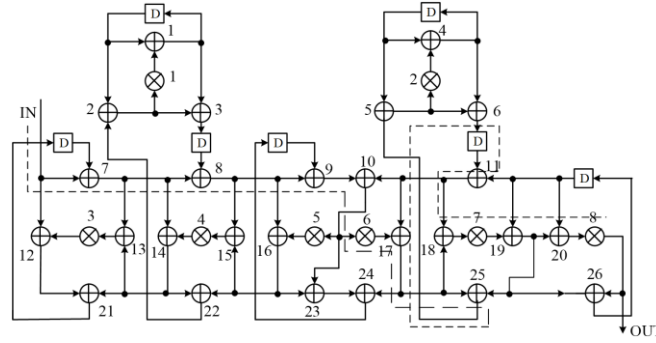


Figure 2. DFG with node numbers and critical path

Table 1. Folding sets for adder and multiplier

Folding Set Name	Operation sequence															
A1	ϕ	1	2	3	6	ϕ	7	4	8	14	22	13	ϕ	25	26	5
Folding orders	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
A2	9	10	16	ϕ	17	18	15	13	19	20	12	21	20	24	ϕ	11
Folding orders	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
M	ϕ	ϕ	5	1	6	2	ϕ	ϕ	7	4	3	ϕ	ϕ	ϕ	8	ϕ
Folding orders	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

Where $t_{critical}$ is the critical path of the DFG, T_M is the time needed for executing one iteration by multiplier and T_A is the time needed for performing one repetition by adder. Equation (1) articulates that there are 3 multipliers and 12 adders in the critical path which decides the speed of the DFG.

The folded structure of the filter is designed with a folding factor of $F = 16$ and uses the folding sets and orders given in Table 1 for adder and multiplier (Parhi, 2007).

Where symbol ϕ is a null operation i.e. arithmetic unit will not complete any action for folding order of ϕ symbol. The functional unit adder A1 performs the addition operations in sequences 1,2,3,6,7,4,8,14,22,13,25,26,5, adder A2 executes the addition operations in 9,10,16,17,18,15,13,19,23,12,21,20,24,11 sequence and functional unit multiplier M executes the multiplication operations in 5,1,6,2,7,4,3 sequence, at the respective time instance. For an edge e connecting the nodes i.e. adder or multiplier or S (source) and D (destination) with $w(e)$ delay elements, the operation of the m^{th} iteration of the node elements S and D are to be arranged at the time instants $(Fm + d)$ and $(Fm + s)$ respectively. Where s and d are the folding orders of the node elements S and D respectively. Equation (2) shows the folding values used.

$$E_F(S \rightarrow D) = Fw(e) - P_s + d - s \quad (2)$$

In Equation (2) P_s is the number of pipelined stages in the hardware unit that carry out an operation of the nodes (Parhi, 2007). There are a total of 54 edges in Figure 2.

The folding equation for all 54 edges of Figure 2 is written in Equation (3).

- DF (A1 $\rightarrow$ A3) = $16(0) - 1 + 3 - 1 = 1$
- DF (A2 $\rightarrow$ A3) = $16(0) - 1 + 3 - 2 = 0$
- DF (A2 $\rightarrow$ M1) = $16(0) - 1 + 3 - 2 = 0$
- DF (M1 $\rightarrow$ A1) = $16(0) - 2 + 1 - 3 = -4$
- DF (A3 $\rightarrow$ A8) = $16(1) - 1 + 10 - 3 = 22$
- DF (A4 $\rightarrow$ A6) = $16(0) - 1 + 5 - 8 = -4$
- DF (A5 $\rightarrow$ A6) = $16(0) - 1 + 5 - 17 = -13$
- DF (A5 $\rightarrow$ M2) = $16(0) - 1 + 5 - 17 = -13$
- DF (M2 $\rightarrow$ A4) = $16(0) - 2 + 8 - 5 = 1$
- DF (A6 $\rightarrow$ A11) = $16(1) - 1 + 17 - 5 = 27$
- DF (A7 $\rightarrow$ A8) = $16(0) - 1 + 10 - 7 = 2$
- DF (A7 $\rightarrow$ A13) = $16(0) - 1 + 13 - 7 = 5$
- DF (A7 $\rightarrow$ A14) = $16(0) - 1 + 11 - 7 = 3$
- DF (A8 $\rightarrow$ A9) = $16(0) - 1 + 0 - 10 = -11$
- DF (A8 $\rightarrow$ A15) = $16(0) - 1 + 8 - 10 = -3$
- DF (A8 $\rightarrow$ A16) = $16(0) - 1 + 4 - 10 = -7$
- DF (A9 $\rightarrow$ A10) = $16(0) - 1 + 1 - 0 = 0$
- DF (A10 $\rightarrow$ A23) = $16(0) - 1 + 11 - 1 = 9$

- DF (A10 → M5) = $16(0) - 1 + 2 - 1 = 0$
- DF (A10 → M6) = $16(0) - 1 + 4 - 1 = 2$
- DF (M5 → A16) = $16(0) - 2 + 4 - 2 = 0$
- DF (M6 → A17) = $16(0) - 2 + 6 - 4 = 0$
- DF (A11 → A10) = $16(0) - 1 + 1 - 17 = -17$
- DF (A11 → A17) = $16(0) - 1 + 6 - 17 = -12$
- DF (A11 → A18) = $16(0) - 1 + 7 - 17 = -11$
- DF (A12 → A21) = $16(0) - 1 + 13 - 12 = 0$
- DF (A13 → M3) = $16(0) - 1 + 10 - 9 = 0$
- DF (M3 → A12) = $16(0) - 2 + 12 - 10 = 0$
- DF (A14 → A22) = $16(0) - 1 + 12 - 11 = 0$
- DF (A14 → A21) = $16(0) - 1 + 13 - 11 = 1$
- DF (A14 → A13) = $16(0) - 1 + 13 - 11 = 1$
- DF (A15 → M4) = $16(0) - 1 + 9 - 8 = 0$
- DF (M4 → A14) = $16(0) - 2 + 11 - 9 = 0$
- DF (A16 → A15) = $16(0) - 1 + 8 - 4 = 3$
- DF (A16 → A22) = $16(0) - 1 + 12 - 4 = 7$
- DF (A16 → A23) = $16(0) - 1 + 11 - 4 = 6$
- DF (A17 → A18) = $16(0) - 1 + 7 - 6 = 0$
- DF (A17 → A24) = $16(0) - 1 + 15 - 6 = 8$
- DF (A17 → A25) = $16(0) - 1 + 15 - 6 = 8$
- DF (A18 → M7) = $16(0) - 1 + 8 - 7 = 0$
- DF (M7 → A19) = $16(0) - 2 + 10 - 8 = 0$
- DF (A19 → A20) = $16(0) - 1 + 14 - 10 = 3$
- DF (A19 → A24) = $16(0) - 1 + 15 - 10 = 4$
- DF (A19 → A25) = $16(0) - 1 + 15 - 10 = 4$

- DF (A20 → M8) = $16(0) - 1 + 15 - 14 = 0$
- DF (M8 → A26) = $16(0) - 2 + 16 - 14 = 0$
- DF (A21 → A7) = $16(1) - 1 + 7 - 13 = 9$
- DF (A22 → A11) = $16(0) - 1 + 17 - 12 = 4$
- DF (A23 → A24) = $16(0) - 1 + 15 - 11 = 3$
- DF (A24 → A9) = $16(1) - 1 + 0 - 15 = 0$
- DF (A25 → A5) = $16(0) - 1 + 17 - 15 = 1$
- DF (A26 → A11) = $16(1) - 1 + 17 - 16 = 16$
- DF (A26 → A19) = $16(1) - 1 + 10 - 16 = 9$
- DF (A26 → A20) = $16(1) - 1 + 14 - 16 = 13$ (3)

Retiming for Folding

According to the folding transformation technique stated in (Parhi, 2007) for proper folding, no edge in the DFG should have a negative value of delay on it. The answer to Equation 3 gives the number of delays on the edge from source to destination. From Equation (3), some edges have negative delays. Retiming (Lao and Parhi, 2015) for folding is to be performed on the filter structure shown in Figure 2 to get all edge values for positive delay. By retiming the transformation technique positions of delay elements can be altered easily in any DFG without interacting with the behavior and input-output characteristics of the DFG (Parhi, 2007).

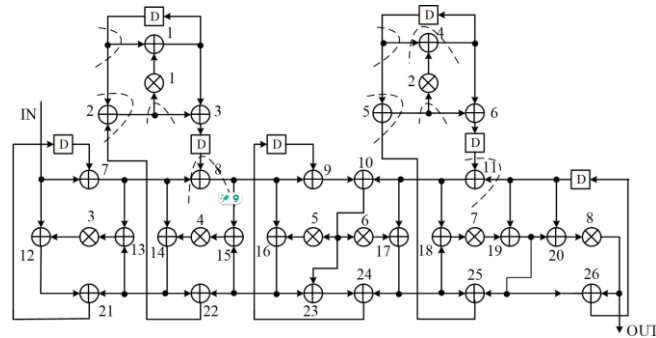


Figure 3. DFG of Wave digital elliptic filter with cutest

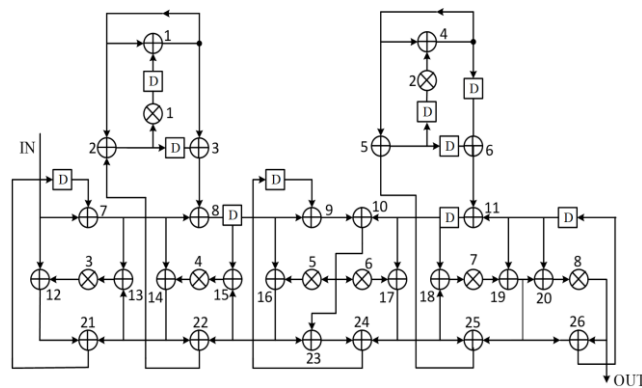


Figure 4. Retimed DFG of Wave digital elliptic filter

A special case of retiming called cutset retiming is used here; in this, a cutset is applied to the edges with DFG in a forward direction. A cutset divides the DFG into 2 parts. Then one delay element is placed on every edge under the cutset. Figures 3 and 4 show the DFG with cutset and after cutset retiming.

The folding equation has to be rewritten for retimed DFG as the number of delays on the edges under the cutset will get changed.

Equation 4 shows the folding Equation for retimed DFG using Equation (2) rewritten

- DF (A1 → A3) = $16(0) - 1 + 3 - 1 = 1$
- DF (A2 → A3) = $16(1) - 1 + 3 - 2 = 16$
- DF (A2 → M1) = $16(0) - 1 + 3 - 2 = 0$
- DF (M1 → A1) = $16(1) - 2 + 1 - 3 = 12$
- DF (A3 → A8) = $16(0) - 1 + 10 - 3 = 6$
- DF (A4 → A6) = $16(1) - 1 + 5 - 8 = 12$
- DF (A5 → A6) = $16(1) - 1 + 5 - 17 = 3$
- DF (A5 → M2) = $16(1) - 1 + 5 - 17 = 3$
- DF (M2 → A4) = $16(0) - 2 + 8 - 5 = 1$
- DF (A6 → A11) = $16(0) - 1 + 16 - 5 = 10$
- DF (A7 → A8) = $16(0) - 1 + 10 - 7 = 2$
- DF (A7 → A13) = $16(0) - 1 + 13 - 7 = 5$
- DF (A7 → A14) = $16(0) - 1 + 11 - 7 = 3$
- DF (A8 → A9) = $16(1) - 1 + 0 - 10 = 5$
- DF (A8 → A15) = $16(1) - 1 + 8 - 10 = 13$
- DF (A8 → A16) = $16(1) - 1 + 4 - 10 = 9$
- DF (A9 → A10) = $16(0) - 1 + 1 - 0 = 0$
- DF (A10 → A23) = $16(0) - 1 + 11 - 1 = 9$
- DF (A10 → M5) = $16(0) - 1 + 2 - 1 = 0$
- DF (A10 → M6) = $16(0) - 1 + 4 - 1 = 2$
- DF (M5 → A16) = $16(0) - 2 + 4 - 2 = 0$
- DF (M6 → A17) = $16(0) - 2 + 6 - 4 = 0$
- DF (A11 → A10) = $16(1) - 1 + 1 - 16 = 0$
- DF (A11 → A17) = $16(1) - 1 + 6 - 16 = 5$
- DF (A11 → A18) = $16(1) - 1 + 7 - 16 = 6$
- DF (A12 → A21) = $16(0) - 1 + 13 - 12 = 0$
- DF (A13 → M3) = $16(0) - 1 + 10 - 9 = 0$
- DF (M3 → A12) = $16(0) - 2 + 12 - 10 = 0$
- DF (A14 → A22) = $16(0) - 1 + 12 - 11 = 0$
- DF (A14 → A21) = $16(0) - 1 + 13 - 11 = 1$
- DF (A14 → A13) = $16(0) - 1 + 13 - 11 = 1$
- DF (A15 → M4) = $16(0) - 1 + 9 - 8 = 0$
- DF (M4 → A14) = $16(0) - 2 + 11 - 9 = 0$
- DF (A16 → A15) = $16(0) - 1 + 8 - 4 = 3$
- DF (A16 → A22) = $16(0) - 1 + 12 - 4 = 7$
- DF (A16 → A23) = $16(0) - 1 + 11 - 4 = 6$
- DF (A17 → A18) = $16(0) - 1 + 7 - 6 = 0$
- DF (A17 → A24) = $16(0) - 1 + 15 - 6 = 8$
- DF (A17 → A25) = $16(0) - 1 + 15 - 6 = 8$
- DF (A18 → M7) = $16(0) - 1 + 8 - 7 = 0$
- DF (M7 → A19) = $16(0) - 2 + 10 - 8 = 0$
- DF (A19 → A20) = $16(0) - 1 + 14 - 10 = 3$
- DF (A19 → A24) = $16(0) - 1 + 15 - 10 = 4$
- DF (A19 → A25) = $16(0) - 1 + 15 - 10 = 4$

- DF (A20 → M8) = $16(0) - 1 + 15 - 14 = 0$
- DF (M8 → A26) = $16(0) - 2 + 16 - 14 = 0$
- DF (A21 → A7) = $16(1) - 1 + 7 - 13 = 9$
- DF (A22 → A11) = $16(0) - 1 + 17 - 12 = 4$
- DF (A23 → A24) = $16(0) - 1 + 15 - 11 = 3$
- DF (A24 → A9) = $16(1) - 1 + 0 - 15 = 0$
- DF (A25 → A5) = $16(0) - 1 + 17 - 15 = 1$
- DF (A26 → A11) = $16(1) - 1 + 17 - 16 = 16$
- DF (A26 → A19) = $16(1) - 1 + 10 - 16 = 9$
- DF (A26 → A20) = $16(1) - 1 + 14 - 16 = 13$ (4)

As per the folding transformation technique defined by Parhi (2007) the greatest value of the answer from Equation (4) of the folding equation provides the greatest number of registers compulsory for the design of the final folded structure. In this case, the greatest value of the answer is 16 i.e. the final folded structure requires a maximum of 16 registers for the implementation. The register optimization technique is explained by Parhi to optimize the number of registers of the filter (Parhi, 2007) so that the needed silicon area for the hardware implementation will be less for the registers.

Register Minimization Technique

The folding transformation technique is applied to the proposed 5th-order WDEF. The DFG of the filter is shown in Figure 4. The number of functional units i.e. adders and multipliers are reduced in folded filter architecture. However, the number of registers in the folded architecture has increased from 10 to 16. A technique for the reduction of an excessive number of registers in the folded architecture is proposed by (Parhi, 1994, 2007).

Lifetime Analysis

To compute the least number of registers required for the proposed construction of folded DFG, lifetime analysis is performed. Lifetime analysis is performed on the DSP algorithm in hardware for minimization of registers (Parhi, 2007). In this, the lifetime of every node of the DFG is considered. In the lifetime analysis, the node can be an adder or multiplier. As shown in Table 2, A₁ to A₂₆ are adders, and M₁ to M₈ are the multiplier nodes. The lifetime of a node starts from the clock cycle in which the input data is applied to a specific node and ends at the clock cycle from which the data is outputted from the node. Table 2 gives the lifetimes for nodes of the filter and Figure 4 shows the Retimed DFG of 5th-order WDEF.

Lifetime Chart

The lifetime chart is drawn from the lifetimes of the nodes given in Table 2 which is shown in Figure 5. The horizontal lines show the clock cycles

Table 2. The lifetime table of DFG is shown in Figure 4

Node	Tinput	Toutput	Tinput→Toutput
A1	2	3	2→3
A2	3	19	3→19
A3	4	10	4→10
A4	9	21	9→21
A5	18	21	18→21
A6	6	16	6→16
A7	8	13	8→13
A8	11	24	11→24
A9	1	1	1→1
A10	2	11	2→11
A11	18	24	18→24
A12	13	13	13→13
A13	10	10	10→10
A14	12	13	12→13
A15	9	9	9→9
A16	5	12	5→12
A17	7	15	7→15
A18	8	8	8→8
A19	11	15	11→15
A20	15	15	15→15
A21	14	23	14→23
A22	13	17	13→17
A23	12	15	12→15
A24	16	16	16→16
A25	16	17	16→17
A26	17	33	17→33
M1	5	17	5→17
M2	7	8	7→8
M3	12	12	12→12
M4	11	11	11→11
M5	4	4	4→4
M6	6	6	6→6
M7	10	10	10→10
M8	16	17	16→17

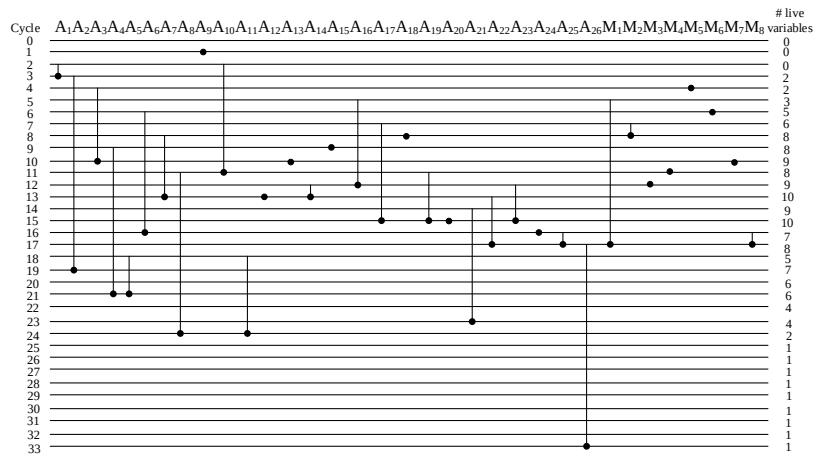
and the vertical lines are the lifetime of a node (Parhi, 2007). To design the folded DFG with the least number of registers, the numbers of live variables per clock cycle are added. The lifetime chart gives the least number of registers required for the design of the folded DFG of a filter. It also gives

the greatest number of alive variables at any clock cycle (Abdel-Kader, 2008). The least number of registers required to design the elliptic filter is the greatest number of alive variables = $\text{great}\{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10\} = 10$. The greatest value of live variables from all clock cycles in Figure 5 is 10. So the least number of registers required for the design of folded architecture are 10 only.

Forward and Backward Register Allocation Technique

For optimization of registers and data allocation in the filter structure, forward and backward register allocation technique is used. In this technique forward register allocation starts by node one by one and then the backward empty registers are allocated, hence the name “forward-backwards” (Parhi *et al.*, 1992). The ten registers used in the design are denoted as R1 to R10 in the allocation table shown in Figure 6.

For example, the lifetime of node A2 is from clock cycle 3 to 19. That is why node A2 is inputted in clock cycle 3 and moved to register R1 in clock cycle 4 and transferred to the next registers in subsequent clock cycles. As the total number of registers is 10 so A2 moved to R2 in clock cycle 5, to R3 in clock cycle 6, to R4 in clock cycle 7, to R5 in clock cycle 8, to R6 in clock cycle 9, to R7 in clock cycle 10, to R8 in clock cycle 11, to R9 in clock cycle 12, to R10 in clock cycle 13. But as node A2 needs to go up to clock cycle 19 backward allocations are performed. As no other register is free, A2 is moved back to R6 in clock cycle 14 then moved to R7 in clock cycle 15, to R8 in clock cycle 16, to R9 in clock cycle 17, to R10 in clock cycle 18, and then to R9 in clock cycle 19 which is output clock cycle for node A2. So A2 is outputted in clock cycle 19 as shown in Figure 6.

**Figure 5. The lifetime chart for DFG is shown in Figure 4**

Cycle	Input	R1	R2	R3	R4	R5	R6	R7	R8	R9	R10	Output
0												
1	A9											A9
2	A1,A10											
3	A2	A1										A1
4	M5,A3	A2	A10									M5
5	M1,A16	A3	A2	A10								
6	M6,A6	M1	A3	A2	A16	A10						M6
7	M2,A17	A6	M1	A3	A2	A16	A10					
8	A18,A7	A17	A6	M1	A3	A2	A16	A10	M2			M2,A18
9	A4,A15	A7	A17	A6	M1	A3	A2	A16	A10			A15
10	A13,M7	A4	A7	A17	A6	M1	A3	A2	A16	A10		A3,A13,M7
11	A8,A19,M4		A4	A7	A17	A6	M1		A2	A16	A10	M4,A10
12	A14,A23,M3	A8	A19	A4	A7	A17	A6	M1		A2	A16	M3,A16
13	A12,A22	A14	A8	A19	A4	A7	A17	A6	M1	A23	A2	A12,A7,A14
14	A21	A22		A8	A19	A4	A2	A17	A6	M1	A23	
15	A20	A21	A22	A23	A8	A19	A4	A2	A17	A6	M1	A20,A17,A19,A23
16	A24,A25,M8		A21	A22		A8		A4	A2	M1	A6	A24,A6
17	A26	A25	M8	A21	A22		A8		A4	A2	M1	A25,M8,A22,M1
18	A5,A11	A26		A21			A8		A4	A2		
19		A5	A26	A11		A21		A8	A2	A4		A2
20			A5	A26	A11		A21		A8	A4		
21				A5	A26	A11		A21	A4	A8		A4,A5,A21
22					A26	A11			A8			
23						A26	A11			A8		
24							A26	A11	A8			A8,A11
25								A26				
26									A26			
27										A26		
28											A26	
29												A26
30												A26
31												A26
32												A26
33												A26

Figure 6. The register allocation table for DFG is shown in Figure 5

Folded Structure

Folded structure for the 5th-order WDEF is shown in Figure 7. The structure consists of two adders with 1 register to store its 1 output and only one multiplier unit with 2 registers to store its 1 output and 10 registers. The DFG of 5th-order WDEF shown in Figure 1 has input connected to the adder unit and output taken from the multiplier unit is shown in the folded structure. One register in the folded DFG represents one delay. The folded edge $D_F (A19 \rightarrow A25) = 16(0) - 1 + 15 - 10 = 4$ for this there is an edge in the folded structure from adder to adder with 4 delays. Again this edge ends at node

A25 which has folding order 13 from the decided folding set. Therefore the folding edge is changed at the input of the adder in the folded structure at $16l + 13$ where 16 is F i.e. folding factor. As the operation needs 4 registers, data is moving from R2 to R3 to R4 and R5 and then to an adder.

Results

Conventional 5th-order WDEF architecture and its folded architecture are compared in this work. Table 3 shows the comparison between the number of

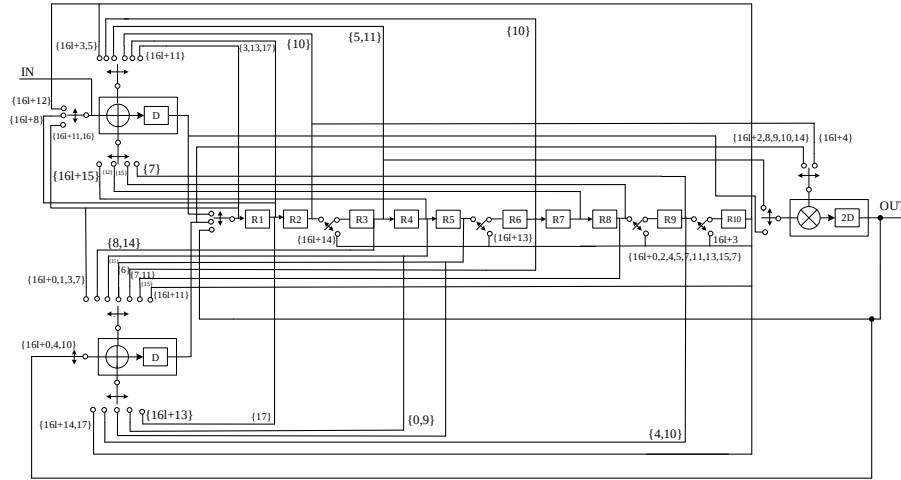


Figure 7. Folded Architecture for DFG is shown in Figure 1

basic computational units as adders and multipliers and several registers. It is clear from the table that the folded architecture takes less number of adders and multipliers as compared to conventional architecture, while the numbers of registers are increased by 3. The number of adders, multipliers, and registers are counted theoretically from the conventional and folded DFG shown in Figure 1 and 7 respectively which shows that there is a 92.31% reduction in the used number of adders as DFG in Figure 1 have 26 adders and 8 multipliers and 87.5% reduction in the used number of multipliers as Folded DFG shown in Figure 7 is designed with only 2 adders and 1 multiplier.

As the 5th-order WDEF DFG has 26 adders and 8 multipliers it needs more area for hardware implementation in FPGA. The proposed folded architecture described above, and the conventional filter structure are modelled in Xilinx using Verilog hardware descriptive language and synthesized with the XST tool. The target implementation platform is the Xilinx ML605 (XC6VLX240T1FFG1156 FPGA) embedded development kit. The total logic cells available in this device are 241,152 and DSP slices are 768.

Table 4 gives the resource utilization of the FPGA device which shows that there is an 85.39% reduction for logic cells and 92.31% for DSP slices in terms of area utilization for folded and conventional 5th-order WDEF architectures. The design achieves a throughput of 1,247 M samples/s.

Table 5 gives the synthesis result of conventional and folded 5th-order WDEF which shows that the power consumption of a folded structure is just 1.488 mW and of conventional DFG is 2.932 mW. So power consumption of folded DFG

Table 3. Comparison of original DFG & folded DFG

DFGs	No. of functional units		No. of registers
	Adders	Multipliers	
Conventional Architecture	26	8	7
Folded Architecture	2	1	10

Table 4. FPGA resources usage

Architecture	Logic cells	DSP slices	Throughput
Conventional	7,709	26	1,247 Ms/s
Folded	1,127	2	1,247 Ms/s

Table 5. Synthesis result of conventional and folded 5th-order WDEF

Parameters	Conventional	Folded
Power	2.932 mW	1.488 mW
Delay	452 ns	608 ns

is 49.25% less than the conventional one. Again the delay is 608 ns and 452 ns for folded and conventional DFGs respectively. As numbers of operations are performed by a single multiplier and two adders in the folded DFG so delay is quiet which again can be managed by proper designing of adders and multipliers.

Discussions

The folded DFG is designed using folding and retiming, the high-level transformation techniques. The DFG of conventional 5th-order WDEF has 26 adders and 8 multipliers. The final folded DFG is designed with only two adders, one multiplier,

and ten registers. As there is a significant reduction in the number of adders and multipliers, the folded DFG needs only 1,127 logic cells and 2 DSP slices when implemented in FPGA. But for the implementation of conventional 5th-order WDEF, 7,709 logic cells and 26 DSP slices were required. It is concluded that the designed folded DFG takes much less space in FPGA hardware implementation and also consumes much less power as compared to the original DFG. The only limitation of this design is that multiple addition and multiplication operations are performed by the same adder and multiplier so the time needed to compute one iteration i.e. to get one output sample is more.

Conclusions

Field-programmable devices are used for high-level optimization and custom architectures. The designed folded architecture is explained above in detail, and the conventional filter DFG of 5th-order WDEF is modeled in Xilinx using Verilog hardware descriptive language and synthesized with the XST tool. The power and speed analysis is also carried out. The designed architecture can be effectively used in high-frequency and accurate applications like communication and biomedical engineering. Experimental results show that the folded architecture of 5th-order WDEF notably minimizes the area of implementation, yet there is always a trade-off between the area and the delay. This increase in delay of filter design can be easily compensated by taking care of the proper designing of registers. The registers should be designed with minimal propagation delay and less power consumption. In addition, the fabrication cost of biomedical instruments is reduced by applying folding techniques to reduce the area of implementation of the wave elliptic filter. As a future suggestion, we will apply high-level transformation techniques on multirate filters.

References

- Abdel-Kader, R.F. (2008). Particle Swarm optimization for constrained instruction scheduling. *VLSI Design*, 2008:Article ID 930610, 7p. <https://doi.org/10.1155/2008/930610>
- Arslan, T. Erdogan, A.T., and Horrocks, D.H. (1996). Low power design for DSP: methodologies and techniques. *Microelectronics Journal*, 27(8):731-744. [https://doi.org/10.1016/0026-2692\(96\)00010-9](https://doi.org/10.1016/0026-2692(96)00010-9)
- Ayinala, M., Brown, M., and Parhi, K.K. (2012). Pipelined parallel FFT architectures via folding transformation. In: *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 20(6):1068-1081. <https://doi.org/10.1109/TVLSI.2011.2147338>
- Bhattacharyya, S.S., Deprettere, E.F., Leupers, R., Takala, J. (2013). *Handbook of Signal Processing Systems*. 2nd ed. New York, NY, USA., Springer.
- Chung, J.-G. and Parhi, K.K. (2012). *Pipelined Lattice and Wave Digital Recursive Filters*. Vol. 344. Springer Science & Business Media.
- de Groot, S.M.H., Gérez, S.H., and Herrmann, O.E. (1992). Range-chart-guided iterative data-flow graph scheduling. In: *IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications*, 39(5):351-364. <https://doi.org/10.1109/81.139286>
- Denk, T.C. and Parhi, K.K. (1998). Synthesis of folded pipelined architectures for multirate DSP algorithms. In: *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 6(4):595-607. <https://doi.org/10.1109/92.736133>
- Ferrer, M., de Diego, M., and Gonzalez, A. (2024). Low cost variable step-size LMS with maximum similarity to the affine projection algorithm. *IEEE Open Journal of Signal Processing*, 5:82-91. <https://doi.org/10.1109/OJSP.2023.3340106>
- Karuppuswamy, R., Arumugam, K., and Swathi, P.M. (2013). Folded architecture for digital gammatone filter used in speech processor of cochlear implant. *ETRI Journal*, 35(4):697-705. <https://doi.org/10.4218/etrij.13.0112.0220>
- Lao, Y. and Parhi, K.K. (2014). Protecting DSP circuits through obfuscation. In: *2014 IEEE International Symposium on Circuits and Systems (ISCAS)*, Melbourne, VIC, Australia, p. 798-801. <https://doi.org/10.1109/ISCAS.2014.6865256>
- Lao, Y. and Parhi, K.K. (2015). Obfuscating DSP circuits via high-level transformations. In: *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 23(5):819-830. <https://doi.org/10.1109/TVLSI.2014.2323976>
- Leiserson, C.E. and Saxe, J.B. (1991). Retiming synchronous circuitry. *Algorithmica*, 6:5-35 <https://doi.org/10.1007/BF01759032>
- Liu, Q., Huo, X., Liu, K., Zhao, H., and Wu, A. (2022). Error-driven tracking control design with preset-input LMS adaptive filter. In: *2022 IEEE 11th Data Driven Control and Learning Systems Conference (DDCLS)*, Chengdu, China, p. 233-237. <https://doi.org/10.1109/DDCLS55054.2022.9858413>
- Memik, S.O., Kastner, R., Bozorgzadeh, E., and Sarrafzadeh, M. (2005). A scheduling algorithm for optimization and early planning in high-level synthesis. *ACM Transactions on Design Automation of Electronic Systems*, 10(1):33-57. <https://doi.org/10.1145/1044111.1044115>
- Oppenheim, A.V. and Schaffer, R.W. (2011). *Digital Signal Processing*. 3rd ed. New Delhi: PHI Learning Private Limited.
- Parhi, K.K. (1992). Systematic synthesis of DSP data format converters using life-time analysis and forward-backward register allocation. In: *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, 39(7):423-440. <https://doi.org/10.1109/82.160168>
- Parhi, K.K. (1994). Calculation of minimum number of registers in arbitrary life time chart. In: *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, 41(6):434-436. <https://doi.org/10.1109/82.300209>
- Parhi, K.K. (1995). High-level algorithm and architecture transformations for DSP synthesis. *Journal of VLSI signal processing systems for signal, image and video technology*, 9:121-143 <https://doi.org/10.1007/BF02406474>
- Parhi, K.K. (2007) *VLSI Digital Signal Processing Systems: Design and Implementation*. John Wiley & Sons, California Technical Publishing.
- Parhi, K.K. (2012). Verifying equivalence of digital signal processing circuits. In: *2012 Conference Record of the Forty Sixth Asilomar Conference on Signals, Systems and Computers (ASILOMAR)*, Pacific Grove, CA, USA,

- p. 99-103. <https://doi.org/10.1109/ACSSC.2012.6488967>
- Parhi, K.K. (2013). Hierarchical folding and synthesis of iterative data flow graphs. In: IEEE Transactions on Circuits and Systems II: Express Briefs, 60(9):597-601. <https://doi.org/10.1109/TCSII.2013.2268658>
- Parhi, K.K., Wang, C.-Y., and Brown, A.P. (1992). Synthesis of control circuits in folded pipelined DSP architectures. IEEE Journal of Solid-State Circuits, 27(1):29-43. <https://doi.org/10.1109/4.109555>
- Proakis, J.G. (2001). Digital Signal Processing: Principles, Algorithms and Applications. Pearson Education India, 968p.
- Sasikala, S., Sivarajan, P., Sountharajan, S., Shangeetha, M., and Udhaya Agilan, K.S.D. (2024). Design of efficient adaptive LMS filter for noise reduction in ECG. In: 2024 Second International Conference on Emerging Trends in Information Technology and Engineering (ICETITE), Vellore, India, p. 1-8. <https://doi.org/10.1109/icETITE58242.2024.10493643>
- Sharma, A. and Rawat, T.K. (2020). Notch/peak filter design and its FPGA implementation through wave digital structure. In: 2020 International Conference on Radar, Antenna, Microwave, Electronics, and Telecommunications (ICRAMET), Tangerang, Indonesia, p. 57-61, <https://doi.org/10.1109/ICRAMET51080.2020.9298644>
- Sindorf, S.L. and Gerez, S.H. (2000). An integer linear programming approach to the overlapped scheduling of iterative data-flow graphs for target architectures with communication delays. In: Proceedings of the 1st PROGRESS workshop on Embedded Systems, p. 95-103.
- Tan, E.C. and Chia, C.Y. (1998). Hardware configuration for effective control of elliptic-based low-pass wave-digital filters. In: ICSP '98. 1998 Fourth International Conference on Signal Processing (Cat. No.98TH8344), Beijing, China, 1998, Vol 2:1597-1599, <https://doi.org/10.1109/ICOSP.1998.770932>
- Tan, W. and He, W. (2022). A novel variable step size adaptive filtering algorithm based on logarithmic function. In: 2022 IEEE 4th International Conference on Power, Intelligent Computing and Systems (ICPICS), Shenyang, China, p. 537-541, <https://doi.org/10.1109/ICPICS55264.2022.9873788>
- Unnikrishnan, N.K. and Parhi, K.K. (2022). Multi-channel FFT architectures designed via folding and interleaving. In: IEEE International Symposium on Circuits and Systems (ISCAS), Austin, TX, USA, 2022, p. 142-146. <https://doi.org/10.1109/ISCAS48785.2022.9937347>
- Zhu, X., Ge, Z., Yang, L., and Gómez-García, R. (2021). Millimeter-wave CMOS passive filters for 5G applications. 2021 IEEE MTT-S International Microwave Filter Workshop (IMFW), p. 198-200. <https://doi.org/10.1109/IMFW49589.2021.9642358>
- Zode, P.P. and Deshmukh, A. (2016). Folded architecture for non canonical least mean square adaptive digital filter used in echo cancellation. International Journal of VLSI Design & Communication Systems, 7(3):13-27. <https://doi.org/10.5121/vlsic.2016.7302>