# Scalable VLSI Architectures for Lattice Structure-Based Discrete Wavelet Transform

Joon Tae Kim, Yong Hoon Lee, Tsuyoshi Isshiki, and Hiroaki Kunieda

***Abstract*—In this paper, we develop a scalable VLSI architecture employing a two-channel quadrature mirror filter (QMF) lattice for the one-dimensional (1-D) discrete wavelet transform (DWT). We begin with the development of systematic scheduling, which determines the filtering instants of each resolution level, on the basis of a binary tree. Then input–output relation between lattices of the QMF bank is derived, and a new structure for the data format converter (DFC) which controls the data transfer between resolution levels is proposed. In addition, implementation of a delay control unit (DCU) that controls the delay between lattices of the QMF is proposed. The structures for the DFC and DCU are regular, scalable, and require a minimum number of registers, and thereby lead to an efficient and scalable architecture for the DWT. A scalable architecture for the inverse DWT is also developed in a similar manner. Finally, pipelining of the proposed architecture is considered.**

***Index Terms*—DCU, DFC, DWT, QMF lattice, scalable, VLSI.**

## I. Introduction

Due to its inherent time-scale locality characteristics, the discrete wavelet transform (DWT) has received considerable attention in digital signal processing applications such as speech and image processing [1]–[5]. The DWT is usually implemented based on the binary tree structured quadrature mirror filter (QMF) bank illustrated in Fig. 1. At each level of the tree for the forward transform, outputs of the two-channel QMF bank $H(z)$ and $G(z)$ are evaluated and decimated by a factor of two. Note that the same filter bank is used at each resolution level. For the inverse, the QMF bank $\tilde{H}(z)$ and $\tilde{G}(z)$ follows the twofold expanders. Here, the filters $H(z)$, $G(z)$, $\tilde{H}(z)$, and $\tilde{G}(z)$ have the perfect reconstruction (PR) property [6]–[8]. An attractive feature of this tree structure, which is useful for VLSI implementation, is stated as follows: in the tree, the number of filter outputs computed during each sample period is upper bounded by two, irrespective of the number of levels. This property naturally leads to VLSI architectures that employ only one pair of filters and iteratively use them for all levels. In fact, all VLSI implementations for DWT introduced so far have such an architecture, which consists of a data format converter (DFC)[1] and a two-channel filter bank [9]–[15]. The DFC controls data

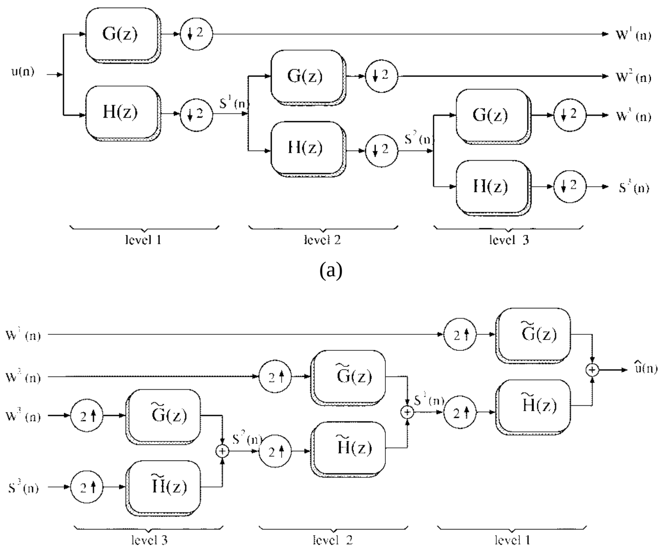


Fig. 1. Three-level binary tree structured QMF banks for DWT: (a) analysis bank for the DWT and (b) synthesis bank for the inverse DWT.

transfer between levels: it stores filtered outputs at a certain level and provides them for filtering at the next level. The two-channel filter bank is implemented either in direct form or in lattice form. Among the architectures with direct form FIR filters [9]–[13], the one in [11] needs less hardware than those in [12] and [13]. The former, however, is not regular, and therefore is difficult to scale; it should be redesigned when either the filter length or the number of resolution levels changes. On the other hand, the ones in [12] and [13], which are based on systolic array, have regular structures and are scalable. The two-channel filter bank in lattice form, which is often refered to as the two-channel QMF lattice, is considerably simpler to implement than the filter bank in direct form: the hardware complexity of the former is about half of that of the latter [8], [16], [17]. Efficient DWT architectures employing the QMF lattice are proposed in [14] and [15]. These, however, are not scalable, and hardware complexity of the DFC increases exponentially as the number of resolution levels increases.

In this paper, we develop a scalable VLSI architectures employing a two-channel QMF lattice for the one-dimensional (1-D) DWT. We begin with the development of systematic scheduling, which determines the filtering instants of each resolution level, on the basis of a binary tree. In contrast to the DWT scheduling algorithms in [9]–[15], the proposed algorithm provides a closed-form expression for scheduling.

Manuscript rceived December 5, 1996; revised November 24, 1997.

J. T. Kim and Y. H. Lee are with the Department of Electrical Engineering, Korea Advanced Institute of Science and Technology, 373-1 Kusong-dong, Yusong-gu, Taejon, 305-701, Korea.

T. Isshiki and H. Kunieda are with the Department of Electrical Engineering, Tokyo Institute of Technology, 2-12-1 Ookayama, Meguro-ku, Tokyo 152, Japan.

Publisher Item Identifier S 1057-7130(98)04670-9.

[1] In [12], this is called the routing network.

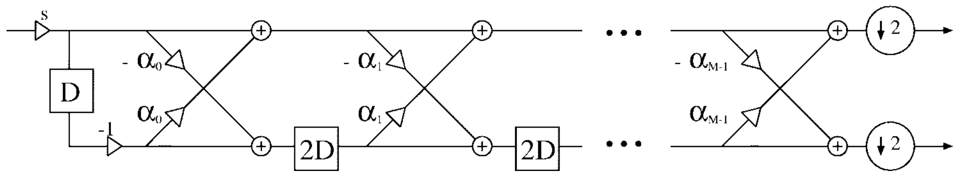


Fig. 2. Two-channel QMF lattice implementing $H(z)$ and $G(z)$ of length $2M$.

Using this expression, a scalable DFC is developed. In addition, a scalable DCU that controls the delay between lattices of the QMF bank is proposed. It will be shown that these DFC and DCU require a minimum number of registers. The proposed DWT architecture consisting of the DFC, DCU, and the QMF lattice is scalable and requires less hardware than those in [9]–[15].

The rest of the paper is organized as follows. In Section II, the proposed architecture for the DWT is developed. In Section III, we present an architecture for the inverse DWT. Finally, pipelining of the proposed architecture is discussed in Section IV.

## II. Analysis Stage Wavelet Architecture

Consider again Fig. 1(a). Due to the decimations, the data rate decreases by a factor of two as we move from one resolution level to the next. Therefore, the operating frequency of the filters $G(z)$ and $H(z)$ in level $j$ is given by $f_s/2^{j-1}$ where $f_s$ is the input sampling frequency. Fig. 2 illustrates the lattice filter implementing $G(z)$ and $H(z)$. The high-pass and low-pass filtered outputs are produced by the upper and lower nodes of the lattice, respectively. For $j$th level filtering, the delay $D$ and $2D$ indicate $2^{j-1}T_s$ and $2^jT_s$ time unit delays, respectively, where $T_s = 1/f_s$. Note that for filter length $2M$, this structure requires only $M$ lattices, each of which consists of two multipliers and two adders. Therefore, its hardware complexity is about half of that of a pair of direct form FIR filters.

Our objective is to implement all resolution levels of the DWT by employing only one lattice filter which operate with frequency $f_s$. To achieve this, we compute only the filter outputs which are not thrown away by the decimations. At level 1, outputs are evaluated every other sample times, say, $n = 2l,\ l = 0, 1, 2, \cdots$. When these are obtained by the lattice filter operating with frequency $f_s$, the filter becomes idle for $n = 2l + 1$. Now we can compute outputs of the other resolution levels during these idle periods. This can be seen from the Data Dependence Graph (DDG's) shown in Fig. 3. In this figure, the processing element (PE) is the lattice consisting of two multipliers and two adders; $u(n)$ is the input to the 1st level; $l$th filtered outputs at level $j$ are denoted by $S^j(l)$ and $W^j(l)$, respectively. The delay $z^{-1}$ indicates $T_s$ $(= 1/f_s)$ time unit delay. Since the delay between adjacent lattices at level $j$ is $2^jT_s$, the delays between lattices in levels 2 and 3 are represented as $z^{-4}$ and $z^{-8}$, respectively. In the DDG for level $j$, the lower level output of the $i$th PE at time $n$ is inputted to the $(i + 1)$th PE at time $n + 2^j$. This holds because the delay between lower levels of the adjacent lattices is $2^jT_s$. As described above, the filtering for level 1 is performed at $n = 2l,\ l = 0, 1, 2, \cdots$. To exploit the idling time slots $n = 2l + 1$, for level 2 we start filtering at $n = 1$ and compute the outputs at $n = 4l + 1$. Similarly, the outputs for level 3 are computed at $n = 8l + 3$. Now it should be noted that the filter execution times $2l$, $4l + 1$, and $8l + 3$ never overlap with each other, and that the $2l$th and $(2l - 1)$th low-pass outputs from level $j - 1$, $S^{j-1}(2l)$ and $S^{j-1}(2l - 1)$, which are required for computing the $l$th output at level $j$, $S^j(l)$, are obtained before initializing the computation of $S^j(l)$. Based on these observations, we can combine the three DDG's in Fig. 3(a)–(c). The result is shown in Fig. 4. This DDG indicates that the outputs of all three resolution levels of the DWT can be evaluated by employing only one lattice filter which operates with frequency $f_s$. In what follows, we shall extend this result to the DWT with arbitrary number of resolution levels.

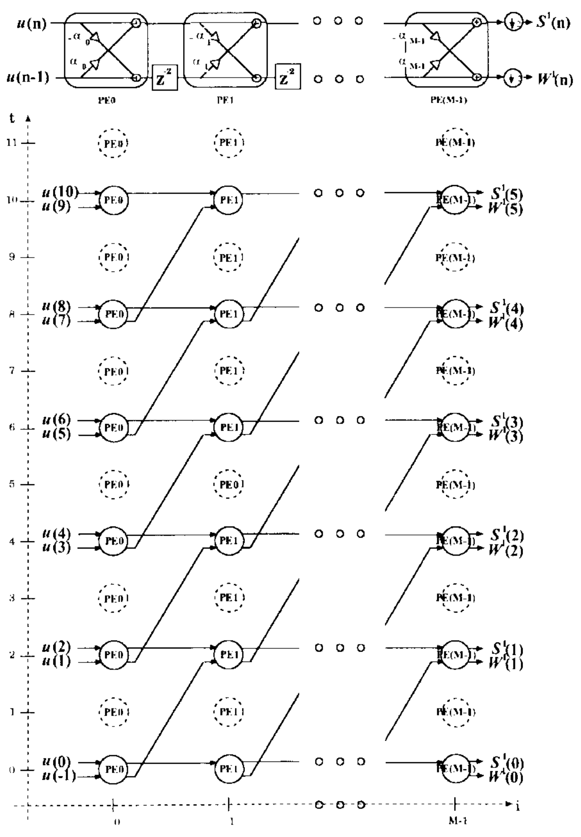
Denote the filtering instants for the $j$th level by $t_j(l)$, $l = 0, 1, 2, \cdots$. Then $t_1(l) = 2l$, $t_2(l) = 4l + 1$, and $t_3(l) = 8l + 3$. Note that the $l$th output of the $j$th resolution level is computed at time $t_j(l)$. Since the outputs of the $j$th level is produced every $2^j$ time unit, $t_j(l)$ is expressed as $t_j(l) = 2^j \cdot l + t_j(0)$, where $t_j(0)$ is the instant at which the output of the $j$th level is calculated for the first time. The constant $t_j(0)$ can be obtained with the help of a binary tree, shown in Fig. 5, which successively decomposes the set of nonnegative integers by half. At the top of the tree, we have the set $\{n \mid n = 0, 1, 2, \cdots\}$. In the second level of the tree, this set is decomposed into even and odd number sets, $\{2l\}$ and $\{2l + 1\}$. The even set $\{2l\}$ provides the filtering instants for the first resolution level, i.e., $t_1(l) = 2l$. The odd set $\{2l + 1\}$ is again decomposed into two sets $\{4l + 1\}$ and $\{4l + 3\}$—note that $\{4l + 1\} \cup \{4l + 3\} = \{2l + 1\}$. From the set $\{4l + 1\}$, we get $t_2(l) = 4l + 1$ and the set $\{4l + 3\}$ is decomposed further. Continuing in this manner, we can obtain the following general expression for $t_j(l)$, $1 \le j \le J$:

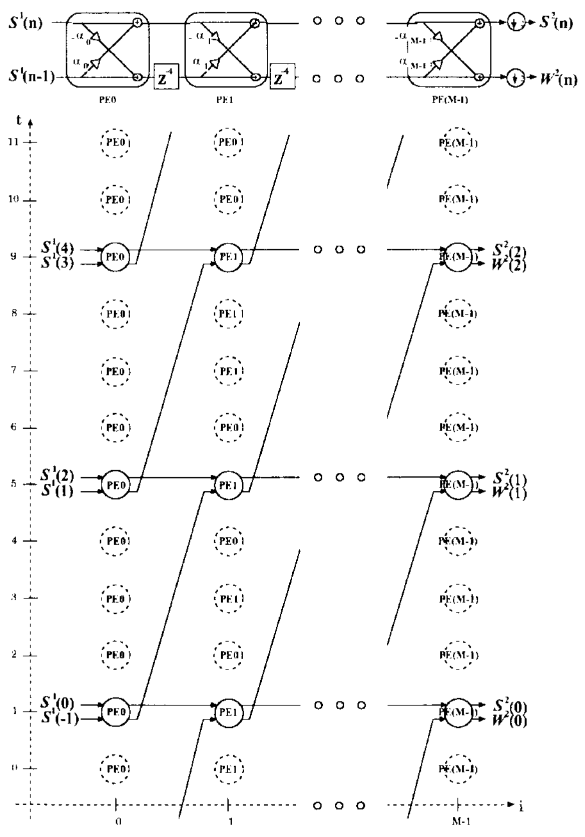
$$t_j(l) = 2^j \cdot l + 2^{j-1} - 1 \tag{1}$$

where $J$ is the number of resolution levels. Due to the successive decompositions, the filtering instants given by (1) never overlap with each other. Furthermore, this scheduling leads to the following observation.

*Observation 1:* If we schedule filtering operations based on (1), then the $2l$th and $(2l - 1)$th low-pass outputs of the $(j-1)$th level, $S^{j-1}(2l-1)$ and $S^{j-1}(2l)$, are obtained before initializing the computation of $S^j(l)$. Specifically, $S^{j-1}(2l-1)$ and $S^{j-1}(2l)$, respectively, are computed $(2^{j-1} + 2^{j-2})$ and $2^{j-2}$ time units before the evaluation of $S^j(l)$.

*Proof:* The first part is proved by showing that $t_{j-1}(2l-1) < t_{j-1}(2l) < t_j(l)$. Here the first inequality is obvious. Now $t_{j-1}(2l) = 2^{j-1} \cdot 2l + 2^{j-2} - 1 = 2^j \cdot l + 2^{j-1}-$


(a)


(b)

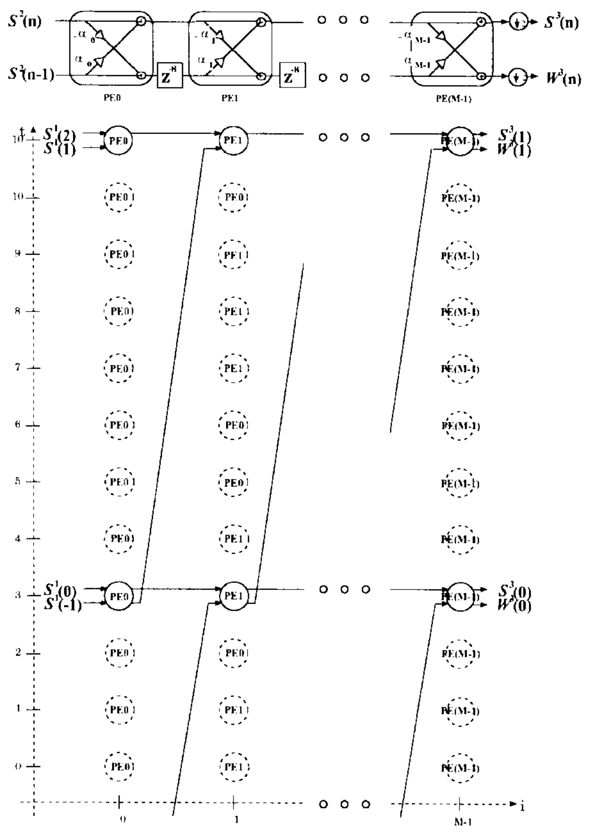

(c)

Fig. 3. Data dependence graph between lattices. (a) Level 1. (b) Level 2. (c) Level 3.

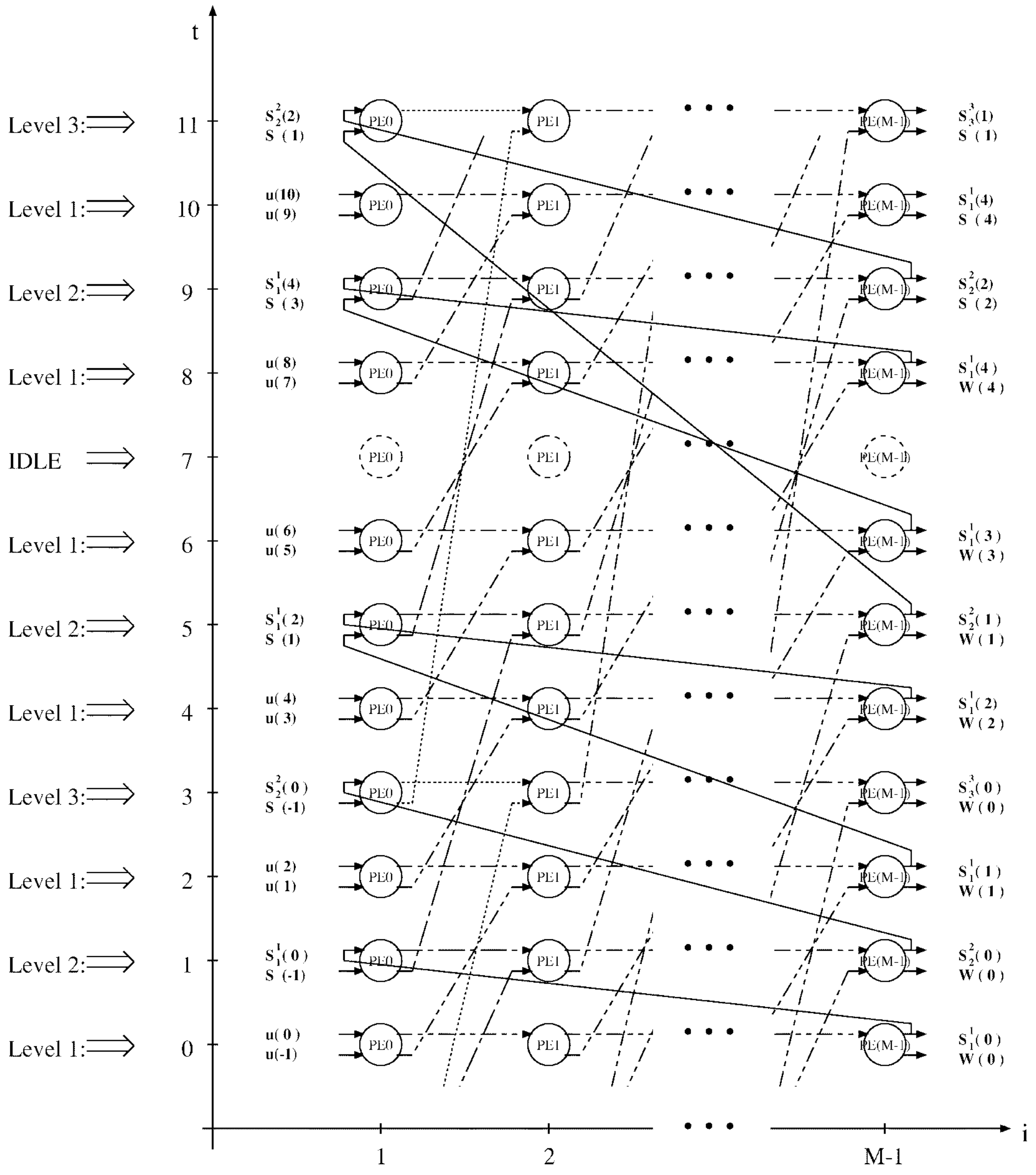


Fig. 4. Data dependence graph for the DWT with three resolution levels.

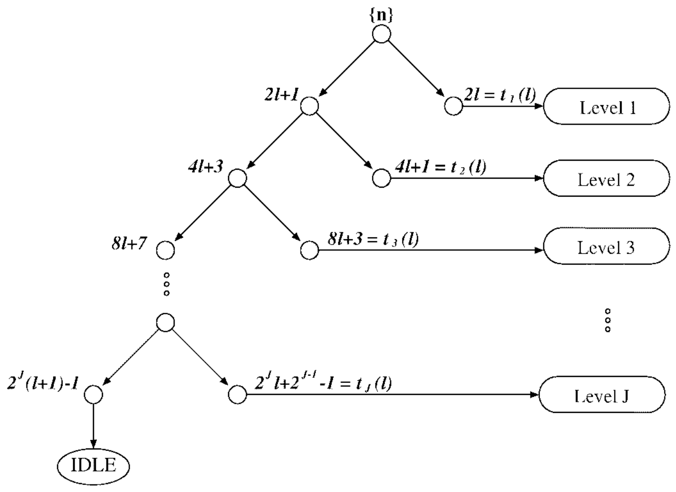


Fig. 5. A binary tree for scheduling.

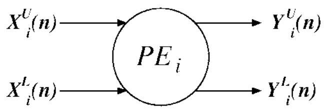


Fig. 6. Notations representing the $n$th inputs and outputs of $\mathrm{PE}_i$.

$1-2^{j-2} < 2^j \cdot l + 2^{j-1} - 1 = t_j(l)$. From these relations, the second part directly follows. □

From this observation we can generalize the DDG in Fig. 4 for an arbitrary number of resolution levels.

For implementing the DWT, we need to establish the input-output relations between the PE's. Denote the $n$th upper and lower inputs associated with the $i$th PE, say $\mathrm{PE}_i$, by $X_i^U(n)$ and $X_i^L(n)$, respectively, where $n = 0, 1, 2, \cdots$. The corresponding outputs are denoted by $Y_i^U(n)$ and $Y_i^L(n)$ (Fig. 6). It should be pointed out that the upper output of $\mathrm{PE}_{M-1}$, $Y_{M-1}^U(n)$, is equal to $S^j(l)$ for $n = 2^j \cdot l + 2^{j-1} - 1$.

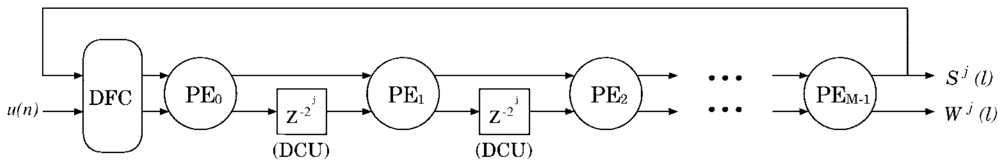


Fig. 7. An architecture for lattice structure-based DWT.

The relation between these inputs and outputs are derived directly from the previous discussions. For the first PE $(\mathrm{PE}_0)$,

$$X_0^U(n) = \begin{cases} u(n), & \text{for } n = 2l \\ Y_{M-1}^U(n-2^{j-2}), & \text{for } n = 2^j \cdot l + 2^{j-1} - 1 \end{cases} \tag{2a}$$

and

$$X_0^L(n) = \begin{cases} u(n-1), & \text{for } n = 2l \\ Y_{M-1}^U(n-2^{j-1}-2^{j-2}), & \text{for } n = 2^j \cdot l + 2^{j-1} - 1. \end{cases} \tag{2b}$$

For the $i$th PE $(\mathrm{PE}_i)$, $1 \le i \le M-1$,

$$X_i^U(n) = Y_{i-1}^U(n) \tag{3a}$$

$$X_i^L(n) = Y_{i-1}^L(n-2^j), \quad \text{for } n = 2^j \cdot l + 2^{j-1} - 1. \tag{3b}$$

These relations lead to the architecture shown in Fig. 7. The Data Format Converter (DFC) controls the input and the feedback sequences depending on (2). The Delay Control Unit (DCU) controls the delay $Z^{-2^j}$, depending on the relation between the time index $n$ and the resolution level $j$, given by (3b). Next we design circuits for implementing the DCU and DFC.

## *A. Design of Delay Control Unit (DCU)*

Consider the design of a DCU for the DWT with three resolution levels $(J = 3)$. For this case, (3b) is rewritten as

$$X_i^L(n) = \begin{cases} Y_{i-1}^L(n-2), & \text{for } n = 2l \\ Y_{i-1}^L(n-4), & \text{for } n = 4l+1 \\ Y_{i-1}^L(n-8), & \text{for } n = 8l+3. \end{cases} \tag{4}$$

This equation directly leads to the structure depicted in Fig. 8. This DCU, which was originally proposed in [15], looks simple but requires about $2^J$ word-level registers for $J$ resolution levels. Note that the required number of registers increases exponentially as the number of levels $J$ increases. It is possible to reduce the number of registers by using the method in [18]. In what follows, we briefly review this method and then develop an alternative approach.

The method in [18] begins with the formation of the lifetime chart that shows the lifetime of each input value. For example, suppose $a, b, c, \cdots$ are the values of $Y_{i-1}^L(n)$ at $n = 0, 1, 2, \cdots$. Their lifetimes are determined according to (4), and marked on a lifetime chart as illustrated in Fig. 9(a). Then, the minimum

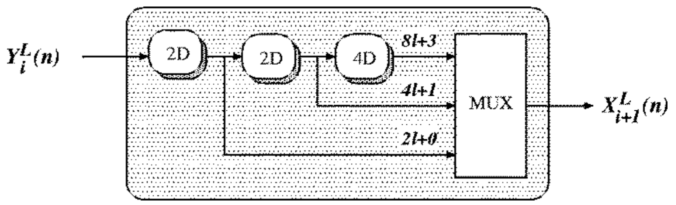


Fig. 8. Directly designed DCU.

number of required registers, say $m$, is obtained by counting the number of live input values at each time instant, and selecting the maximum among the numbers. For the example in Fig. 9(a), $m = 3$. The registers, which are denoted by $R_1, \cdots, R_m$, are connected in cascade. At each time, a value in $R_i$, $i = 1, \cdots, m-1$, is shifted to $R_{i+1}$ if the value is alive; otherwise it is sent to the next PE through a multiplexer. If the value in $R_m$ is alive, it is stored in an empty register. Data flow among registers is summarized in the register allocation table [Fig. 9(b)], and an efficient DCU structure [Fig. 9(c)] and proper switching time are derived by examining the table. The resulting DCU requires minimal number of registers, but is not scalable; if $J$ varies, it should be redesigned.

An alternative structure for the DCU, that we propose, is based on a parallel connection of registers. Suppose that the registers $R_1, \cdots, R_m$ are connected in parallel, where $m$ is the minimal number of registers obtained via the lifetime chart. In this scheme, if an input value is loaded in $R_i$, it is remained in $R_i$ during its lifetime. A clock signal is given to $R_i$ only when a new input is loaded to $R_i$ and the old one is discarded. The register allocation table for the proposed scheme is illustrated in Fig. 10(a) [lifetime chart in Fig. 9(a) is assumed]. It is seen that all input values can be stored during its lifetime without any collision. The resulting DCU structure is shown in Fig. 10(b). The proposed architecture is scalable, as shown in the observation below.

*Observation 2:* For a given number of resolution levels $J$, DCU satisfying (3b) can be implemented as in Fig. 11. The instant, say $ck_j$, at which loading clock signal is applied to the register $R_j$, is expressed as

$$ck_j = 2^j \cdot l + 2^{j-1}.$$

*Proof:* We assume that $Y_i^L(t_j(l))$ is stored in $R_j$, where $t_j(l) = 2^j \cdot l + 2^{j-1} - 1$ and $j = 1, 2, \cdots, J$. This observation can be proved by showing that the lifetime of $Y_i^L(t_j(l-1))$ does not overlap with that of $Y_i^L(t_j(l))$ for all $l$. $Y_i^L(t_j(l-1))$ is generated at $n = t_j(l-1)$ and should be stored in $R_j$ from $n = t_j(l-1)+1$ to $n = t_j \cdot (l-1) + 2^j$. On the other hand, $Y_i^L(t_j(l))$ is generated at $n = t_j(l)$ and should

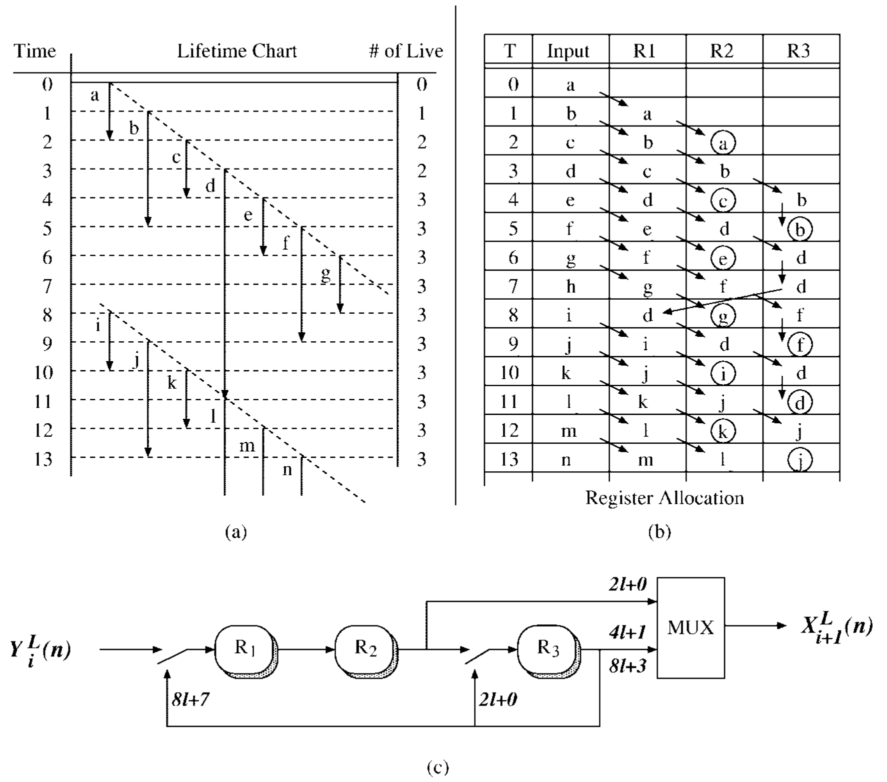
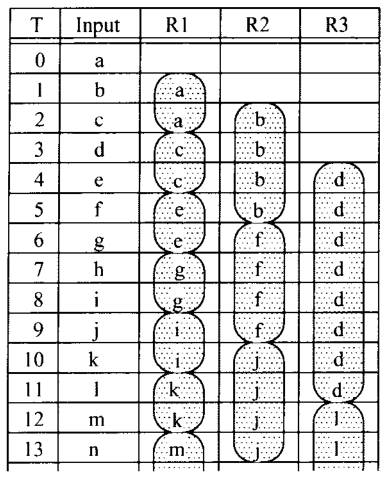
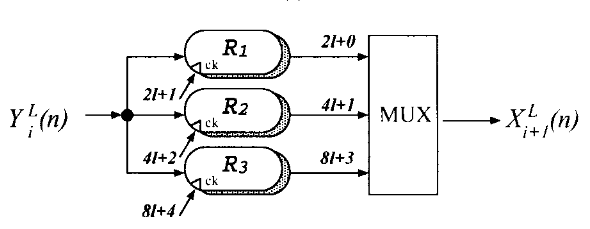


Fig. 9. Procedure for designing DCU using the method in [18]. (a) Lifetime chart. (b) Register allocation table. (c) The resulting DCU.



| T | Input | R1 | R2 | R3 |
|---|---|---|---|---|
| 0 | a | | | |
| 1 | b | a | | |
| 2 | c | a | b | |
| 3 | d | c | b | |
| 4 | e | c | b | d |
| 5 | f | e | b | d |
| 6 | g | e | f | d |
| 7 | h | g | f | d |
| 8 | i | g | f | d |
| 9 | j | i | f | d |
| 10 | k | i | j | d |
| 11 | l | k | j | d |
| 12 | m | k | j | l |
| 13 | n | m | j | l |

(a)



(b)

Fig. 10. The proposed design procedure for the DCU. (a) Register allocation table. (b) The resulting DCU.

be stored in $R_j$ from $n = t_j(l) + 1$ to $n = t_j(l) + 2^j$. Note that $t_j(l-1) + 2^j = t_j(l)$. This proves the nonoverlapping property, and $Y_i^L(t_j(l))$ should be latched at the beginning of $n = t_j(l) + 1$. This indicates that $ck_j$ should be $2^j \cdot l + 2^{j-1}$. □

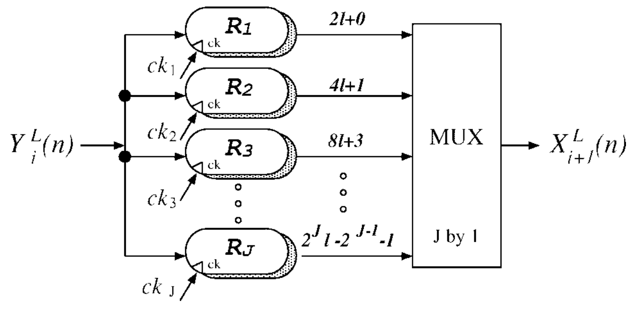


Fig. 11. The proposed DCU structure, where $ck_j = 2^j \cdot l + 2^{j-1}$.

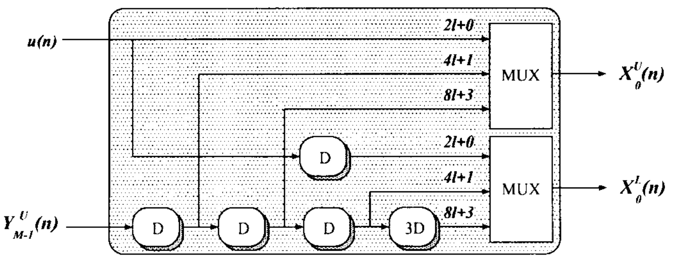


Fig. 12. Directly designed DFC.

## *B. Design of DFC*

Now we consider the design of a DFC for the DWT with three resolution levels $(J = 3)$. In this case, (2) can be rewritten as

$$X_0^U(n) = \begin{cases} u(n), & \text{for } n = 2l \\ Y_{M-1}^U(n-1), & \text{for } n = 4l+1 \\ Y_{M-1}^U(n-2), & \text{for } n = 8l+3 \end{cases} \tag{5a}$$

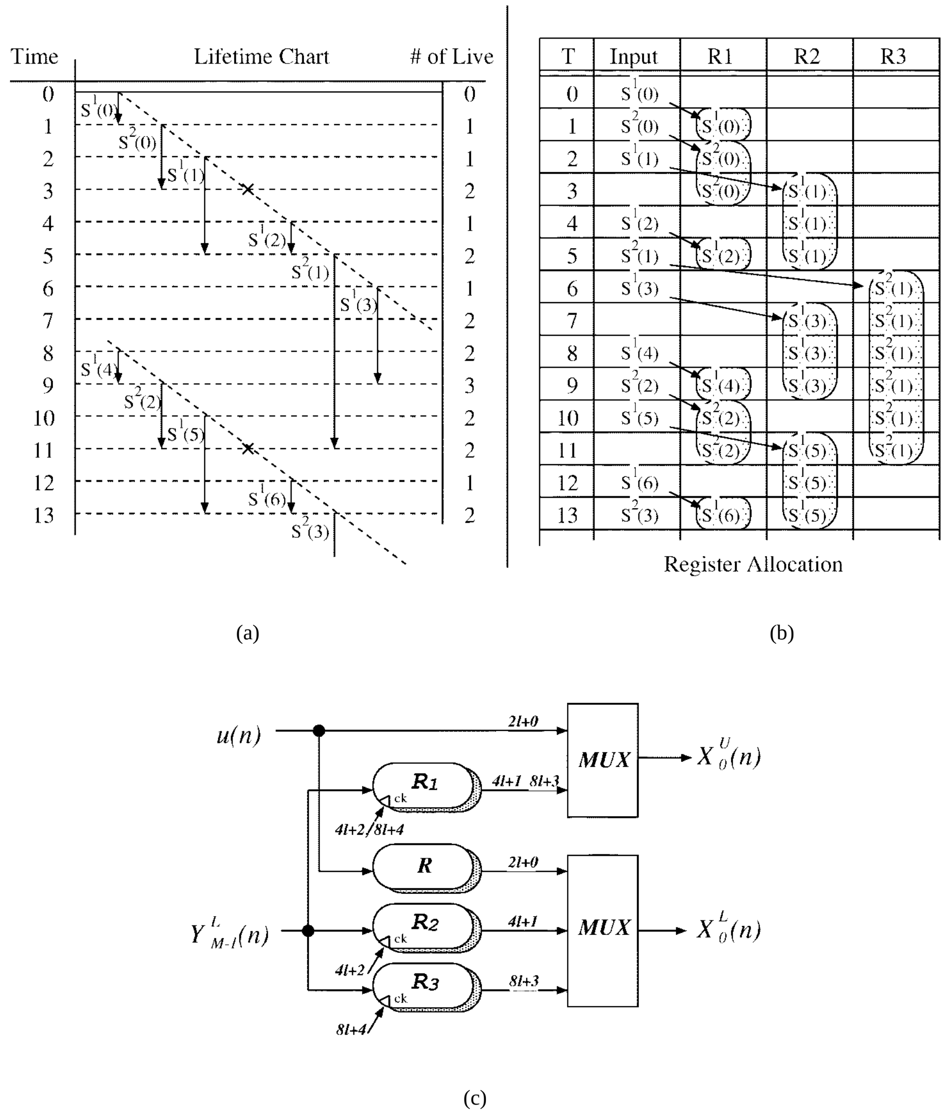


Fig. 13. The proposed design procedure for DFC. (a) Lifetime chart. (b) Register allocation table. (c) The resulting DFC, where $R$ denotes a register to which a clock signal is given at each time instant.

and

$$X_0^L(n) = \begin{cases} u(n-1), & \text{for } n = 2l \\ Y_{M-1}^U(n-3), & \text{for } n = 4l+1 \\ Y_{M-1}^U(n-6), & \text{for } n = 8l+3. \end{cases} \tag{5b}$$

Direct design of DFC with these equations leads to the structure in Fig. 12. This DFC, which was also employed in [15], requires $3 \cdot 2^{J-2} + 1$ word-level registers. The number of registers can be reduced by applying the method for DCU design. Specifically, from the lifetime chart and the register allocation table in Fig. 13(a) and (b), respectively, we can obtain the DFC structure in Fig. 13(c). This structure, which is based on parallel connection of registers, employs a minimal number of registers. The scalability of the structure is described below.

*Observation 3:* For a given number of resolution levels $J$, DFC satisfying (2) can be implemented as in Fig. 14. The instant, say $ck_k$, at which clock signal for data loading is

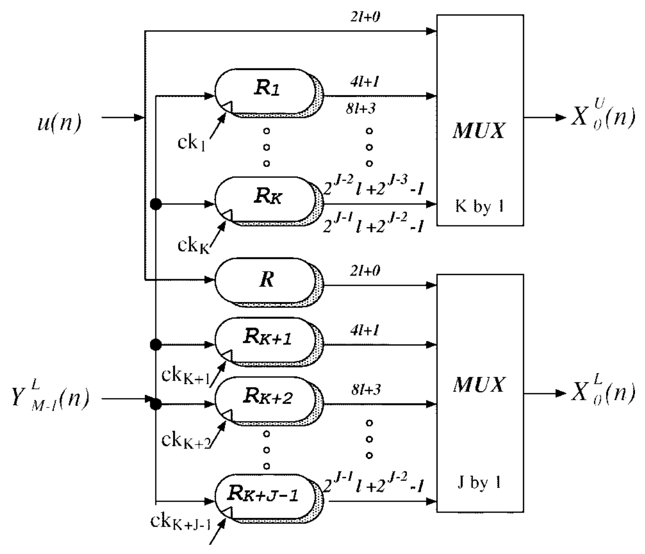


Fig. 14. The proposed DFC structure, where $K = \lceil (J-1)/2 \rceil$ and $R$ denotes a register towhich a clock signal is given at each time.

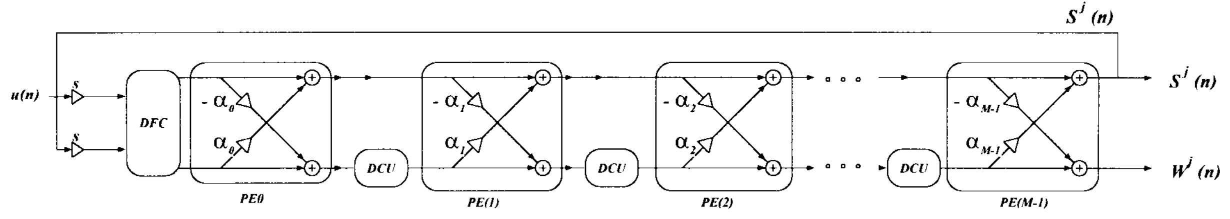


Fig. 15. The proposed architecture for the DWT, where $2M$ is the duration of the analysis filter's impulse response.

TABLE I
HARDWARE COMPLEXITY COMPARISON FOR DWT ARCHITECTURES

| | Direct form [11] | Lattice form [14] | Lattice form [15] | Proposed one |
|---|---|---|---|---|
| # of multi. | 8 | 6 | 6 | 6 |
| # of adder | 6 | 4 | 4 | 4 |
| # of registers | 10 | 45 | 16 | 8 |
| scalability | no | no | no | yes |

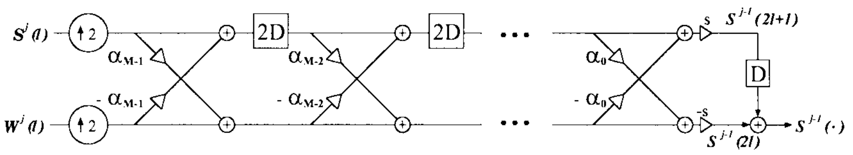


Fig. 16. Two-channel QMF lattice implementing $\tilde{H}(z)$ and $\tilde{G}(z)$ of length $2M$.

applied to the register $R_k$, is expressed as

$$ck_k = \begin{cases} \{t_{2k-1}(2l)+1\} \oplus \{t_{2k}(2l)+1\}, \\ \qquad \text{for } 1 \le k \le K \\ \{t_{k-K}(2l-1)+1\}, \\ \qquad \text{for } K < k \le K+J-1 \end{cases}$$

where $K = \lceil (J-1)/2 \rceil$, and $\{t_{2k-1}(2l)+1\} \oplus \{t_{2k}(2l)+1\}$ implies that the clock is applied both at $\{t_{2k-1}(2l)+1\}$ and $\{t_{2k}(2l)+1\}$.

*Proof:* As mentioned in Observation 1, $S^{j-1}(2l)$ and $S^{j-1}(2l-1)$, $1 \le j < J$, are fed into $X_0^U(\cdot)$ and $X_0^L(\cdot)$ after $(2^{j-2})$ and $(2^{j-1}+2^{j-2})$ time units delay. We assume that the even outputs of adjacent odd and even levels, $S^{2k-1}(2l)$ and $S^{2k}(2l)$, are stored in $R_k$, $1 \le k \le K$. For example, if $J=4$, $R_1$ stores both $S^1(2l)$ and $S^2(2l)$, and $R_2$ stores both $S^3(2l)$ and $S^4(2l)$. In addition, we assume that odd outputs of the $j$th level, $S^j(2l-1)$, is stored in $R_{j+K}$ for $1 \le j < J$. This observation is proved by showing that the lifetimes of variables allocated to a register do not overlap with each other. First, consider $R_k$, $1 \le k \le K$. The lifetime intervals of $S^{2k-1}(2l)$ and $S^{2k}(2l)$ are $(t_{2k-1}(2l), t_{2k}(l)]$ and $(t_{2k}(2l), t_{2k+1}(l)]$, respectively. Then $(t_{2k-1}(2l), t_{2k}(l)] \cap (t_{2k}(2l), t_{2k+1}(l)] = \{(t_{2k-1}(2(2l)), t_{2k}(2l)] \cup (t_{2k-1}(2(2l+1)), t_{2k}(2l+1)]\} \cap (t_{2k}(2l), t_{2k+1}(l)] = \phi$. Here the inequality $t_{2k+1}(l) \le t_{2k-1}(2(2l+1))$ is used. This inequality is always true because $t_{2k-1}(4l+2) = 2^{2k-1}(4l+2)+2^{2k-2}-1 = 2^{2k+1}l+2^{2k}-1+2^{2k-2} > 2^{2k+1}l+2^{2k}-1 = t_{2k+1}(l)$. Therefore, $S^{2k-1}(2l)$ and $S^{2k}(2l)$ can be stored in the register $R_k$ without any collision. Now consider $R_{j+K}$, $1 \le j < J$. Since only the output of $S_j(2l-1)$, $1 \le j < J$, are allocated to $R_{j+K}$, it is sufficient to show that the lifetimes of consecutive outputs of $S_j(2l-1)$ do not overlap with each other. The lifetimes of $S_j(2l-1)$ and $S_j(2(l+1)-1)$ are $(t_j(2l-1), t_{j+1}(l)]$ and $(t_j(2(l+1)-1), t_{j+1}(l+1)]$, respectively. Since $t_{j+1}(l) = 2^{j+1}\cdot l + 2^j - 1 < 2^j \cdot 2l + 2^j + 2^{j-1} - 1 = t_j(2(l+1)-1)$, $(t_j(2l-1), t_{j+1}(l)] \cap (t_j(2(l+1)-1, t_{j+1}(l+1)] = \phi$, and $S_j(2l-1)$ can be stored in $R_{j+K}$ without collision. This proves the first part of this observation. A value allocated to a register should be latched one time unit later than its generation time. This indicates that the $ck_k$ should be applied both at $\{t_{2k-1}(2l)+1\}$ and $\{t_{2k}(2l)+1\}$ for $1 \le k \le K$, and at $\{t = t_{k-K}(2l-1)+1\}$ for $K < k \le K+J-1$. □

## C. The Proposed Architecture for DWT

According to Observations 2 and 3, we can design DCU and DFC for arbitrary number of resolution levels $J$ even without knowing the details of their design methods. Fig. 15 shows a general architecture for the lattice structure based forward DWT. In this figure, all DCU's between lattices have the structure shown in Fig. 11, and the DFC has the structure in Fig. 14. This architecture is valid for any $J$ and $M$. If $J$ varies, only the DFC and DCU's are modified according to Figs. 11 and 14; and we can add or remove the lattice blocks depending on $M$. This architecture requires less hardware than the existing architectures, since the DFC and DCU's employ minimal number of registers and the number of multipliers required by the lattice blocks is about a half of those required by the direct form FIR filters. Table I compares the hardware complexity of the architectures for the DWT

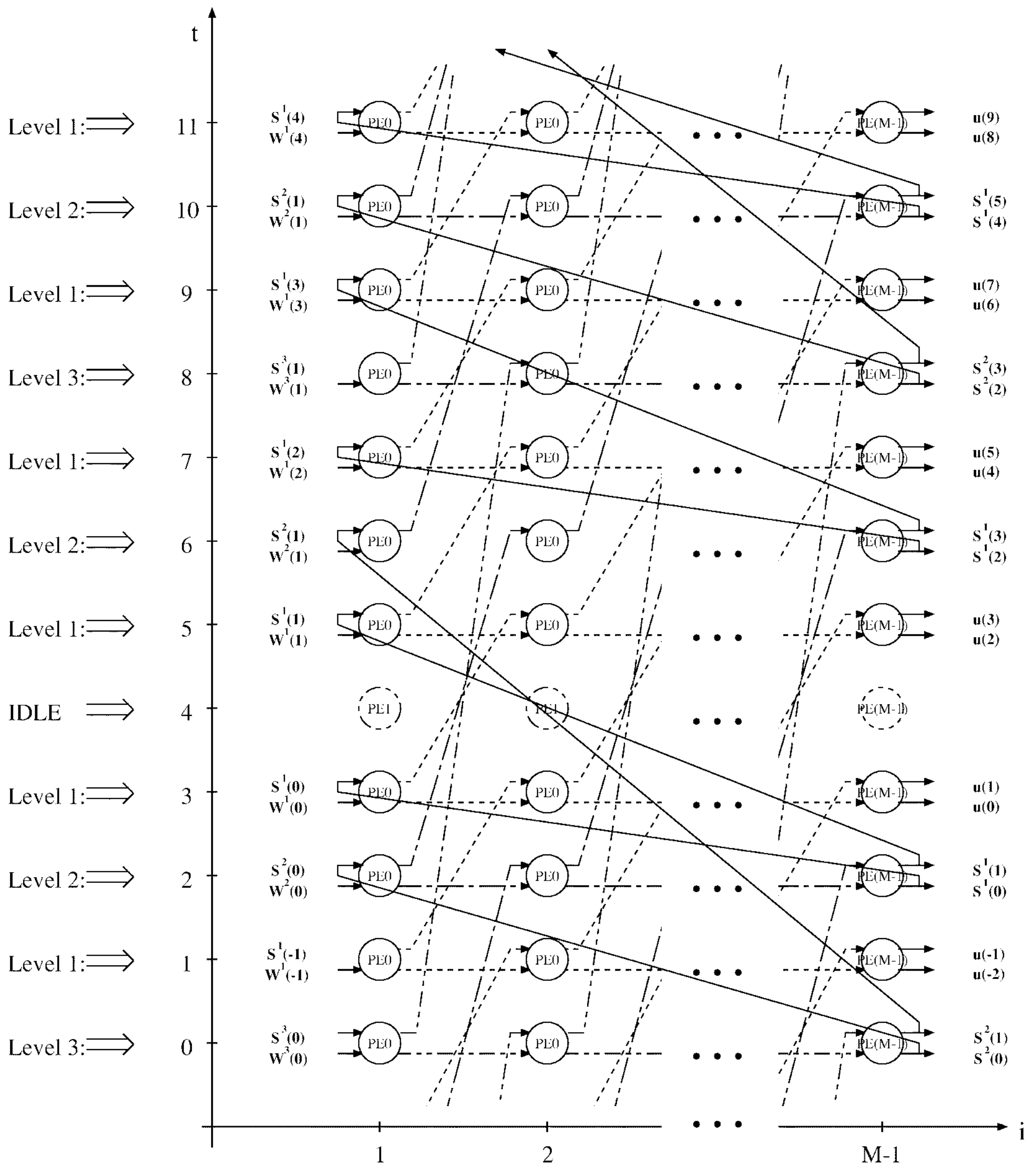


Fig. 17. Data dependence graph for the inverse DWT with three resolution levels.

when $J = 3$, $M = 2$ and pipelining is not considered. As expected, the proposed architecture is considerably simpler to be implemented than the others.

## III. Synthesis Stage Wavelet Architecture

In this section, we develop lattice structure-based architecture for the inverse DWT. Consider again Fig. 1(b). Each level consists of the same synthesis two-channel QMF bank operating with a frequency $f_s/2^{j-1}$ corresponding to level $j$, where $f_s$ is output data rate at level 1. At each level $j$, $S^j(l)$ and $W^j(l)$ are inputted to low-pass and high-pass filters $\tilde{H}(z)$ and $\tilde{G}(z)$, respectively, after being expended by a factor of 2. Due to the expansion, $S^{j-1}(2l)$ is computed only with even coefficients of $\tilde{G}(z)$ and $\tilde{H}(z)$, while $S^{j-1}(2l+1)$ is computed only with corresponding odd coefficients. Fig. 16 illustrates the lattice filter implementing $\tilde{G}(z)$ and $\tilde{H}(z)$. At even time

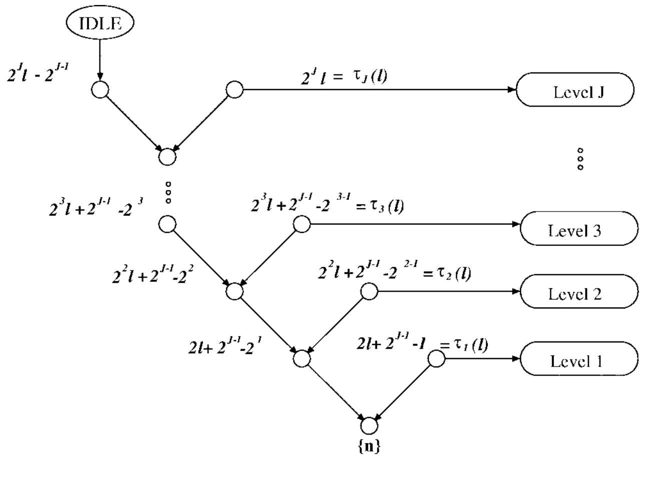


Fig. 18. A flipped binary tree for scheduling of the inverse DWT.

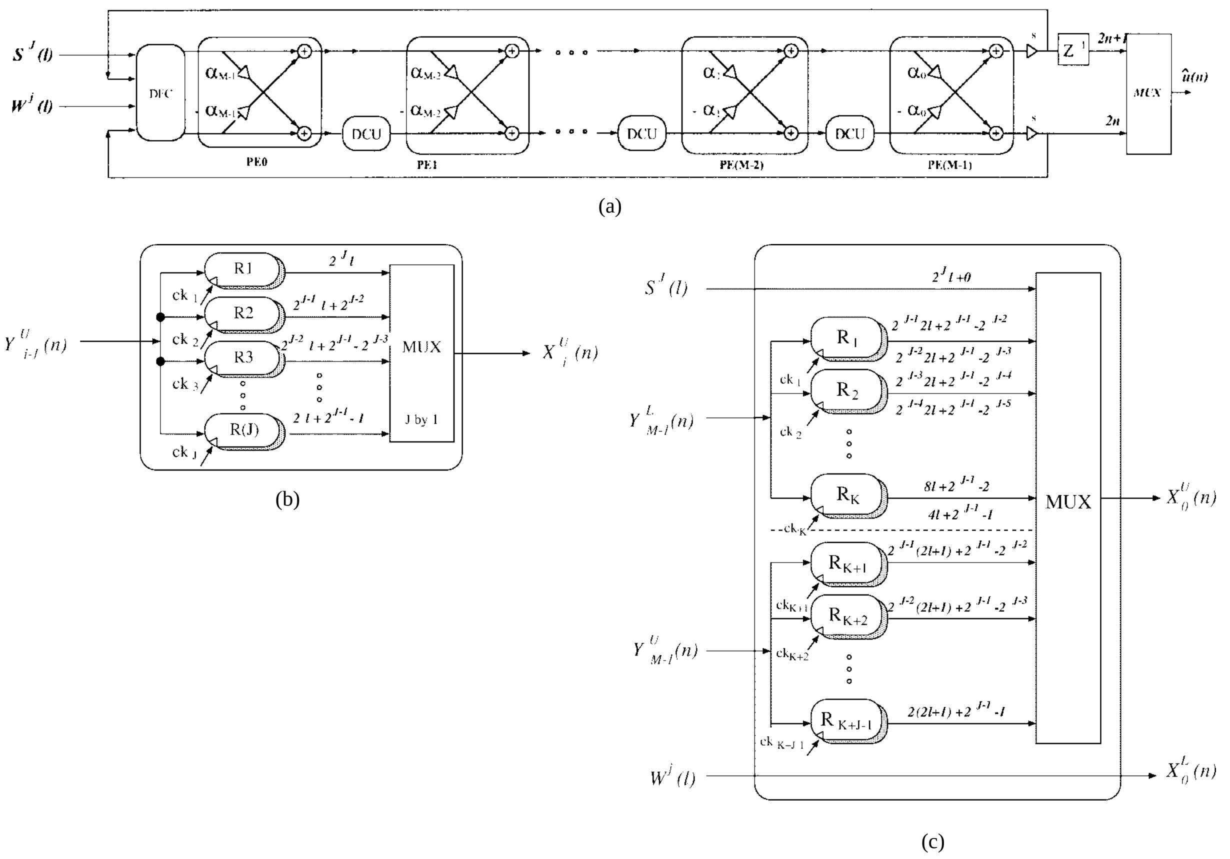


Fig. 19. (a) The proposed architecture for the inverse DWT. (b) The DCU structure with $ck_j$ given by (9). (c) The DFC structure with $ck_j$ given by (10), where $K = \lceil (J-1)/2 \rceil$.

instants of the output, this filter generates the outputs $S^{j-1}(2l)$ and $S^{j-1}(2l+1)$ simultaneously from the upper and lower levels of the lattice, respectively, and produces zeros at odd time instants of the output. By reusing this odd time instants for higher level computation, all resolution levels can be computed based on a single synthesis QMF lattice. One thing to be mentioned is that the feeding relations between adjacent levels are reverse compared to that of analysis stage: higher level should be computed earlier than lower one. Fig. 17 shows the DDG for the inverse DWT with three resolution levels. Here, the time slots $\{8l\}$, $\{4l+2\}$, and $\{2l+3\}$ are dedicated to the computation of levels 3, 2, and 1, respectively. Next we consider the scheduling problem for an arbitrary $J$.

Denote the filtering instants for the $j$th level of the inverse DWT by $\tau_j(l)$, $l = 0, 1, 2, \cdots$. Note that $l$th outputs of the $j$th resolution level, $S^{j-1}(2l+1)$ and $S^{j-1}(2l)$, are computed at time $\tau_j(l)$. Since the outputs of level $j$ should be produced every $2^j$ time unit, $\tau_j(l)$ should be expressed as $\tau_j(l) = 2^j \cdot l + \tau_j(0)$ where $\tau_j(0)$ is the instant at which the output of level $j$ is calculated for the first time. The constant $\tau_j(0)$ can be obtained with the help of a flipped binary tree, shown in Fig. 18, which successively combines two sets of nonoverlapped time slots. From this figure, we get the following expression for $\tau_j(l)$:

$$\tau_j(l) = 2^j \cdot l + 2^{J-1} - 2^{j-1}. \tag{6}$$

It can be seen that the filtering instants given by (6) never overlap with each other. Furthermore, this scheduling leads to the following observation.

*Observation 4:* If we schedule filtering operations for the inverse DWT based on (6), then the $l$th upper and lower outputs of level $j$, $S^{j-1}(2l+1)$ and $S^{j-1}(2l)$, are obtained before initializing the computations of $(2l+1)$th and $(2l)$th outputs of level $(j-1)$. Specifically, $S^{j-1}(2l+1)$ and $S^{j-1}(2l)$, respectively, are produced $(2^j - 2^{j-2})$ and $(2^{j-1} - 2^{j-2})$ time units before the evaluation of $(2l+1)$th and $(2l)$th outputs of the $(j-1)$th level.

*Proof:* The first part is proved by showing that $\tau_j(l) < \tau_{j-1}(2l) < \tau_{j-1}(2l+1)$. Here the second inequality is obvious. Now $\tau_{j-1}(2l) = 2^{j-1} \cdot (2l) + 2^{J-1} - 2^{j-2} = 2^j \cdot l + 2^{J-1} - 2^{j-1} + (2^{j-1} - 2^{j-2}) > 2^j \cdot l + 2^{J-1} - 2^{j-1} = \tau_j(l)$. From this relation, the second part directly follows. □

Using the notation in Fig. 6, input–output relations between lattices for the synthesis QMF bank can be derived as follows: for the first PE $(\mathrm{PE}_0)$

$$X_0^U(n) = \begin{cases} S^J(l), & \text{for } n = 2^J \cdot l \\ Y^U_{M-1}(n - 2^j - 2^{j-2}), & \text{for } n = 2^j \cdot (2l+1) + 2^{J-1} - 2^{j-1} \\ Y^U_{M-1}(n - 2^{j-1} - 2^{j-2}), & \text{for } n = 2^j \cdot (2l) + 2^{J-1} - 2^{j-1} \end{cases} \tag{7a}$$

and

$$X_0^L(n) = W^j(l) \quad \text{for } n = 2^j \cdot l + 2^{J-1} - 2^{j-1} \tag{7b}$$

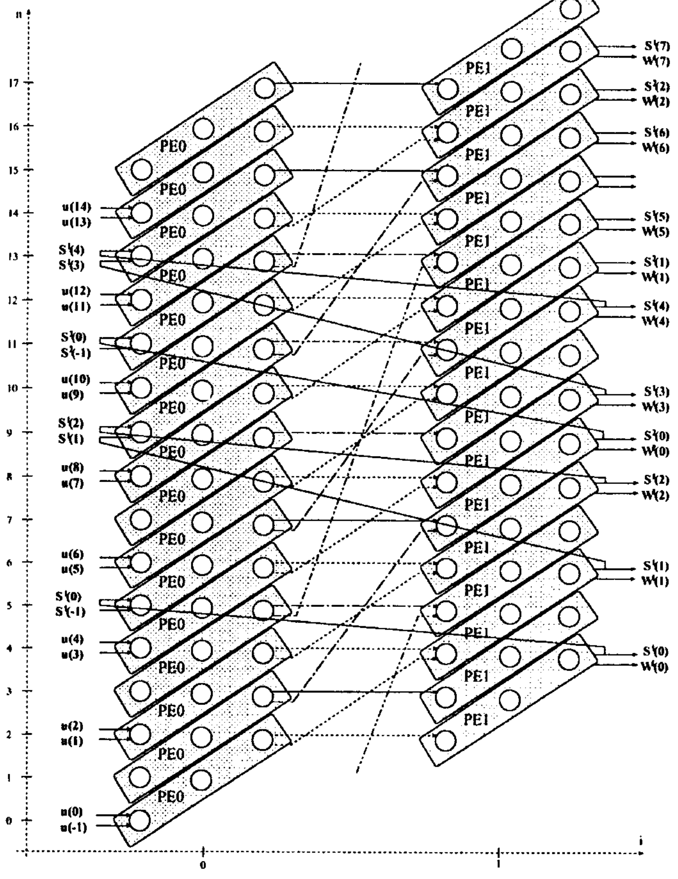


Fig. 20. Data dependence graph for pipelined DWT with $J = 3$, $M = 2$, and $P_d = 2$.

where $J$ is the number of resolution levels. For the $i$th PE (PE$_i$), $1 \leq i \leq M-1$,

$$X_i^U(n) = Y_{i-1}^U(n - 2^j), \quad \text{for } n = 2^j \cdot l + 2^{J-1} - 2^{j-1} \tag{8a}$$

$$X_i^L(n) = Y_{i-1}^L(n). \tag{8b}$$

These relations lead to the architecture shown in Fig. 19(a). The structures for the DCU and DFC shown in Figs. 19(b) and (c) are obtained by using the methods described in the previous section. The instants $ck_j$ at which the clock signal for data loading is applied to registers of the DCU and DFC are expressed as follows for the DCU:

$$ck_j = \tau_{J-j+1}(l) + 1 \tag{9}$$

and for the DFC

$$ck_k = \begin{cases} \{\tau_{2k-1}(l)+1\} \oplus \{\tau_{2k}(l)+1\}, \\ \qquad \text{when } 1 \leq k \leq K \\ \{\tau_{k-K}(l)+1\}, \\ \qquad \text{when } K < k \leq K+J-1 \end{cases} \tag{10}$$

where $K = \lceil (J-1)/2 \rceil$, and $\{\tau_{2k-1}(l)+1\} \oplus \{\tau_{2k}(l)+1\}$ implies that the clock is applied both at $\{\tau_{2k-1}(l)+1\}$ and $\{\tau_{2k}(l)+1\}$. As in the case of the forward DWT, the proposed architecture is scalable and requires less hardware than the existing architectures.

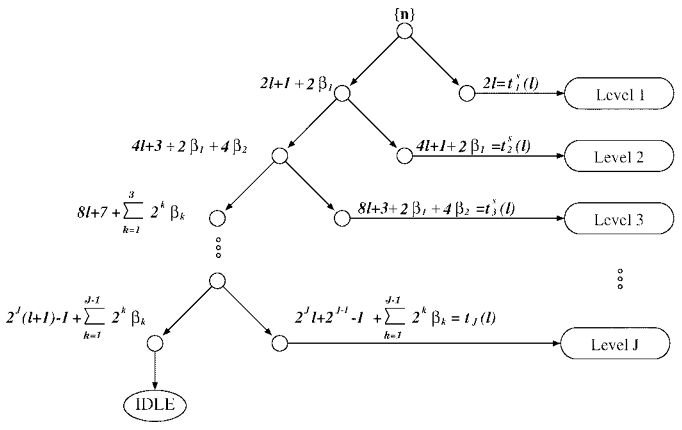

{n}
$2l+1+2\beta_1$
$2l=t_1^s(l)$
Level 1
$4l+3+2\beta_1+4\beta_2$
$4l+1+2\beta_1=t_2^s(l)$
Level 2
$8l+7+\sum_{k=1}^{3} 2^k\beta_k$
$8l+3+2\beta_1+4\beta_2=t_3^s(l)$
Level 3
$2^J(l+1)-1+\sum_{k=1}^{J-1} 2^k\beta_k$
$2^J l+2^{J-1}-1+\sum_{k=1}^{J-1} 2^k\beta_k=t_J(l)$
Level J
IDLE

Fig. 21. A binary tree for scheduling the pipelined DWT architecture, where $\beta$ is given by (11b).

## IV. Pipelining the DWT Architecture

The architectures shown in Figs. 15 and 19 may suffer from long critical path in high-speed computations. This problem can be overcome through pipelining, as shown below.

Let $P_d$ be the number of pipelining stages of each PE, and $M$ be the number of lattices. The pipelining introduces $M \cdot P_d$ time unit latency to the QMF lattice. The computation for the $l$th output of the $j$th level, $S^j(l)$, should be started at least $M \cdot P_d$ time units after starting the computation for $S^{j-1}(2l)$. Details of scheduling can be seen from the DDG in Fig. 20, which illustrates the data dependency for $J = 3$, $P_d = 2$, and $M = 2$. Filtering of the first level is initiated at $n = 2l$, $l = 0, 1, 2, \cdots$. Note that the computation of $S^2(l)$ can be started after $S^l(2l)$ is available. Since $S^1(2l)$ is available at $n = 4l+4$, we can compute $S^2(l)$ right after this time. However, this time instant is occupied for computing $S^1(2l+2)$, and thus $S^2(l)$ is computed at $n = 4l+5$. In a similar manner, we can see that $S^3(l)$ can be computed at $n = 8l+11$. Here, the time slots $\{2l\}$, $\{4l+5\}$, and $\{8l+11\}$ never overlap with each other. Now we extend this result to the DWT with an arbitrary $J$. Suppose that the computation of $S^j(l)$ is started at $t_j^s(l)$ and ended at $t_j^e(l)$. Then $t_j^e(l) = t_j^s(l) + M \cdot P_d$. In the binary tree scheduling illustrated in Fig. 21, the nonnegative integer set $\{n\}$ is decomposed into $\{2l\}$ and $\{2l+1+2\beta_1\}$. Here $\beta_1$ is the smallest integer satisfying $2l+1+2\beta_1 \geq 2l + M \cdot P_d$. The even set $\{2l\}$ provides the starting instants for first level computation, i.e., $t_1^s(l) = 2l$. The remaining set is decomposed into $\{4l+1+2\beta_1\}$ and $\{4l+3+2\beta_1+4\beta_2\}$, where $\beta_2$ is the smallest intgeger satisfying $4l+3+2\beta_1+4\beta_2 \geq 4l+1+2\beta_1+M\cdot P_d$. Continuing in this manner, we can get the following expression for $t_j^s(l)$:

$$t_j^s(l) = 2^j \cdot l + 2^{j-1} - 1 + \sum_{k=1}^{j-1} 2^k \beta_k \tag{11a}$$

where $\beta_k$ is the smallest nonnegative integer satisfying

$$\beta_k \geq \frac{M \cdot P_d - 2^{k-1}}{2^k}. \tag{11b}$$

It can be seen that the filtering instants given by (11a) never overlap with each other. The observation below shows the validity of this scheduling.

TABLE II
SCHEDULING AND CORRESPONDING LATENCY TIME FOR THE VARIATION OF $P_d$ WITH $J = 4$ AND $M = 3$

| $M = 3$ | $\beta_i$ | | | $t_j^s(l)$ | | | | latency |
|---|---|---|---|---|---|---|---|---|
| | $\beta_1$ | $\beta_2$ | $\beta_3$ | $t_1^s(l)$ | $t_2^s(l)$ | $t_3^s(l)$ | $t_4^s(l)$ | |
| $P_d = 0$ | 0 | 0 | 0 | $2l + 0$ | $4l + 1$ | $8l + 3$ | $16l + 7$ | 7 |
| $P_d = 1$ | 2 | 1 | 0 | $2l + 0$ | $4l + 3$ | $8l + 9$ | $16l + 13$ | 16 |
| $P_d = 2$ | 3 | 1 | 1 | $2l + 0$ | $4l + 7$ | $8l + 13$ | $16l + 25$ | 31 |
| $P_d = 3$ | 4 | 2 | 1 | $2l + 0$ | $4l + 9$ | $8l + 19$ | $16l + 31$ | 40 |
| $P_d = 4$ | 6 | 3 | 1 | $2l + 0$ | $4l + 13$ | $8l + 27$ | $16l + 39$ | 51 |
| $P_d = 5$ | 7 | 4 | 2 | $2l + 0$ | $4l + 15$ | $8l + 33$ | $16l + 53$ | 68 |

*Observation 5:* For a given $J$, $M$, and $P_d$, if we schedule filtering based on (11a), then $S_{j-1}(2l)$ and $S_{j-1}(2l-1)$ are available when we initiate the computation of $S_j(l)$.

*Proof:* We can prove this observation by showing $t^e_{j-1}(2l) \le t^s_j(l)$. $t^s_j(l) - t^e_{j-1}(2l) = (2^j \cdot l + 2^{j-1} - 1 + \sum_{k=1}^{j-1} 2^k\beta_k) - (2^{j-1}(2l) + 2^{j-2} - 1 + \sum_{k=1}^{j-2} 2^k\beta_k + M\cdot P_d) = 2^{j-2} + 2^{j-1}\beta_{j-1} - M\cdot P_d \ge 2^{j-2} + (M\cdot P_d - 2^{j-2}) - M\cdot P_d \ge 0$. □

From (11a), input–output relations between lattices can be derived directly. The results are summarized as follows. For the first PE $(\mathrm{PE}_0)$

$$X_0^U(n) = \begin{cases} u(n), \\ \quad \text{for } n = 2l \\ Y_{M-1}^U(n - 2^{j-2} + 2^{j-1}\beta_{j-1} - M\cdot P_d), \\ \quad \text{for } n = t_j^s(l) \end{cases} \tag{12a}$$

and

$$X_0^L(n) = \begin{cases} u(n-1), \\ \quad \text{for } n = 2l \\ Y_{M-1}^U(n - 2^{j-1} - 2^{j-2} + 2^{j-1}\beta_{j-1} - M\cdot P_d), \\ \quad \text{for } n = t_j^s(l). \end{cases} \tag{12b}$$

For the $i$th PE $(\mathrm{PE}_i)$

$$X_i^U(n) = Y_{i-1}^U(n), \tag{13a}$$
$$X_i^L(n) = Y_{i-1}^L(n - 2^j), \quad \text{for } n = t_j^s(l) + i\cdot M\cdot P_d. \tag{13b}$$

These relations lead to the DCU and DFC structures in Figs. 11 and 14; but the instants $ck_j$ for data loading should be changed due to pipelining. Specifically, for the DCU between $\mathrm{PE}_i$ and $\mathrm{PE}_{i+1}$

$$ck_j = 2^j \cdot l + 2^{j-1} + i \cdot P_d \tag{14}$$

and for the DFC

$$ck_j = \begin{cases} \{\tau^e_{2j-1}(2l) + 1\} \oplus \{\tau^e_{2j}(2l) + 1\}, \\ \quad \text{for } 1 \le j \le K \\ \{\tau^e_j(2l-1) + 1\}, \\ \quad \text{for } K < j \le K + J - 1 \end{cases} \tag{15}$$

where $K = \lceil (J-1)/2 \rceil$ and $\{\tau^e_{2j-1}(2l) + 1\} \oplus \{\tau^e_{2j}(2l) + 1\}$ implies that the clock is applied both at $\{\tau^e_{2j-1}(2l) + 1\}$ and $\{\tau^e_{2j}(2l) + 1\}$.

Finally, we derive the latency time caused by the pipelining stages. The latency time of the pipelined DWT architecture is written as $t^e_J(0) - t^s_1(0)$. Since $t^s_1(l) = 0$, the latency time is given by

$$t^e_J(0) = 2^{J-1} - 1 + \sum_{k=1}^{J-1} 2^k\beta_k + M\cdot P_d. \tag{16}$$

The latency time is a function of $J$, $M$, and $P_d$. Table II tabulates the latency time when $M = 3$ and $J = 4$. The latency time in (16) is useful for examining the tradeoff between the latency and throughput of the pipelined DWT architecture. Pipelining of the architecture for the inverse DWT can be done in a similar manner, and will not be considered here due to space limitation.

## V. SUMMARY AND CONCLUSION

In this paper, we developed a scalable VLSI architecture employing a two-channel QMF lattice for the 1-D DWT. Based on the development of a systematic scheduling, the input-output relation between lattices of the QMF bank has been derived, and new structures for the DFC and the DCU have been proposed. The proposed structures are regular, scalable, and require a minimum number of registers, and thereby lead to an efficient and scalable architecture for the DWT. An architecture for the inverse DWT has been also developed in a similar manner, and finally, pipelining of the proposed architecture has been considered. Future work in this direction will be concentrated on the design of two-dimensional (2-D) DWT and $M$-ary tree structured filter banks.

## REFERENCES

[1] R. Kronland-Martinet, J. Morlet, and A. Grossmann, "Analysis of sound patterns through wavelet transforms," *Int. J. Pattern Recogn. Artificial Intell.*, vol. 1, pp. 273–302, Feb. 1987.
[2] S. G. Mallet, "A theory for multiresolution signal decomposition: The wavelet representation," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 19, pp. 674–693, July 1989.
[3] ——, "Multifrequency channel decompositions of images and wavelet models," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-37, pp. 2091–2110, Dec. 1989.
[4] M. Antonini, M. Barlaud, P. Mathieu, and I. Daubechies, "Image coding using vector quantization in the wavelet transform domain," in *Proc. 1990 IEEE Int. Conf. Acoust., Speech, Signal Processing*, Apr. 1990, pp. 2297–2300.
[5] M. Vetterli and C. Herley, "Wavelets and filter banks: Theory and design," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. 40, pp. 2207–2232, Sept. 1992.
[6] M. J. T. Smith and T. P. Barnwell, "Exact reconstruction for tree-structured subband coders," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-34, pp. 434–441, June 1986.
[7] T. Q. Nguyen and P. P. Vaidyanathan, "Maximally decimated perfect-reconstruction FIR filter banks with pairwise mirror image analysis (and synthesis)," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-36, pp. 693–706, May 1988.
[8] P. P. Vaidyanathan, *Multirate Systems and Filter Banks.* Englewood Cliffs, NJ: Prentice-Hall, 1993.
[9] G. Knowles, "VLSI architecture for the discrete wavelet transform," *Electron. Lett.*, vol. EL-26, pp. 1184–1185, July 1990.
[10] A. S. Lewis and G. Knowles, "VLSI architecture for 2-D Daubechies wavelet transform without multiplier," *Electron. Lett.*, vol. EL-27, pp. 171–173, Jan. 1991.
[11] K. K. Parhi, "VLSI architectures for discrete wavelet transform," *IEEE Trans. VLSI Syst.*, vol. 1, pp. 191–202, June 1993.
[12] M. Vishwanath, R. M. Owens, and M. J. Irwin, "VLSI architectures for the discrete wavelet transform," *IEEE Trans. Circuits Syst. II*, vol. 42, pp. 305–316, May 1995.

[13] J. Fridman and E. S. Manolakos, “Distributed memory and control VLSI architectures for 1-D discrete wavelet transform,” in *IEEE VLSI Signal Processing*, pp. 303–312, 1994.
[14] T. C. Denk and K. K. Parhi, “Architectures for lattice structure based orthonormal discrete wavelet transforms,” *IEEE Trans. Circuits Syst. II*, vol. 44, pp. 129–132, Feb. 1997.
[15] ——, “Systematic design of architectures for $M$-ary tree-structured filter bank,” in *Proc. 1995 IEEE Workshop VLSI Signal Processing*, Oct. 1995, pp. 157–166.
[16] P. P. Vaidyanathan and P. Hoang, “Lattice structures for optimal design and robust implementation of two-channel perfect reconstruction QMF banks,” *IEEE Trans. Acoust. Speech, Signal Processing*, vol. ASSP-36, pp. 506–526, Jan. 1988.
[17] ——, “Multirate digital filters, filter banks, polyphase networks, and applications: A tutorial,” *Proc. IEEE*, Jan. 1990, vol. 78, pp. 56–93.
[18] K. K. Parhi, “Systematic synthesis of DSP data format converters using life-time analysis and forward-backward register allocation,” *IEEE Trans. Circuits Syst. II*, vol. 39, pp. 423–440, July 1992.
[19] S. Y. Kung, *VLSI Array Processors.* Englewood Cliffs, NJ: Prentice-Hall, 1988.

**Joon Tae Kim** was born in Seoul, Korea, on September 27, 1967. He received the B.S.E.E. and M.S.E.E. degrees from Korea Advanced Institute of Science and Technology (KAIST), Taejon, Korea, in 1991 and 1993, respectively. He is currently working toward the Ph.D. degree in electrical engineering at KAIST.

His research interests include digital signal processing, digital communication, and VLSI signal processing.

**Yong Hoon Lee** was born in Seoul, Korea, on July 12, 1955. He received the B.S. and M.S. degrees in electrical engineering from Seoul National University, Seoul, Korea, in 1978 and 1980, respectively, and the Ph.D. degree in systems engineering from the University of Pennsylvania, Philadelphia, in 1984.

From 1984 to 1988, he was an Assistant Professor with the Department of Electrical and Computer Engineering, State University of New York, Buffalo. Since 1989, he has been with the Department of Electrical Engineering, Korea Advanced Institute of Science and Technology (KAIST), where he is a Professor. His research activities are in the areas of 1-D and 2-D digital signal processing, VLSI signal processing, and digital communication systems.

**Tsuyoshi Isshiki** received the Bachelor’s and Master’s degrees in electrical and electronics engineering from Tokyo Institute of Technology in 1990 and 1992, respectively, and the Ph.D. degree in computer engineering from the University of California at Santa Cruz in 1996.

He is currently a Research Associate in Electrical and Electronics Engineering at Tokyo Institute of Technology. His research interests include high-level design methodology for configurable systems, bit-serial synthesis, FPGA architecture, image processing, computer graphics, and speech synthesis.

**Hiroaki Kunieda** was born in Yokohama in 1951. He received the B.E., M.E., and Dr.Eng. degrees from Tokyo Institute of Technology in 1973, 1975, and 1978, respectively.

He was a Research Associate in 1978 and an Associate Professor in 1985, at Tokyo Institute of Technology. He is currently a Professor in the Department of Electrical Engineering, Tokyo Institute of Technology. He has been engaged in research on distributed circuits, switched capacitor circuits and IC circuit simulation. His current research focuses on VLSI signal processing, Parallel processing, VLSI microarchitecture design, VLSI CAD, and parallel image processing.

Dr. Kunieda is a member of the IEEE Circuits and Systems and IEEE Signal Processing societies, and IPSJ.