International Journal of Electronics Signals and Systems

Volume 2 | Issue 2 Article 1

October 2012

A VLSI DSP DESIGN AND IMPLEMENTATION OF COMB FILTER USING UN-FOLDING METHODOLOGY

PURU GUPTA
Dept. of Electronics and Communication Engineering, Netaji Subhas Institute of Technology -Dwarka Sector 3, New Delhi, India., purugupta5000@gmail.com

TARUN KUMAR RAWAT
Dept. of Electronics and Communication Engineering, Netaji Subhas Institute of Technology -Dwarka Sector 3, New Delhi, India., tarundsp@gmail.com

A VLSI DSP DESIGN AND IMPLEMENTATION OF COMB FILTER USING UN-FOLDING METHODOLOGY

[1]PURU GUPTA & [2]TARUN KUMAR RAWAT

[1&2]Dept. of Electronics and Communication Engineering,
Netaji Subhas Institute of Technology -Dwarka Sector 3, New Delhi, India.
purugupta5000@gmail.com; tarundsp@gmail.com

Abstract: In signal processing, a comb filter adds a delayed version of a signal to itself, causing constructive and destructive interference. Comb filters are used in a variety of signal processing applications that is Cascaded Integrator-Comb filters, Audio effects, including echo, flanging, and digital waveguide synthesis and various other applications. Comb filter when implemented has lower through-put as the sample period can not be achieved equal to the iteration bound because node computation time of comb filter is larger than the iteration bound. Hence throughput remains less. This paper present the comb filter using one of the methodology needed to design custom or semi custom VLSI circuits named as Un-Folding which increases the throughput of the comb filter. Un-Folding is a transformation technique that can be applied to a DSP program to create a new program describing more than one iteration of the original program. It can unravel hidden con-currency in digital signal processing systems described by DFGs. Therefore, unfolding has been used for the sample period reduction of the comb filter for its higher throughput.

Keywords: *Data Flow Graph, Iteration, Iteration Period, Iteration Bound, Loop bound, Critical Path.*

1. INTRODUCTION

Digital signal processing (DSP) is used in numerous applications such as video compression, digital set-top box, cable modems, digital versatile disk, wireless communications, digital radio, digital still and speech processing, radar imaging etc. The field of DSP has always been driven by the advances in DSP applications and in scaled very-large-scale-integrated (VLSI) technologies. Therefore at any given time, DSP applications impose several challenges on the implementations of the DSP systems. These implementations must satisfy the enforced sampling rate constraints of the real-time DSP applications and must require less space and power consumption [3].

There are various methodologies needed to design custom or semi-custom VLSI circuits for these applications. DSP computation is different from general-purpose computation. In DSP computation, the same program is executed repetitively on an infinite time series. The non-terminating nature can be exploited to design more efficient DSP systems by exploiting the dependency of tasks both within an iteration and among multiple iterations. Furthermore, long critical paths [1][3] in DSP algorithms limit the performance of DSP systems. These algorithms need to be transformed for design of high-speed or low-area or low-power implementations. DSP Algorithms are used in various real-time applications with differ-ent sampling rate requirements that can vary from about 20KHZ in speech applications to over 500 MHz in radar and HD television applications. The compu-tation requirement of a video compression system for HDTV can range from 10 to 100 giga operations per second. The dramatically different sample rate and computation requirements necessitate different architecture considerations for implementations of DSP algorithms [3]. There are several high-level architectural transformations mentioned as follows [3]: Pipelining and Parallel processing, Retiming, Un-Folding, Folding, Systolic array design methodology.

In this paper we have presented unfolding technique to implement comb filter with higher throughput. In section II some basic definitions are covered required for understanding of this paper. Section III, IV and V presents about the unfolding and its algorithm. Section VI and VII presents the comb filter and implementation of comb filter using unfolding. Section VIII and IX presents the results.

2. BASIC DEFINITIONS

2.1 Data Flow Graphs

Many signal processing algorithms can be represented by data flow graphs (DFGs) [1], where nodes represent tasks while edges between nodes represent dependencies among tasks. Because the most time-consuming part of DFGs is the existence of loops, we should explore the parallelism found there in order to achieve lower execution time It is common that edges in DFGs are loaded with delays, which are represented by short vertical bars located on edges to represent inter-iteration dependencies between nodes. An edge e from node *u* to node *v* is denoted by (*u*, *v*) with delay *d*(*e*) indicating that the computation of nodes v at iteration j depends on the computation of node u at iteration $j - d(e)$. For a meaningful data flow graph, the total delay count should not be equal to zero. Fig. 1 is an example of a DFG where we have three nodes named A, *B, and C*. We also have three edges, (A, B), (B, C) and (C, A) connecting the nodes and indicating the consecutive order of execution. For instance, during the same iteration in the Fig. 1, node B can not start execution before node A finishes its execution. The numbers over the nodes tell us the time needed to execute each node on its own and referred to as the execution time $t(v)$ (where v is any node in the graph). Executing all nodes once is referred to as iteration.

2.2 Iteration

An iteration [3] of a node is the execution of the node exactly once and an iteration of the DFG is the execution of each node in DFG exactly once. In Fig. 2 DFG Iteration will be called as execution of Node A and Node B exactly once. Each edge in a DFG describes a precedence constraint between two nodes. This precedence constraint is an intra-iteration precedence constraint if the edge has zero delays or an inter-iteration precedence constraint if the edge has one or more delays. Together, intra and inter precedence constraints specify the order in which the nodes in the DFG can be executed. The edge from A to B enforces the intra-iteration precedence constraint which states that the k-th iteration of A must be executed before the k-th iteration of B and is denoted by $A_k \rightarrow B_k$. B to A enforces the inter iteration precedence constraint, which states that the k-th iteration of B must be executed before (k+1)-th iteration of A and is denoted by $B_k \Rightarrow A_{k+1}$.

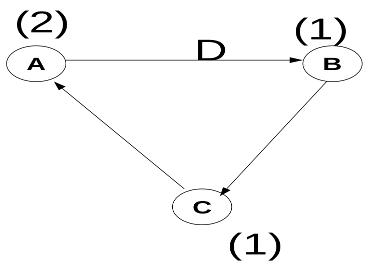

Figure 1. Data Flow Graph

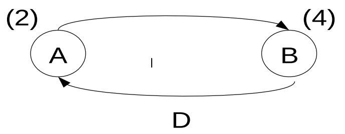

Figure 2. Iteration

2.3 Iteration period

An iteration period [3] is the time required for the execution of one iteration of the algorithm. As Fig. 2 the iteration period will be Computation time of node A *i.e.* 2 u.t. (Unit) plus Computation time of node B *i.e.* 4 u.t., so the iteration period of DFG will be 6 u.t. The iteration rate is the reciprocal of the iteration period. The iteration period of a repeating schedule is the average computation time per iteration. A repeating schedule with cycle period c and unfolding factor f has iteration period c/f .

2.4 Loop bound and iteration bound

A Loop [1][3] is a directed path that begins and ends at the same node, such as the path $A \rightarrow B \rightarrow A$ in Fig. 2. The amount of time required to execute a loop can be determined from the precedence relations described by the edges of the DFG. In Fig. 2 the edges describe the precedence constraints: $A_0 \rightarrow B_0 \Rightarrow A_1$. According to these precedence constraints, iteration k of the loop consists of sequential execution of A_k *and* B_k. Given that the execution times of nodes A and B *are* 2 and 4 u.t. respectively. One iteration of the loop requires 6 u.t. This is the loop bound, which represents the lower bound on the loop computation time. Finally, the loop bound of the l-th loop is defined as t_l / w_l, where t_l is the loop computation time and w_l is the number of delays in the loop. The loop bound for the DFG shown in Fig. 2 is 6/1 u.t. The DFG shown in Fig. 3 contains two loops, namely loop $l_1 = A \rightarrow B \rightarrow A$ and $l_2 = A \rightarrow B \rightarrow C \rightarrow A$. The loop bounds for l_1 and l_2 are $6/2 = 3$ u.t and $11/1 = 11$ u.t., respectively. The critical loop [1][3] is the loop with the maximum loop bound, so the loop l_2 is the critical loop in this example. The loop bound of the critical loop is the iteration bound of the DSP program, which is the lower bound on the iteration period or sample period of the DSP program regardless of the amount of computing resources available. Formally iteration bound is defined as [3]:

$$T_\infty = \max \{ tl / wl \} \text{ where } l = 1, 2, 3, \quad (1)$$

So as shown in Fig. 3 we have:

$$T_\infty = \max \{6/2, 11/1\} = 11 \text{ u.t.} \quad (2)$$

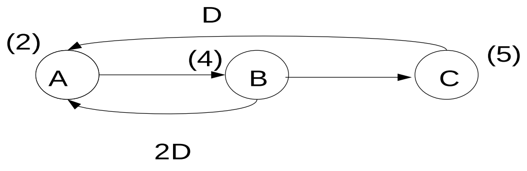

Figure 3. Iteration Bound

3. UN-FOLDING

Un-Folding [3][6] is a transformation technique that can be applied to a DSP program to create a new program describing more than one iteration of the original program. More specifically, unfolding a DSP program by the unfolding factor *J* creates a new program that describes *J* consecutive iterations of the original program. Un-folding is also referred to as loop unrolling and has been used in compiler theory. Un-Folding has applications in designing high-speed and low power VLSI architectures. One application is to unfold the program to reveal hidden concurrencies so that the program can be scheduled to a smaller iteration period, thus increasing the throughput of the implementation. Another application is to design parallel architectures at the word level and bit level to increase the throughput or decrease the power consumption of the implementation. Un-Folding is equivalent to parallel processing. As shown in Fig. 4(b), unfolding factor 2 is used. As we have seen in Fig. 4 that T_∞ is same in both the cases i.e. in Fig. 4(a) and Fig. 4(b) but in Fig. 4(a) one sample was the output of the system but in Fig. 4(b) which is 2 Un-Folded. In this two samples are output in same time hence increasing the throughput of the system.

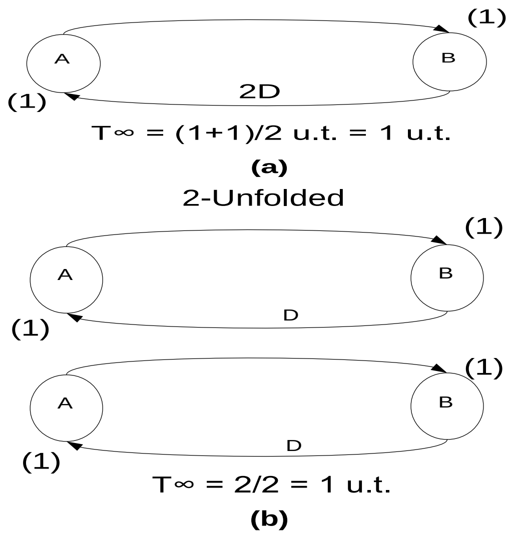

Figure 4. (a) Simple DFG
(b) Un-Folded DFG by 2 factor

Un-Folding is done to meet the following: (i) Sample Period Reduction, (ii) Parallel Processing. Where Sample Period Reduction is applicable in three cases when: (i) Iteration period is not equal to iteration bound and a node in the DFG having computation time greater than its T_∞ , (ii) Iteration period is not equal to iteration bound and Iteration bound is not an integer, (iii) Iteration period is not equal to iteration bound, Longest node computation is larger than the iteration bound T_∞, and T_∞ is not an integer. In this paper we are performing sample period reduction of comb filter where a node in comb filter has computation time greater than its T_∞ and T_∞ is not an integer. An implementation of the DSP program can never achieve an iteration period less than the iteration bound, even if infinite processors are available. The comb filter has sample period more than iteration bound. Un-Folding is the only way we can achieve iteration period equal to iteration bound.

4. ALGORITHM FOR UN-FOLDING

In this section , an algorithm for Un-Folding [3][6] a DFG by an unfolding factor *J* for sample period reduction is described. Some notations going to be used in algorithm are explained as follows: The operation $\lfloor x \rfloor$ is the floor of x, which is the largest integer less than or equal to x. For example, $\lfloor 13/4 \rfloor = 3$. The operation $a \% b$ is the remainder after dividing a by b, where a and b are integers. For example, 11%3 = 2. Before unfolding algorithm is described, we also have a look upon as how to decide the value of unfolding factor *J* . In this paper we are performing Sample period reduction on comb filter which has three cases and there are corresponding ways to find Un-Folding factor *J* described as follows: (i) Iteration period is not equal to iteration bound and a node in the DFG having computation time greater than its T_∞ : If the computation time of node *U*, t_U is greater than the iteration bound, T_∞, then $\lceil t_U / T_\infty \rceil$ - unfolding should be used, where $\lceil x \rceil$ is the ceiling of x, which is the smallest integer greater than or equal to x. For example, $t_U = 4$ and $T_\infty = 3$, so $\lceil 4/3 \rceil$ - unfolding (i.e., 2-unfolding) will be used, (ii) Iteration period is not equal to iteration bound and Iteration bound is not an integer: If a iteration bound is of the form t_l / w_l where t_l and w_l are mutually coprime, then w_l - unfolding should be used. For example, a DFG has iteration bound $T_\infty = 4/3$ and it does not have any longest computation node time greater than it's iteration period. This DFG can be unfolded with unfolding factor 3, (iii) Iteration period is not equal to iteration bound, Longest node computation is larger than the iteration bound T_∞, and T_∞ is not an integer: In this case, the minimum unfolding factor that allows the iteration period to equal to the iteration bound is the minimum value of *J* such that $J\,T_\infty$ is an integer and is greater than or equal to the longest node computation time. For example, if $T_\infty = 4/3$ and the longest node computation time is 6, then the minimum unfolding factor that allows the iteration period to equal the iteration bound is $J = 6$ because this is the minimum value of J such that $J(4/3)$ is an integer and $J(4/3) \geq 6$. In this paper we will use the third case of sample period reduction to determine unfolding factor J.

For each U in the original DFG, there are J nodes with the same function as U in the J unfolded DFG and also for each edge in the original DFG, there are J edges in the J-unfolded DFG. Keeping in mind all above mentioned points following steps of unfolding algorithm can be used to construct a J unfolded Data flow graph:

1. For each node U in the original DFG, draw the J nodes $U_0, U_1, \cdots, U_{J-1}$.
2. For each edge $U \rightarrow V$ with w delays in the original DFG, draw the J edges $U_i \rightarrow V_{(i+w)\%J}$ with $\lfloor (i+w)/J \rfloor$ delays for $i = 0, 1, \cdots, J-1$.

5. PROPERTIES OF UN-FOLDING

The properties of Un-Folding [3][6] are described as follows:

1. Un-Folding preserves the number of delays in a DFG. This can be stated as follows:

   $\lfloor w/J \rfloor + \lfloor (w+1)/J \rfloor + \lfloor (w+J-1)/J \rfloor = w$ (3)

2. J-unfolding of a loop l with w_l delays in the original DFG leads to gcd (w_l, J) loops in the unfolded DFG, and each of these gcd(w_l, J) loops contains w_l /gcd(w_l, J) delays and J /gcd(w_l, J) copies of each node that appears in l.
3. Un-Folding a DFG with iteration bound T_∞ results in a J unfolded DFG with iteration bound $J T_\infty$.

6. COMB FILTER

In signal processing, a comb filter [10] [11] adds a delayed version of a signal to itself, causing constructive and destructive interference. The frequency response of a comb filter consists of a series of regularly spaced spikes, giving the appearance of a comb. Comb filters are used in a variety of signal processing applications. These include: (i) Cascaded Integrator-Comb (CIC) filters, commonly used for anti-aliasing during interpolation and decimation operations that change the sample rate of a discrete-time system [8], (ii) 2D and 3D comb filters implemented in hardware (and occasionally software) for PAL and NTSC television decoders. The filters work to reduce artifacts such as dot crawl [9], (iii) Audio effects, including echo, flanging, and digital waveguide synthesis. For instance, if the delay is set to a few milliseconds, a comb filter can be used to model the effect of acoustic standing waves in a cylindrical cavity or in a vibrating string [10]. In acoustics, comb filtering can arise in some unwanted ways. For instance, when two loudspeakers are playing the same signal at different distances from the listener, there is a comb filtering effect on the signal. In any enclosed space, listeners hear a mixture of direct sound and reflected sound. Because the reflected sound takes a longer path, it constitutes a delayed version of the direct sound and a comb filter is created where the two combine at the listener.

Comb filters exist in two different forms: (i) feed-forward, (ii) feedback. The names refer to the direction in which signals are delayed before they are added to the input. Comb filters may be implemented in discrete time or continuous time.

6.1 Feedback form

Similarly, the general structure of a feedback comb filter is shown in Fig. 6. It may be described by the following difference equation:

$$y[n] = x[n] + ay[n-k] \quad (5)$$

In Fig. 6, K is the delay length (measured in samples), and a is a scaling factor applied to the delayed signal.

In this paper we are using feed backward form of comb filter with delay of 3 units.

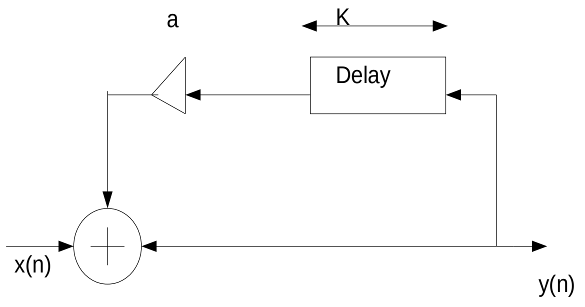

Figure 6. Feed Backward Comb Filter

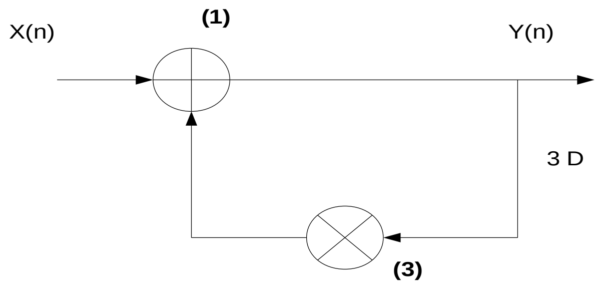

Figure 7. Comb Filter with delay of 3 units.

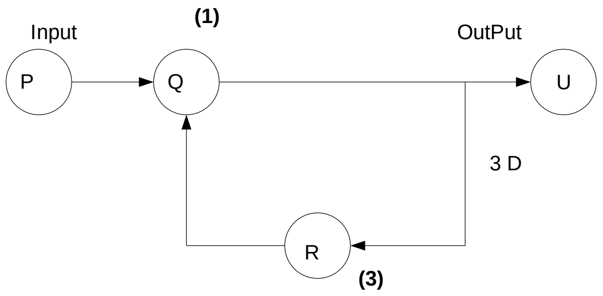

Figure 8. DFG of Comb Filter with delay of 3 units.

7. IMPLEMENTATION OF COMB FILTER USING UNFOLDING

Comb filter is shown in the Fig. 7 with delay of 3 u.t. The DFG for the same is given in Fig 8.As can be seen from Fig. 8 that the node R is 3 u.t. and node Q is 1 u.t. Therefore the sampling time for the filter would be the max(3,1) i.e. 3 u.t. There is only one loop present in the filter so the computation time for the loop is 5 u.t. Using equation (1) we can find the iteration bound i.e. from Fig. 8, t_l is 3 u.t. and w_l is 3 u.t and also there is only one single loop present in DFG hence the iteration bound will be max(4⁄3) = 4⁄3. As it can be seen from the results that the sample period for DFG shown in Fig. 8 can not be achieved equal to the iteration bound because node computation time is larger than the iteration bound. Un-Folding is the technique which allows the DSP program to be implemented with an iteration period equal to iteration bound.

The comb filter we have taken has iteration bound $T_\infty = 4/3$ and is not an integer. The sample period or iteration period will be the longest computation node time i.e. as shown in Fig. 8 is node R having computation time of 3 u.t. Furthermore it has iteration bound greater than iteration period so it falls under the third case of sample period reduction as described in section *V I*. Here, the minimum unfolding factor that allows the iteration period to equal the iteration bound is $J = 3$ because this is the minimum value of J such that J (4⁄3) is an integer i.e. 3(4⁄3) = 4 and J (4⁄3) ≥ 4 i.e. 3(4⁄3) ≥ 4. Thus, Un-Folding factor used here is 3. So for each node (*P, Q, R, U)* as shown in Fig. 8, draw the 3 nodes for each node i.e. $P(P_0, P_1, P_2)$, $Q(Q_0, Q_1, Q_2)$, $R(R_0, R_1, R_2)$, $U(U_0, U_1, U_2)$ as shown in Fig. 9.

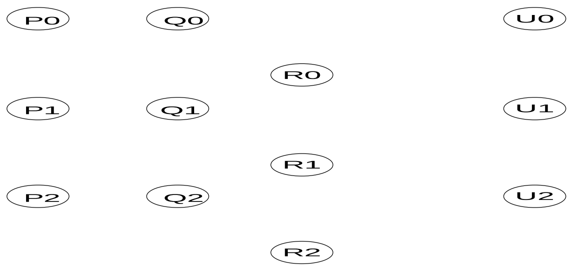

Figure. 9

Using the Un-Folding algorithm following are the calculation steps to obtain unfolded DFG:

1. $U_0 \rightarrow R_{(0+3)\%3}$ *i.e.* $U_0 \rightarrow R_0$ and Delay will be $\lfloor(0+3)/3\rfloor = 1$.
2. $U_1 \rightarrow R_{(1+3)\%3}$ *i.e.* $U_1 \rightarrow R_1$ and Delay will be $\lfloor(1+3)/3\rfloor = 1$.
3. $U_2 \rightarrow R_{(2+3)\%3}$ *i.e.* $U_2 \rightarrow R_2$ and Delay will be $\lfloor(2+3)/3\rfloor = 1$.
4. $R_0 \rightarrow Q_{(0+0)\%3}$ *i.e.* $R_0 \rightarrow Q_0$ and Delay will be $\lfloor(0+0)/3\rfloor = 0$.
5. $R_1 \rightarrow Q_{(1+0)\%3}$ *i.e.* $R_0 \rightarrow Q_1$ and Delay will be $\lfloor(1+0)/3\rfloor = 0$.
6. $R_2 \rightarrow Q_{(2+0)\%3}$ *i.e.* $R_0 \rightarrow Q_2$ and Delay will be $\lfloor(2+0)/3\rfloor = 0$.
7. $Q_0 \rightarrow U_{(0+0)\%3}$ *i.e.* $Q_0 \rightarrow U_0$ and Delay will be $\lfloor(0+0)/3\rfloor = 0$.
8. $Q_1 \rightarrow U_{(1+0)\%3}$ *i.e.* $Q_1 \rightarrow U_1$ and Delay will be $\lfloor(1+0)/3\rfloor = 0$.
9. $Q_2 \rightarrow U_{(2+0)\%3}$ *i.e.* $Q_2 \rightarrow U_2$ and Delay will be $\lfloor(2+0)/3\rfloor = 0$.
10. $P_0 \rightarrow Q_{(0+0)\%3}$ *i.e.* $P_0 \rightarrow Q_0$ and Delay will be $\lfloor(0+0)/3\rfloor = 0$
11. $P_1 \rightarrow Q_{(1+0)\%3}$ *i.e.* $P_1 \rightarrow Q_1$ and Delay will be $\lfloor(1+0)/3\rfloor = 0$.
12. $P_2 \rightarrow Q_{(2+0)\%3}$ *i.e.* $P_2 \rightarrow Q_2$ and Delay will be $\lfloor(2+0)/3\rfloor = 0$.

As per the calculations shown above join the nodes in Fig. 9 for example, U_0 *with* R_0, R_0 *with* Q_0, Q_0

with U_0 *and* P_0 *with* Q_0 and also mark the corresponding delays calculated above for each pair of node to produce unfolded DFG shown in Fig. 10.

8. RESULTS

In this paper we had implemented comb filter using unfolding technique. The DFG of the same is given in Fig. 8. It has one addition nodes having computation time of 1 u.t. and one multiplication nodes having computation time of 3 u.t. It has sampling time of 3 u.t. and the iteration bound as 4⁄3. It was not possible to achieve its sample period equal to the iteration bound because node computation time is larger than the iteration bound. But after applying unfolding on comb filter Un-Folded DFG is produced shown in Fig. 10 has iteration period equal to iteration bound i.e. 4⁄3. The iteration bound for the Un-Folded DFG is T_∞ = 4 u.t. The Un-Folded DFG performs 3 iterations of original DSP program in 4 u.t., so the sample period of the Un-Folded DFG is 4⁄3 which is equal to the iteration bound of original DFG.

9. CONCLUSIONS

We have presented comb filter shown in Fig. 10 with sample period equal to iteration bound. Initially comb filter did not have sample period (3 u.t.) equal to iteration bound (4⁄3) and it was difficult to implement the comb filter with high sample period of 3 u.t. as it reduces the efficiency of the filter. But after unfolding we are able to achieve sample period reduction i.e. 4⁄3 which is equal to iteration bound of original comb filter. As unfolded DFG shown in figure Fig. 10. performs 3 iteration in 4 u.t. Hence, unfolding has increased the efficiency of the comb filter and also reduced the sample period as well.

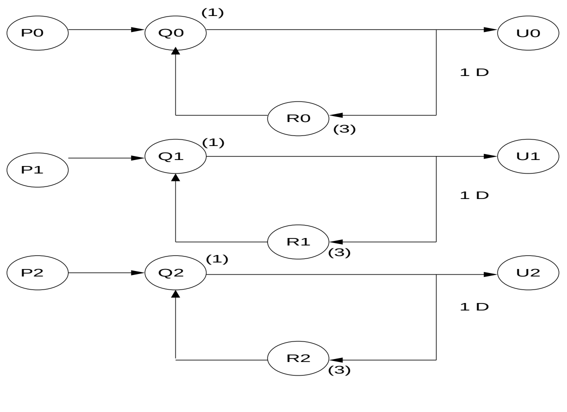

Figure. 10 DFG of Un-Folded Comb Filter of 3 unit delay

REFERENCES

[1]. P.Lapsley, J.Bier, A. Shoham, and E.A. Lee, DSP Processor Fundamentals: Architectures and Features. Elsevier, 1993.

[2]. K.K. Parhi and D.G. Messerschmitt, “Static rate-optimal scheduling of iteration data-flow programs via optimum unfolding,” IEEE Trans. on Computers, vol. 40, no. 2, pp. 178-195, Feb. 1991.

[3]. Keshab K. Parhi, VLSI Digital Signal Processing Systems: Design and Implementation. Wiley, 2009.

[4]. L.E. Lucke and K.K. Parhi, “Data-Flow transformations for critical path time reduction in high-level DSP synthesis,” IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems, vol. 12, no. 7, pp 1063-1068, July 1993.

[5]. L.-F Chao and E. Sha, “Retiming and Unfolding data-flow graphs,” in Proc. of 1992 International Conference on Parallel Processing. part II,(St. Charles,IL), pp. 33-40, Aug. 1992.

[6]. K.K. Parhi, “A systematic approach for design of digit-serial signal processing architectures,” IEEE Trans. On Circuits and Systems, vol. 38, no. 4, pp. 358-375, April 1991.

[7]. Dolecek G.J. and Mitra S.K., “Two-stage CIC-based decimator with improved characteristics,” in Signal Processing, IET, vol. 4, no. 1, pp 22-29, 2010.

[8]. Chua-Chin Wang, Ching-Li Lee, Ming-Kai Chang, “Low-cost video decoder with 2D2L comb filter for NTSC digital TVs,” in Consumer Electronics, IEEE Transactions on, vol. 51, no. 2, pp. 694-698, 2005.

[9]. Klapuri A.P., Eronen A.J., Astola J.T., “Analysis of the meter of acoustic musical signals” in Audio, Speech, and Language Processing, IEEE Transactions on, vol. 14 , no. 1, pp 342-355, 2006.

[10]. John G. Proakis and Dimitris G. Manoakis, Digital Signal Processing: Principles, Algorithms, and Applications, PHI, Third Edition.

[11]. Julius O. Smith III, Physical Audio Signal Processing For Virtual Musical Instruments and audio effects,W3K,2010.

◈◈◈