On LUT Cascade Realizations of FIR Filters

Tsutomu Sasao [1]   Yukihiro Iguchi [2]   Takahiro Suzuki [2]

[1] Kyushu Institute of Technology, Dept. of Comput. Science & Electronics, Iizuka 820-8502, Japan
[2] Meiji University, Dept. of Computer Science, Kawasaki 214-8571, Japan

Abstract

This paper first defines the n-input q-output WS function, as a mathematical model of the combinational part of the distributed arithmetic of a finite impulse response (FIR) filter. Then, it shows a method to realize the WS function by an LUT cascade with k-input q-output cells. Furthermore, it 1) shows that LUT cascade realizations require much smaller memory than the single ROM realizations; 2) presents new design method for a WS function by arithmetic decomposition, and 3) shows design results of FIR filters using FPGAs with embedded memories.

Keywords: Digital filter, Distributed arithmetic, LUT cascade, Functional decomposition, Binary decision diagram, FPGA.

1   Introduction

Digital filters are important elements in signal processing [9], and can be classified into two types: FIR (Finite Impulse Response) filters and IIR (Infinite Impulse Response) filters. FIR filters have nonrecursive structure, and so always have stable operations. Also, FIR filters can have linear phase characteristics, so they are useful for waveform transmission.

To realize FIR filters on FPGAs, we can use Distributed Arithmetic to convert the multiply-accumulation operations into table-lookup operations [7, 13, 14]. For table lookup, embedded memory blocks in FPGAs can be used. In this paper, we propose a method to implement the combinational part of the FIR filter by an LUT cascade, shown in Fig. 3.1, a series connection of memories. The LUT cascade realizations require much smaller memory than a single memory realization. Our method is useful in the design of FIR filters by embedded memories in FPGAs[2], or dedicated LUT cascade chips [8].

The rest of the paper is organized as follows. Section 2 introduces FIR filters. Section 3 introduces LUT cascades
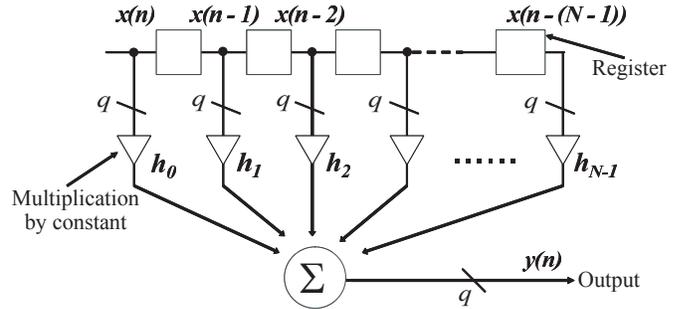
Figure 2.1. Parallel Realization of FIR Filter.

and functional decomposition. Section 4 defines the WS function, and shows its realization by an LUT cascade. Section 5 introduces the arithmetic decomposition of WS functions, Section 6 shows experimental results, and Section 7 concludes the paper.

2   FIR Filter

Definition 2.1 *The* **FIR filter** *computes*

$$\mathcal{Y}(n) = \sum_{i=0}^{N-1} h_i \cdot \mathcal{X}(n-i), \qquad (2.1)$$

where $\mathcal{X}(i)$ is the value of the input $\mathcal{X}$ at the time i, and $\mathcal{Y}(i)$ is the value of the output $\mathcal{Y}$ at the time i [1] . h_i is a **filter coefficient** *represented by a q-bit fixed-point binary number, and N is the* **number of taps** *in the filter [2] .*

Fig. 2.1 implements (2.1) directly. It consists of an N-stage q-bit shift register, N copies of q-bit multipliers, and an adder for N q-bit numbers. To reduce the amount of hardware in Fig. 2.1, we use the bit-serial method shown in Fig. 2.2. Here, PSC denotes the parallel to series con-

---

[1] In this paper, $\mathcal{X}$ and $\mathcal{Y}$ denotes the values of signal in the filters, x_i denotes a logic variable, $\vec{X}_1$ and $\vec{X}_2$ denote the vectors of logic variables.

[2] In general, the number of bits for h_i and $\mathcal{Y}$ can be different. However, for simplicity, we assume that they are both represented by q bits.
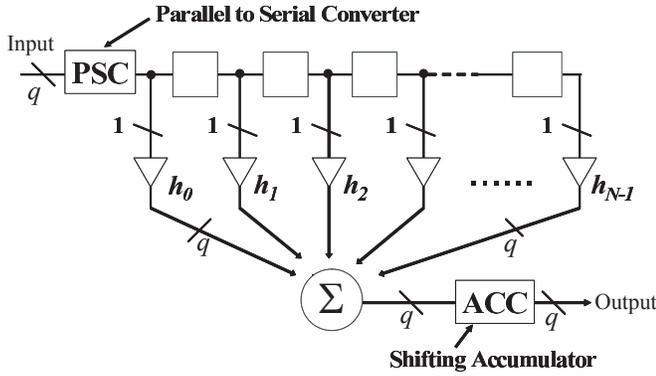
Figure 2.2. Series Realization of FIR Filter.

Figure 2.4. Series-and-Symmetric Realization of FIR Filter.

Figure 2.3. Realization of ACC.
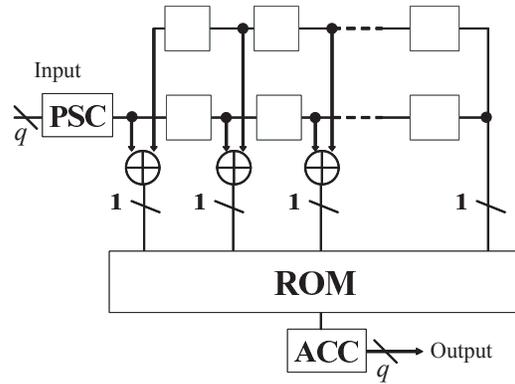
Figure 2.5. FIR Filter (Series and Symmetric ROM Realization).

verter. In this case, the inputs to $h_0, h_1, \ldots, h_{N-1}$ are either 0 or 1, so the multipliers can be replaced by AND gates. In Fig. 2.2, ACC denotes the shifting accumulator, which accumulates the numbers while doing shifting operations. The ACC can be implemented by, for example, the network in Fig. 2.3.

This method reduces the amount of hardware to $1/q$, but increases the computation time q times. The combinational part in Fig. 2.2 has N-inputs and q-outputs. This part realizes the WS function, which will be defined later. In FIR filters that have linear phase characteristics, coefficients satisfy the relation $h_i = h_{N-i-1}$. Such a filter is **symmetric**. A symmetric filter can be implemented by Fig. 2.4. It requires less hardware than Fig. 2.2. In this case, we use an $(N + 1)/2$-input adder for q-bit numbers. The $\oplus$ symbol in Fig. 2.4 denotes a serial adder [3].

In Fig. 2.5, the combinational part is implemented by a ROM. For example, when the number of inputs is three, the ROM stores the precomputed values shown in Table 2.1. Note that this is an example of a WS function defined later.

This method is called **Distributed Arithmetic**, and is

often used to implement convolution operations, since many multipliers and multi-input adders can be replaced by one memory [1, 3, 6, 7, 13, 14]. This method is applicable only when the coefficients h_i are constants. In FIR filters, the coefficients h_i are constants, so we can apply this method. Note that a ROM with n-inputs and q-outputs requires $q2^n$ bits.

3 LUT Cascade and Functional Decomposition

This chapter describes a relationship between LUT cascades and functional decompositions.

Definition 3.1 *An* **LUT cascade** *realizes a given logic function by the network structure shown in Fig. 3.1. Each block is called a* **cell**, *and each cell realizes an arbitrary logic function. The signal lines connecting adjacent cells are called* **rails**.

---

[3] At the beginning of the serial addition, we have to clear the register and flip-flops by adding the reset signals.

Table 2.1. Example of WS Function.

Address $x_2 x_1 x_0$	Data
000	0
001	h_0
010	h_1
011	$h_0 + h_1$
100	h_2
101	$h_0 + h_2$
110	$h_1 + h_2$
111	$h_0 + h_1 + h_2$
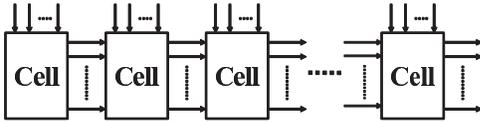
Figure 3.1. LUT Cascade.

Definition 3.2 $\vec{F}(\vec{X})$ *has a (multi-output)* **functional decomposition** *if* $\vec{F}(\vec{X})$ *is to represented as* $\vec{F} = g(h(\vec{X}_1), \vec{X}_2)$. *In this case, the variables in* $\vec{X}_1$ *are* **bound variables***, and the variables in* $\vec{X}_2$ *are* **free variable***. Note that h is a multiple-valued function* [4] .

Definition 3.3 *In the* **decomposition chart** *of* $\vec{F}(\vec{X}_1, \vec{X}_2)$, *variables values of* $\vec{X}_1$ *label columns, and variables values of* $\vec{X}_2$ *label rows. The values of* $\vec{F}(\vec{X}_1, \vec{X}_2)$ *are the entries of the chart. In the case of a q-output function, each element of the chart is a q-bit vector. The number of different column patterns is the* **column multiplicity***.

Example 3.1 *Table 3.1 shows an example of a decomposition chart for a 5-input 3-output logic function, where* $\vec{X}_1 = (x_0, x_1, x_2)$ *and* $\vec{X}_2 = (x_3, x_4)$. *In this case, the column multiplicity is five.* ∎

Theorem 3.1 *Let* $\vec{F}(\vec{X}) = g(h(\vec{X}_1), \vec{X}_2)$, *and let the column multiplicity of the decomposition chart be* μ. *Then,*

Table 3.1. Example of Decomposition Chart.

	$\vec{X}_1 = (x_0, x_1, x_2)$							
	000	001	010	011	100	101	110	111
$\vec{X}_2 = (x_3, x_4)$ 00	001	010	011	100	001	100	100	100
01	111	110	000	011	111	010	010	011
10	010	110	010	001	010	010	010	001
11	010	011	101	010	010	011	011	010

---

[4] [4] only considers the case where h is a two-valued function. [5] only considers the case where $\vec{F}$ is a single-output function.
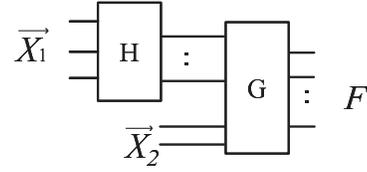
Figure 3.2. Functional Decomposition.

$\vec{F}(\vec{X})$ *is realized by the structure shown in Fig. 3.2. The number of signal lines between two block H and G is at most* $\lceil \log_2 \mu \rceil$.

Theorem 3.2 *Let* $\vec{X} = (x_0, x_1, \ldots, x_{n-1})$ *be the variables of function* $\vec{F}$. *If the column multiplicity of the decomposition chart for* $\vec{F}(\vec{X}_1, \vec{X}_2)$ *is at most* 2^q *for all* i $(q < i < n-1)$, *where* $\vec{X}_1 = (x_0, x_1, \ldots, x_i)$ *and* $\vec{X}_2 = (x_{i+1}, \ldots, x_{n-1})$, *then* $\vec{F}$ *is realized by the LUT cascade with k-input q-output cells, where* $k > q$.

(Proof) Let $\vec{X}_1 = (x_0, x_1, \ldots, x_{k-1})$ and $\vec{X}_2 = (x_k, \ldots, x_{n-1})$. By decomposing $\vec{F}(\vec{X})$, we have the network in Fig. 3.2. Let q be the number of signal lines between two blocks. Then, G is an $(n-k+q)$-input network. Apply the functional decomposition to G again. In this case, include all the the intermediate variables (i.e., the output signals of H) in the bound variables. Note that, the column multiplicity of the decomposition chart is also at most 2^q. By applying the functional decompositions repeatedly, we form the cascade shown in Fig. 3.1. □

4 WS Function and Its LUT Cascade Realization

Definition 4.1 *An* n-*input* q-*output* **WS function**[11] $\vec{F}(x_0, x_1, \ldots, x_{n-1})$ *computes*

$$\sum_{i=0}^{n-1} h_i \cdot x_i \qquad (4.1)$$

and represent a value as a q-bit binary number [5] . *Here,* h_i *denotes a coefficient represented as a q-bit binary number. We use the fixed-point 2's complement representation, and assume that q includes one bit for the sign. Let the binary representation of* h_i *be* $(\vec{h}_i)_2$, *then* $\vec{F}$ *satisfies relations:*

$$\vec{F}(1, 0, \ldots, 0) = (\vec{h}_0)_2$$
$$\vec{F}(0, 1, \ldots, 0) = (\vec{h}_1)_2$$
$$\vdots$$
$$\vec{F}(0, 0, \ldots, 1) = (\vec{h}_{n-1})_2.$$

---

[5] For non-symmetric filter $n = N$, and for symmetric filter $n = (N+1)/2$

Note that the WS function is generated first by rounding the coefficients into q bits, and then adding the coefficients.

The next lemma shows that the column multiplicity of the decomposition chart for any q-output WS functions is at most 2^q.

Lemma 4.1 *Let $\vec{F}(x_0, x_1, \ldots, x_{n-1})$ be an n-input q-output WS function. Let $(\vec{X}_1, \vec{X}_2)$ be a partition of $\vec{X} = (x_0, x_1, \ldots, x_{n-1})$, where $\vec{X}_1 = (x_0, x_1, \ldots, x_{i-1})$ and $\vec{X}_2 = (x_i, x_{i+1}, \ldots, x_{n-1})$. Consider the decomposition chart of $\vec{F}$, where $\vec{X}_1$ are the bound variables and $\vec{X}_2$ are the free variables. The column multiplicity of the decomposition chart is at most 2^q.*

(Proof) When $i \leq q$, the column multiplicity is at most 2^q. So, we will examine the case of $i > q$. Consider the first row of the decomposition chart, i.e., the row for $\vec{X}_2 = (0, 0, \ldots, 0)$. The number of different elements is at most 2^q, since each element of the decomposition chart is a vector of q bits. Thus, for two different vectors $\vec{a}, \vec{b} \in \{0,1\}^i$, there exist two columns that correspond to the assignments $\vec{a}, \vec{b}$ to $\vec{X}_1$ such that $\vec{F}(\vec{a}, \vec{0}) = \vec{F}(\vec{b}, \vec{0})$.

Next, consider the j-th row ($j > 0$). Let $\vec{c}$ be the value of $\vec{X}_2 = (x_i, x_{i+1}, \ldots, x_{n-1})$. Then, by Definition 4.1, $\vec{F}$ satisfies the relation

$$\vec{F}(\vec{a}, \vec{c}) = \vec{F}(\vec{a}, \vec{0}) + \vec{F}(\vec{0}, \vec{c})$$
$$\vec{F}(\vec{b}, \vec{c}) = \vec{F}(\vec{b}, \vec{0}) + \vec{F}(\vec{0}, \vec{c}),$$

where the symbol $+$ denotes the integer addition of binary numbers. Therefore, we have the relation: $\vec{F}(\vec{a}, \vec{c}) = \vec{F}(\vec{b}, \vec{c})$. Since this relation holds for all $j > 0$, two column patterns that correspond to vectors $\vec{a}$ and $\vec{b}$ are the same.

From the above, we can show that the column multiplicity of the decomposition chart is at most 2^q. □

Example 4.1 *Table 4.1 shows an example of a decomposition chart for $n = 5$, where $\vec{X}_1 = (x_0, x_1, x_2)$ and $\vec{X}_2 = (x_3, x_4)$. Suppose that $q = 2$, that is, each element is a binary vector of two bits. In this case, only four different vectors can exist. So, in the first row of the decomposition chart, the row for $(x_3, x_4) = (0, 0)$, at least two elements are equal. Suppose that the values for the columns $(x_0, x_1, x_2) = (0, 1, 1)$ and $(x_0, x_1, x_2) = (1, 0, 0)$ are equal: $h_1 + h_2 = h_0$. This implies that in the second row of the decomposition table, that is in the row for $(x_3, x_4) = (0, 1)$, the corresponding two elements are equal: $h_1 + h_2 + h_4 = h_0 + h_4$. This is obvious since the same numbers are added to the both elements. In similar ways, we can show that the remaining rows, the entries for the columns $(x_0, x_1, x_2) = (0, 1, 1)$ and $(x_0, x_1, x_2) = (1, 0, 0)$ are equal. That is, if the two elements in the first*
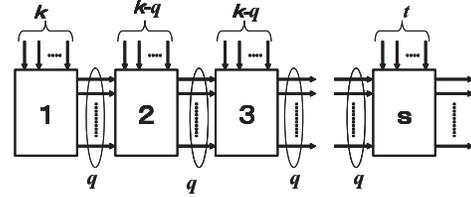
Figure 4.1. LUT Cascade Realization of WS Function.

row are equal, then the patterns of the two columns are the same. Hence, we can see that the column patterns for $(x_0, x_1, x_2) = (0, 1, 1)$ and $(x_0, x_1, x_2) = (1, 0, 0)$ are the same. ■

Theorem 4.1 *An arbitrary q-output WS function can be implemented by an LUT cascade consisting of cells with $(q + 1)$ inputs and q outputs.*

(Proof) By Lemma 4.1 and Theorem 3.2, the number of lines between two blocks is q. We need at least one external input for each cell. □

Theorem 4.2 *An arbitrary n-input q-output WS function can be implemented by an LUT cascade of at most $\lceil \frac{n - k - 1}{k - q} \rceil + 2$ cells with k inputs and q outputs.*

(Proof) When we implement the WS function by the method of Theorem 3.2, we have the LUT cascade shown in Fig. 4.1. Let q be the number of rails in the cascade, and s be the number of cells. Let t be the number of inputs to the final cell, then we have the relations $k + (s - 2)(k - q) + t = n$ and $1 \leq t \leq k - q$. From these, we have $s \leq \lceil \frac{n - k - 1}{k - q} \rceil + 2$. □

Since a k-input q-output cell requires $q2^k$ bits, we have the following:

Corollary 4.1 *To implement an n-input q-output WS function by an LUT cascade with k-input q-output cells, we need at most $q(\lceil \frac{n - k - 1}{k - q} \rceil + 2)2^k$ bits.*

Lemma 4.2 *The ROM for an n-input q-output WS function requires $q2^n$ bits.*

Corollary 4.2 *Assume that an LUT cascade uses k-input q-output cells, then the ratio for the amount of memory for the LUT cascade to the ROM for an n-input q-output WS function is $(\lceil \frac{n - k - 1}{k - q} \rceil + 2)2^{k-n}$.*

Table 4.1. Example of Decomposition Chart for WS Function.

<table>
<tr><td colspan="9" style="text-align:center">$\vec{X}_L = (x_0, x_1, x_2)$</td></tr>
<tr><td></td><td>000</td><td>001</td><td>010</td><td>011</td><td>100</td><td>101</td><td>110</td><td>111</td></tr>
<tr><td>00</td><td>0</td><td>h_2</td><td>h_1</td><td>$h_1 + h_2$</td><td>h_0</td><td>$h_0 + h_2$</td><td>$h_0 + h_1$</td><td>$h_0 + h_1 + h_2$</td></tr>
<tr><td>01</td><td>h_4</td><td>h_2+ h_4</td><td>h_1+ h_4</td><td>$h_1 + h_2+$ h_4</td><td>h_0+ h_4</td><td>$h_0 + h_2+$ h_4</td><td>$h_0 + h_1+$ h_4</td><td>$h_0 + h_1 + h_2+$ h_4</td></tr>
<tr><td>10</td><td>h_3</td><td>h_2+ h_3</td><td>h_1+ h_3</td><td>$h_1 + h_2+$ h_3</td><td>h_0+ h_3</td><td>$h_0 + h_2+$ h_3</td><td>$h_0 + h_1+$ h_3</td><td>$h_0 + h_1 + h_2$ h_3</td></tr>
<tr><td>11</td><td>$h_3 + h_4$</td><td>h_2+ $h_3 + h_4$</td><td>h_1+ $h_3 + h_4$</td><td>$h_1 + h_2+$ $h_3 + h_4$</td><td>h_0+ $h_3 + h_4$</td><td>$h_0 + h_2+$ $h_3 + h_4$</td><td>$h_0 + h_1+$ $h_3 + h_4$</td><td>$h_0 + h_1 + h_2+$ $h_3 + h_4$</td></tr>
</table>

$\vec{X}_H = (x_3, x_4)$ labels the rows 00, 01, 10, 11.

By setting $k = q + 1$ in Corollary 4.2, we have the ratio $(n - q)2^{q-n+1}$. This shows that the larger the value $n - q$, the larger the reduction ratio by using an LUT cascade.

5 Arithmetic Decomposition of WS Functions

Consider filter realizations where q is the number of quantization bits. Experimental results show that q output WS functions require $(q + 1)$-input q-output cells. Thus, when q is large, large embedded memories in an FPGA are required. To implement WS functions with many outputs on a small FPGA, we can decompose WS functions into smaller ones.

A $2q$-output WS function can be decomposed into a pair of WS functions as follows: Let, h_i be a coefficient of $2q$ output WS function. Then, h_i can be written as

$$h_i = 2^q h_{Ai} + h_{Bi},$$

where h_{Ai} denotes the most significant q bits, and h_{Bi} denotes the least significant q bits. In this case, we can implement the $2q$ output WS function by using a pair of WS functions and an adder, as shown in Fig. 5.1. Note that the adder has $2q$ inputs and q outputs.

Theorem 5.1 *A $2q$-output WS function $F(\vec{X})$ that represents*

$$\sum_{i=0}^{n-1} h_i x_i$$

can be decomposed into a pair of WS functions $\vec{F}_A(\vec{X})$ and $\vec{F}_B(\vec{X})$, where $\vec{F}_A(\vec{X})$ is a q-output WS function representing

$$\sum_{i=0}^{n-1} h_{Ai} x_i,$$

and $\vec{F}_B(\vec{X})$ is a $q + \lceil \log_2 n \rceil$-output WS function representing

$$\sum_{i=0}^{n-1} h_{Bi} x_i,$$
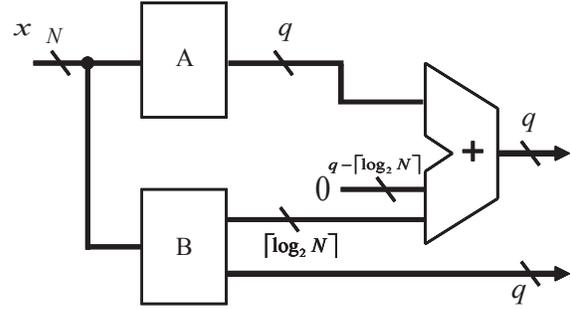
Figure 5.1. Arithmetic Decomposition of $2q$ output WS Function.
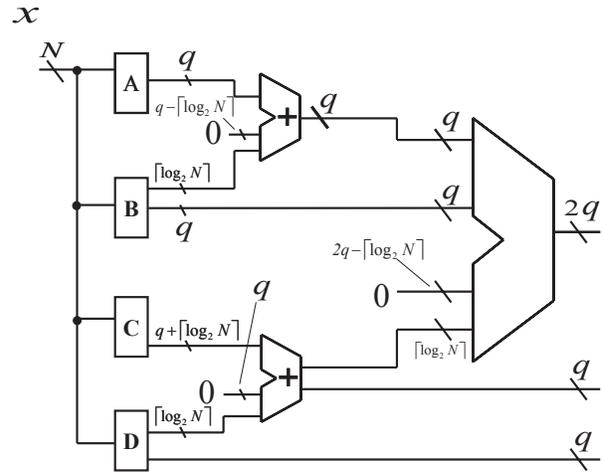
Figure 5.2. Arithmetic Decomposition of $4q$-output WS Function.

and

$$h_i = 2^q h_{Ai} + h_{Bi}.$$

This is an **arithmetic decomposition of a WS function**. In a similar way, a $4q$ output WS function can be de-

5

compose into four WS functions as follows: Let, h_i be a coefficient of the $4q$ output WS function. Then, h_i can be written as

$$h_i = 2^{3q} h_{Ai} + 2^{2q} h_{Bi} + 2^q h_{Ci} + h_{Di},$$

where h_{Ai}, h_{Bi}, h_{Ci}, and h_{Di} denote q-bit numbers. As shown in Fig. 5.2, we realize the $4q$-output WS function by using four q-output WS functions and adders. Note that block A realizes a q-output WS function, while blocks B, C, and D realize $(q + \lceil \log_2 N \rceil)$-output WS functions.

Note that the output adder has $4q$ inputs and $2q$ outputs.

6 Experimental Results

6.1 Method of Experiment

To confirm the theoretical results in the previous chapters, we designed many WS functions for FIR filters by using LUT cascades. To design LUT cascades, we used binary decision diagrams (BDDs) instead of decomposition charts [10, 12]. In this case, the width of the BDD corresponds to the column multiplicity of the decomposition chart. The parameters of the filters are as follows:

- Type of the filters: (HPF, LPF, BPF, BEF)
- The number of taps N: (9,17,33)
- The number of quantization bits q: (8,10,12,14,16)
- Type of the window functions: (Kaizer window function, Hanning window function)

In total, we designed more than 100 different WS functions. We rounded the filter coefficients by ignoring the lower bits when the $(q + 1)$-th bit is 0, and by adding 1 to the q-th bit when the $(q + 1)$-th bit is 1.

6.2 Mapping to FPGAs

We used Altera Cyclone II FPGAs [2] which contain M4K embedded memory blocks in addition to LUT-type logic elements (LEs). The M4K has 4096 bits and 521 parity bits, and can be configured as memories with different numbers of inputs and different numbers of outputs. We implemented LUT cascades by using M4Ks. The environment of FPGA mapping is shown in Table 6.1.

6.3 Analysis of Results

Table 6.2 compares ROM and LUT cascade realizations of WS functions. Due to the page limitation, only the cases for ($N = 33$, and Kaizer window function) are shown in the table. Also, we considered symmetric filters, so $n = (N + 1)/2$.

Table 6.1. Environment and conditions for experiments.

FPGA device: Cyclone II	
Device type:	EP2C70F896C7[2]
Number of LEs:	68416
I/O pins:	622 (out of 896 total pins)
Memory bits:	1152000
Number of M4Ks:	250
DSP blocks (9-bit):	300
Tool for Logic Synthesis and Fitting	
Altera, Quartus II V4.1 [2]	
Optimization Parameter for the Tool	
Timing-driven compilation:	
Optimize timing,	
Analysis Synthesis Settings:	
Auto Fit	
(reduce Fitter effort after meeting timing requirements)	

When the number of the quantization bits is less than 16, many of the filter coefficients $h_0, h_1, \ldots, h_{n_1}$ are rounded to zero. Thus, the WS function depends on only the part of the input variables, and the filters do not work properly. So, we have to increase the number of quantization bits. Since a q-bit output WS function requires $(q + 1)$-input q-output cells, the implementation requires a large embedded memory in an FPGA. To reduce the size of the memory cells, we used the arithmetic decomposition shown in Fig. 5.2, where $q = 4$. Since each WS function has at most $q + \lceil \log_2 N \rceil$ outputs, where $q = 4$ and $N = 17$, the WS function can be realized with cells with at most $q + \lceil \log_2 N \rceil + 1 = 10$ inputs, as proven in Theorem 4.1. Table 6.2 shows realizations of filters in the form of Fig. 5.2, where the number of the quantization bits is 16. The table shows the size of each cascade, maximum width of BDDs, memory bits for each cascade, total memory bits, number of M4Ks, number of LEs, and operating frequency. In all cases, cascade realizations reduced the sizes of memory. A single memory realization requires $2^{17} \times 16 = 2^{21} = 2M$ bits, while LUT cascade realizations require $29 \sim 40$ k bits. These realizations used 4 to 5 % of total M4K of the FPGA, and less than 1 % of LEs. Note that the target FPGA(Cyclone II) contains 250 M4Ks. Thus, the total amount of memory is 1 Mega bits. So, the simple memory realization does not fit into the target FPGA.

The operating frequency is 101 to 109 MHz.

In Fig. 2.4, the variable x_0 that corresponds to h_0 is placed to the root-side of the BDD, and the variable $x_{(N-1)/2}$ that corresponds to $h_{(N-1)/2}$ is placed to the leaf-side of the BDD. This initial ordering of the BDD produced smaller BDDs, and produced smaller LUT cascades. The WS functions with larger q and larger N tend to have larger

Table 6.2. ROM and LUT Cascade Realization of WS Functions.

Filter Type	HPF				LPF				BPF				BEF			
(Cut Off Frequency)	(5kHz)				(5kHz)				(5k-10kHz)				(5k-10kHz)			
Window Function	Kaiser				Kaiser				Kaiser				Kaiser			
Tap [n]				33				33				33				33
Quantization Bit [bits]				16				16				16				16
ROM #IN				17				17				17				17
ROM #OUT				16				16				16				16
#Cascades				4				4				4				4
Cascade	A	B	C	D	A	B	C	D	A	B	C	D	A	B	C	D
#Cells	1	2	2	3	1	2	3	3	1	2	3	2	1	2	3	3
Cell0 #IN	10	10	10	10	7	8	9	10	7	10	9	10	9	10	10	10
Cell0 #OUT	4	6	7	7	4	5	5	6	4	7	6	6	4	7	7	7
Cell1 #IN		8	9	10		9	10	10		6	10	10		7	10	9
Cell1 #OUT		6	7	6		8	7	7		5	6	6		6	7	6
Cell2 #IN				7			8	9			5				6	6
Cell2 #OUT				5			6	7			4				5	5
Max Width BDD	11	56	66	90	8	130	91	111	8	88	63	58	10	50	71	78
Cascade Memory Bits [bits]	4096	7680	14336	13952	512	5376	11264	16896	512	7488	9344	12288	2048	7936	14656	10560
Total Memory Bits [bits]				40064				34048				29632				35200
Number of M4K's (Out of 250)				13 — 5 %				12 — 4 %				12 — 4 %				13 — 5 %
Number of LEs (Out of 68,416)				586 — < 1 %				585 — < 1 %				564 — < 1 %				568 — < 1 %
Operating frequency [MHz]				105				104				109				101

column multiplicities.

7 Conclusion

In this paper, we defined the WS function that represents the combinational part of the distributed arithmetic in an FIR filter. Also, we showed methods to realize a WS function by LUT cascades and adders. Major results are

1. WS functions have arithmetic decomposition, and can be implemented in the form of Figs. 5.1 or 5.2.

2. LUT cascade realizations require much smaller memory than single ROM realizations.

3. Arithmetic decompositions are effective in implementing FIR filters.

When the number of quantization bits is large, we have to partition the outputs into several groups by arithmetic decomposition.

Also, when the number of taps are large, we have to partition the inputs into groups. For each group, we can implement a WS function, and finally, we can obtain the sum by using an adder [1]. This greatly reduces the necessary amount of memory. Note that LUT cascades can be used for these WS functions and the adder.

In this paper, we defined the WS function as a model of the combinational part of the distributed arithmetic for digital filters. Note that WS functions can be used to implement Discrete Cosine Transform (DCT), Discrete Fourier Transform (DFT) and other convolution operations.

References

[1] Actel Corporation, "CoreFIR: Finite impulses response (FIR) filter generator,"
http://www.actel.com/ipdocs/CoreFIR_IP_DS. BPF

[2] Altera: http://www.altera.com/

[3] R. J. Andraka, "FIR filter fits in an FPGA using a bit serial approach," *Proceedings of the EE-Times 3rd Annual PLD design Conference and Exhibit*, March 1993.

[4] R. L. Ashenhurst, "The decomposition of switching functions," *In Proceedings of an International Symposium on the Theory of Switching*, pp. 74-116, April 1957.

[5] H. A. Curtis, *A New Approach to The Design of Switching Circuits*, D. Van Nostrand Co., Princeton, NJ, 1962.

[6] T-T. Do, H. Kropp, C. Reuter, and P. Pirsch, "A flexible implementation of high-performance FIR filters on Xilinx FPGAs," *FPL 1998*, Estonia, Aug. 31 - Sept. 3, 1998, pp. 441-445.

[7] L. Mintzer, "FIR filters with field-programmable gate arrays," *Journal of VLSI Signal Processing*, pp. 120-127, Aug. 1993.

[8] K. Nakamura, T. Sasao, M. Matsuura, K. Tanaka, K. Yoshizumi, H. Qin, and Y. Iguchi, "Programmable logic device with an 8-stage cascade of 64K-bit asynchronous SRAMs," *Cool Chips VIII, IEEE Symposium on Low-Power and High-Speed Chips*, April 20-22, 2005, Yokohama, Japan.

[9] K. K. Parhi, *VLSI Digital Signal Processing Systems Design and Implementation*, John Wiley, New York, 1999.

[10] T. Sasao, M. Matsuura, and Y. Iguchi, "A cascade realization of multiple-output function for reconfigurable hardware," *International Workshop on Logic and Synthesis (IWLS01)*, Lake Tahoe, CA, June 12-15, 2001, pp. 225-230.

[11] T. Sasao, "Analysis and synthesis of weighted-sum functions," *International Workshop on Logic and Synthesis*, Lake Arrowhead, CA, USA, June 8-10, 2005 (to be published).

[12] T. Sasao and M. Matsuura, "A method to decompose multiple-output logic functions," *41st Design Automation Conference*, San Diego, CA, USA, June 2-6, 2004, pp. 428-433.

[13] S. Yu and E. E. Swartzlander, "DCT implementation with distributed arithmetic," *IEEE Trans. on Computers*, Vol. 50, No. 9, Sept. 2001, pp. 985-991.

[14] S. A. White, " Applications of distributed arithmetic to digital signal processing: a tutorial review," *IEEE ASSP Magazine*, Vol. 6, No. 3, July 1989, pp. 4-19.