IPSJ Transactions on System LSI Design Methodology Vol.5 96–105 (Aug. 2012)

[DOI: 10.2197/ipsjtsldm.5.96]

Regular Paper

A Fast Weighted Adder by Reducing Partial Product for Reconstruction in Super-Resolution

Hiromine Yoshihara[1,a)] Masao Yanagisawa[2] Nozomu Togawa[1]

Received: November 14, 2011, Revised: March 2, 2012,
Accepted: April 19, 2012, Released: August 6, 2012

Abstract: In recent years, it is quite necessary to convert conventional low-resolution images to high-resolution ones at low cost. Super-resolution is a technique to remove the noise of observed images and restore its high frequencies. We focus on reconstruction-based super-resolution. Reconstruction requires large computation cost since it requires many images. In this paper, we propose a fast weighted adder for reconstruction-based super-resolution. From the viewpoint of reducing partial products, we propose two approaches to speed up a weighted adder. First, we use selector logics to halve its partial products. Second, we propose a weights-range limit method utilizing negative term. By applying our proposed approaches to a weighted adder, we can reduce carry propagations and our weighted adder can be designed by a fast circuit as compared to conventional ones. Experimental evaluations demonstrate that our weighted adder reduces its delay time by a maximum of 25.29% and its area to a maximum of 1/3, compared to conventional implementations.

Keywords: selector-logics, weighted adder, super-resolution, reconstruction

1. Introduction

High-resolution output devices such as television sets and computers with large screens are very widely used in recent years. The resolution difference between conventional low-resolution output devices and high-resolution ones has become larger and larger. It is quite necessary to convert conventional low-resolution images to high-resolution ones at low cost. To solve this problem, conventional interpolation methods such as a bilinear interpolation and a cubic convolution [4] are used. These methods can interpolate the pixels of observed images. They cannot restore the high frequencise of them and interpolated images result in indistinct ones.

Super-resolution [2], [5], [8], [9], [12] is a technique to remove the noise of observed images and restore the high frequencise of ones. We focus on reconstruction-based super-resolution which is able to restore their own brightnesses. Reconstruction-based super-resolution can be divided into two approaches; one is a spatial domain approach and the other is a frequency domain approach [11]. The spatial domain approach can accommodate global, non-global motion, optical blur, motion blur, compression artifacts and more. The frequency domain approach provides the advantages of theoretical simplicity, low computational complexity, which is appropriate to hardware design. We focus on frequency domain super-resolution methods.

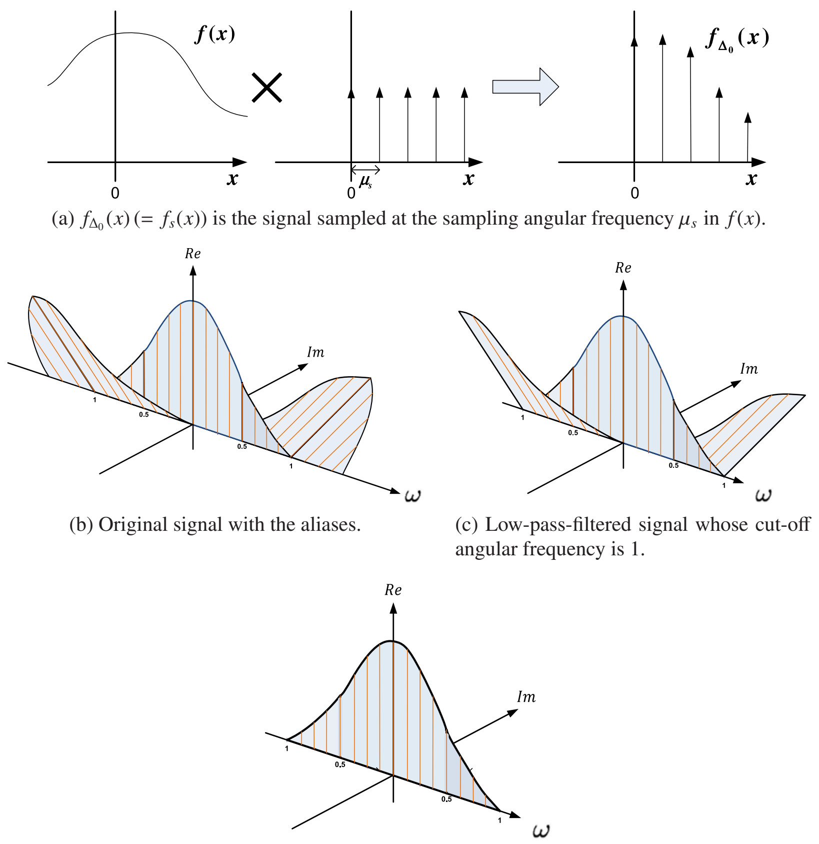
Reconstruction is one of the processes in reconstruction-based super-resolution based on a frequency-domain approach. It requires large computation cost since we need many images in reconstruction. It is strongly necessary to improve arithmetic circuits' performance specific to reconstruction.

Veterli et al. [14] proposes an algorithm that can estimate rotation, horizontal and vertical shifts between the reference image and each of the other observed images for reconstruction-based super-resolution in frequency domain. Tanaka and Okutomi [13] proposes a fast registration algorithm for reconstruction-based super-resolution. As far as we know, there are no previous approaches which focus on speeding-up reconstruction.

In this paper, we propose a weighted adder for reconstruction-based super-resolution. From the viewpoint of reducing partial products, we propose two approaches to speed up a weighted adder. First, we use selector logics to halve its partial products. Second, we propose a weights-range limit method utilizing negative term. Applying our proposed approaches to a weighted adder, we can reduce carry propagations and our weighted adder can be designed by a fast circuit as compared to conventional ones. Experimental results show that our proposed weighted adder improves its performance by a maximum of 33.85% and reduces its area to a maximum of 1/3, compared to conventional ones.

This paper is organized as follows: Section 2 introduces the reconstruction-based super-resolution by Aoki method; Section 3 proposes two approaches: one is a selector-logic-based approach and the other is a weights-range limit method, to speed up a weighted adder; Section 4 demonstrates experimental results; Section 5 gives concluding remarks.

1 Department of Computer Science and Engineering, Waseda University, Shinjuku, Tokyo 169–8555, Japan
2 Department of Electronic and Photonic Systems, Waseda University, Shinjuku, Tokyo 169–8555, Japan
a) hiromine.yoshihara@togawa.cs.waseda.ac.jp



(a) $f_{\Delta_0}(x)\,(= f_s(x))$ is the signal sampled at the sampling angular frequency μ_s in $f(x)$.

(b) Original signal with the aliases.

(c) Low-pass-filtered signal whose cut-off angular frequency is 1.

(d) Frequecy-domain signal after weighted-adding and removing the aliases.

Fig. 2 Restoring the original signal by removing the aliases.

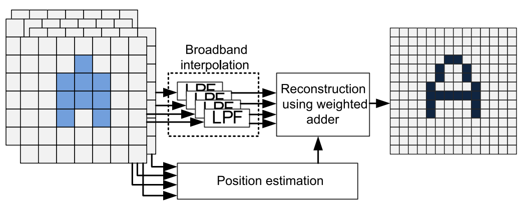


Fig. 1 A block diagram of super-resolution process by Aoki method [2].

2. Reconstruction-based Super-resolution by Aoki Method

In this section, we introduce the reconstruction-based super-resolution by Aoki method [2], [5]. Aoki method is composed of the three steps as follows:

1. Position estimation
2. Broadband interpolation
3. Reconstruction using weighted adder

A block diagram of Aoki method is shown in **Fig. 1**. In the position estimation step, we compute a pixel shift or a subpixel shift[*1] between the referenced image and each observed image and register them. We can use a traditional block matching method or the spatio-temporal image derivative method[*2] [1] by obtaining a pixel shift or a subpixel shift. In the broadband interpolation step, we apply broadband low-pass filters[*3] to the registered images, which results in many aliases but they include high frequencise completely. In the reconstruction using weighted adder step, we perform weighted-sum of input signals to remove the aliases and restore the original signal theoretically. The reconstruction using weighted adder step is composed of weight calculation, weighted sum, and sinc-function-based interpolation.

Let us explain how to remove the aliases and restore the original signal theoretically using the exmaple as depicted in **Fig. 2**. Let $f(x)$ be an original signal and $f_{\Delta_0}(x)$ be a sampled signal whose sampling angular frequency μ_s is 1 ($\mu_s = 1$) as in Fig. 2 (a). However, assume that $f(x)$ has a bandwidth of μ_s and its Fourier transform $F_s(u)$ is depicted as in Fig. 2 (b) where only the center part of $F_s(u)$ is shown. According to the sampling theorem, $f(x)$ cannot be restored by using $f_{\Delta_0}(x)$ only, since its Nygust frequency is $\mu_s/2$ but $f(x)$ contains frequencies higher than $\mu_s/2$. If $F_s(u)$ is low-pass-filtered whose cut-off frequency is μ_s ($\mu_s = 1$), it has many aliases as shown in Fig. 2 (c). As described just below in **Reconstruction Using Weighted Adders**, if weights meet Eq. (4) and perform weighted sum of signals as in Eq. (6), we

*1 A pixel shift can be represented by a motion vector between the referenced image and each observed image with pixel-wise accuracy. In the same way, a subpixel shift can be represented by a motion vector between the referenced image and each observed image with "fractional"-pel accuracy.

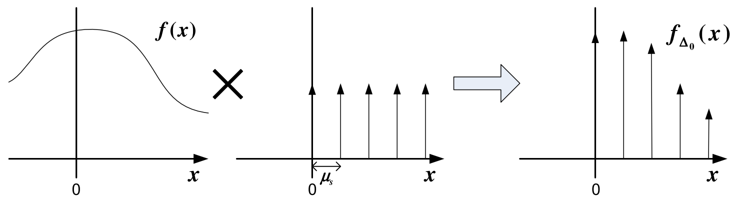
*2 The spatio-temporal image derivative method is one of the methods to obtain a velocity or a (sub-)pixel shift of moving images using spatio-temporal derivative.

*3 A broadband low-pass filter here means a low-pass filter whose cut-off frequency is more than the Nyquist frequency.

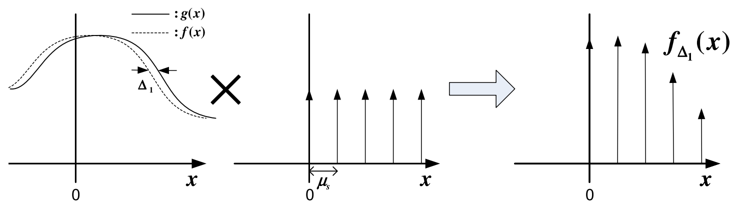
IPSJ Transactions on System LSI Design Methodology Vol.5 96–105 (Aug. 2012)

Table 1 Experimental results of the CPU times in Aoki method.

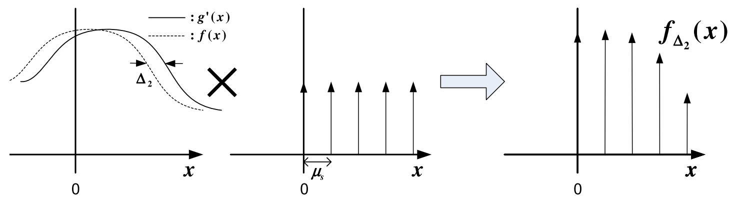
Images	CPU time [s]				
	Position estimation	Broadband interpolation	Reconstruction using weighted adder		
			Weight calculation	Weighted sum	Bicubic interpolation
Lena	1.68	0.34	1.62	1.40	2.34
Mandrill	1.35	0.31	1.56	1.55	2.56



(a) $f_{\Delta_0}(x)$ $(= f_s(x))$ is the signal sampled at the sampling angular frequency μ_s in $f(x)$.



(b) $f_{\Delta_1}(x)$ is the signal sampled at the sampling angular frequency μ_s in Δ_1-shifted $f(x)$.



(c) $f_{\Delta_2}(x)$ is the signal sampled at the sampling angular frequency μ_s in Δ_2-shifted $f(x)$.

Fig. 3 Sampling with Δ_0-, Δ_1- and Δ_2-shifted signals.

can remove these aliases and extract the original high frequencise without aliases as shown in Eq. (5) and as in Fig. 2 (d). By applying inverse Fourier transform to the frequency-domain signal as shown in Fig. 2 (d), we can have a complete original signal $f(x)$ theoretically*4.

Aoki method has several advantages as follows; First, Aoki method can accommodate horizontal and vertical blur. Second, it does not need very complex and iterative operations which conventional reconstruction-based super-resolutions need by using weighted adders in the reconstruction step. Finally, it can remove the alias of observed images and restore their high frequencise theoretically.

Now we demonstrate experimental evaluations on Aoki method. In this experiment, we implement super-resolution processes using Octave [10] on Intel Core i7 CPU at 2.13 clock GHz whose memory size is 8.00 GB. We use two types of images whose size is 128×128. In each image type, we use seven images to reconstruct a super-resolution image. In the position estimation step, we use traditional block matching where we use a block size of 1×3 and a search window of 6×6-pixel size and in the reconstruction using weighted adder step, we use bicubic interpolation [3]. As shown in **Table 1**, "Weighted sum" is one of the most time-consuming processes. We focus on weighted sum and speed-up it as a first step to speed-up the super-resolution processes.

*4 Note that we cannot restore a complete original signal practically, since we have numerical errors and interpolation errors.

Reconstruction Using Weighted Adders:

Consider implementing a weighted adder to cancel out the alias of each low-pass-filtered observed image. In the rest of this paper, we consider 1-dimensional signals for simplicity, but the discussion here can be applied to 2-dimensional images very easily.

Let $f(x)$ be an original signal. $f_s(x)$ is the signal sampled at the sampling angular frequency μ_s in $f(x)$. $f_{\Delta_n}(x)$ is the signal sampled at the sampling angular frequency μ_s in $f(x)$ which is shifted by Δ_n $(n = 1, 2, \cdots)$. We further define $\Delta_0 = 0$ and $f_{\Delta_0}(x) = f_s(x)$.

Note that, given a set of $(n + 1)$ signals $(n \geq 2)$, Δ_n are determined based on these $(n + 1)$ signals. Thus, if we are given another set of $(n + 1)$ signals, we have to re-calculate Δ_n and thus their associated weights.

For example, assume that we have an original signal $f(x)$ and its sampled signal $f_{\Delta_0}(x)$ where we have no x-axis shifts $(\Delta_0 = 0)$ as in **Fig. 3** (a). Assume also that we have signals $g(x)$ and $g'(x)$ as in Figs. 3 (b) and 3 (c). Given a set of three signals $f(x)$, $g(x)$, and $g'(x)$, we have Δ_1 and Δ_2 where $g(x) = f(x - \Delta_1)$ and $g'(x) = f(x - \Delta_2)$. Note that Δ_n is an x-axis shift in a 1-

dimensional signal and this corresponds to a (sub-)pixel shift in a 2-dimensional image. The estimation of Δ_n also corresponds to the position estimation in a 2-dimensional image. After that, we can calculate the weights w_0, w_1, and w_2 based on the values of Δ_1 and Δ_2 so that they can satisfy Eq. (4).

Let $F(u)$ be the Fourier transform of $f(x)$. $F_s(u)$ and $F_\Delta(u)$ are the Fourier transforms of $f_s(x)$ and $f_{\Delta_n}(x)$ $(n = 0, 1, 2)$, respectively, and they can be expressed by:

$$F_s(u) = \sum_{k=-\infty}^{\infty} F(u - k\mu_s) \tag{1}$$

$$F_{\Delta_n}(u) = \sum_{k=-\infty}^{\infty} e^{-j2\pi k\Delta_n} F(u - k\mu_s). \tag{2}$$

where $F_s(u) = F_{\Delta_0}(u)$ since $\Delta_0 = 0$.

Assume the original signal $f(x)$ has a bandwidth of twice Nyquist frequency. If low-pass filters whose cut-off frequency is the twice Nyquist frequency are applied to $f_s(x) = f_{\Delta_0}(x)$, $f_{\Delta_1}(x)$ and $f_{\Delta_2}(x)$, the sum of these signals includes alias but all high frequencise of the original $f(x)$.

Let us consider the terms of $k = 0$ and $k = \pm 1$ in the above Eqs. (1) and (2). By introducing the weights w_n $(n = 0, 1, 2)$, we can compute the weighted-sum operation in frequency domain as follows:

$$\sum_{n=0,1,2} w_n F_{\Delta_n}(u) = \sum_{n=0,1,2} w_n F(u) + \sum_{n=0,1,2} w_n e^{j2\pi\Delta_n} F(u + \mu_s) + \sum_{n=0,1,2} w_n e^{-j2\pi\Delta_n} F(u - \mu_s). \tag{3}$$

If the weights w_n meet the conditions as follows:

$$\sum_{n=0,1,2} w_n = 1 \quad \text{and} \quad \sum_{n=0,1,2} w_n e^{j2\pi\Delta_n} = 0, \tag{4}$$

we can remove all the alias and restore the high frequencies of the original signal, i.e.,

$$\text{Eq. (3)} = F(u). \tag{5}$$

In practice, we compute weighted-sum of them by using real signals. Thus the reconstruction is finally expressed as follows under the condition Eq. (4):

$$\sum_{n=0,1,2} w_n f_{\Delta_n}(x). \tag{6}$$

Let us explain how to interpolate signals. We interpolate signals by using the sinc function. We assume the original signal $f(x)$ has a bandwidth of μ_s. The time-domain transfer function $h(x)$ corresponding to a window function whose cut-off frequency is μ_s is expressed by:

$$h(x) = \frac{\sin(\mu_s x)}{\mu_s x}. \tag{7}$$

By using Eq. (7), we can interpolate signals as follows:

$$f(x) = \sum_{n=0,1,2} w_n f_{\Delta_n}(x) \otimes h(x). \tag{8}$$

where $\sum_{n=0,1,2} w_n f_{\Delta_n}$ shows the weighted sum of sampled signals and $\otimes$ shows convolution. In practice, we cannot execute interpolation using the sinc fuction above directly, since it requires near-infinite addition. Then we use bicubic interpolation [3] which approximates sinc-function-based interpolation.

3. Efficient Weighted Adders for Reconstruction in Super-Resolution

Bit-level transformation is one of the methods to optimize arithmetic units [7]. In this section, we propose two bit-level transformation techniques such that we can reduce partial products generated for reconstruction using weighted adders. First, we propose a method that halves partial products by utilizing selector logics. Second, we propose a weights-range limit method that will reduce the partial products furthermore by utilizing a negative term.

3.1 Bit-level Representation for Weighted Adders

Eq. (4) can be generalized for n inputs as follows:

$$w_0 f_{\Delta_0}(x) + w_1 f_{\Delta_1}(x) + \cdots + w_{n-1} f_{\Delta_{n-1}}(x) \tag{9}$$

$$w_0 + w_1 + \cdots + w_{n-1} = 1. \tag{10}$$

We denote $f_{\Delta_n}(x)$ as f_n for simplicity.

An input signal f_j is an m-bit signed fixed-point variable and a weight w_j is an m-bit unsigned fixed-point variable[*5]. They are

[*5] We define f_j to be an m-bit signed fixed-point signal due to the following two reasons:

More generalized weighted adder:
By considering singed valuables, we can construct "more general" weighted adders which can be applied to many applications other than image processing.
The weight w_j must be an unsigned variable because it is defined by $0 \le w_j \le 1$.

We employ an m-bit signed fixed-point variable so that we can apply selector logics to weighted addition:
Even when we consider signed valuables, they can be easily applied to image processing as follows:
In order to obtain an m-bit signed signal f_j from an m-bit unsigned signal g_j, one of the easiest ways is just subtracting 2^{m-1}, i.e., $f_j = g_j - 2^{m-1}$. This can be done by just inverting the MSB of g_j. In our cell library, 1-bit inverter just requires 0.076 ns, which can be negligible.
A weight addition can be expressed by:

$$\sum_{j=0}^{n-1} w_j f_j = \sum_{j=0}^{n-1} w_j \left(g_j - 2^{m-1}\right) = \sum_{j=0}^{n-1} w_j g_j - \sum_{j=0}^{n-1} w_j 2^{m-1}. \tag{11}$$

Using the definition of the weight as in Eq. (10), Eq. (11) can be transformed into:

$$\text{Eq. (11)} = \sum_{j=0}^{n-1} w_j g_j - 2^{m-1}. \tag{12}$$

Finally, we have:

$$\sum_{j=0}^{n-1} w_j g_j = \sum_{j=0}^{n-1} w_j f_j + 2^{m-1}. \tag{13}$$

Thus we can have a weighted sum of unsigned variables by using singed variables and inverting its MSB.
When we use singed valuables, we can effectively use selector logics in weighted addition as discussed in Section 3.2. Then, we can speed-up the weighted addition by reducing partial products. As in our experiments (Section 4), our selector-logic-based results are superior in terms of circuit speed compared to other implementations, even when we consider the delays to invert MSBs to convert unsigned signals into singed signals and vice versa.

represented by:

$$f_j = -f_{j,(m-1)}2^{m-1} + \sum_{i=0}^{m-2} f_{j,i}2^i \qquad (0 \le j \le n-1) \tag{14}$$

$$w_j = \sum_{i=0}^{m-1} w_{j,i}2^{i-m} \qquad (0 \le j \le n-1), \tag{15}$$

where $f_{j,i}$ represents the i-th bit of f_j and $w_{j,i}$ represents the i-th bit of w_j. Since we use here a 2's complementary form, $\left(-f_j\right)$ and $\left(-w_j\right)$ are expressed by:

$$-f_j = -\overline{f_{j,(m-1)}}2^{m-1} + \sum_{i=0}^{m-2} \overline{f_{j,i}}2^i + 1 \tag{16}$$

$$-w_j = -1 \cdot 2^0 + \sum_{i=0}^{m-2} \overline{w_{j,i}}2^{i-m} + 1 \cdot 2^{-m}. \tag{17}$$

By using Eqs. (14)–(17), we will perform a bit-level transformation to Eq. (9) such that a selector logic can be applied to them.

Let us explain a bit-level transformation for a weighted adder with three input signals ($n = 3$). Using Eqs. (10) and (15), Eq. (9) with $n = 3$ can be transformed into:

$$\begin{aligned}
&(\text{Eq. }(9))\Big|_{n=3}\\
&= w_0 f_0 + w_1 f_1 + w_2 f_2\\
&= (1 - w_1 - w_2) f_0 + w_1 f_1 + w_2 f_2\\
&= f_0 + (f_1 - f_0)\sum_{i=0}^{m-1} w_{1,i}2^{i-m} + (f_2 - f_0)\sum_{i=0}^{m-1} w_{2,i}2^{i-m}\\
&= f_0 + f_1 \sum_{i=0}^{m-1} w_{1,i}2^{i-m} + f_0\left(-\sum_{i=0}^{m-1} w_{1,i}2^{i-m}\right)\\
&\quad + f_2 \sum_{i=0}^{m-1} w_{2,i}2^{i-m} + f_0\left(-\sum_{i=0}^{m-1} w_{2,i}2^{i-m}\right).
\end{aligned} \tag{18}$$

Using Eq. (17), we have:

$$\begin{aligned}
-\sum_{i=0}^{m-1} w_{1,i}2^{i-m} &= -1 + \sum_{i=0}^{m-1} \overline{w_{1,i}}2^{i-m} + 1 \cdot 2^{-m}\\
-\sum_{i=0}^{m-1} w_{2,i}2^{i-m} &= -1 + \sum_{i=0}^{m-1} \overline{w_{2,i}}2^{i-m} + 1 \cdot 2^{-m}.
\end{aligned} \tag{19}$$

Using Eqs. (14) and (19), Eq. (18) can be transformed into:

$$\begin{aligned}
&\text{Eq. }(18)\\
&= \left(-1 + 2^{-m+1}\right) f_0\\
&\quad + \left\{ f_0 \sum_{i=0}^{m-1} \overline{w_{1,i}}2^{i-m} + f_1 \sum_{i=0}^{m-1} w_{1,i}2^{i-m} \right\}\\
&\quad + \left\{ f_0 \sum_{i=0}^{m-1} \overline{w_{2,i}}2^{i-m} + f_2 \sum_{i=0}^{m-1} w_{2,i}2^{i-m} \right\}
\end{aligned} \tag{20}$$

$$\begin{aligned}
&= \underset{\sim}{-f_0} + f_0 2^{-m+1}\\
&\quad + \underline{\underline{\sum_{i=0}^{m-1}\{ f_{0,(m-1)}(-\overline{w_{1,i}}) + f_{1,(m-1)}(-w_{1,i}) + f_{0,(m-1)}(-\overline{w_{2,i}})}}\\
&\quad \underline{\underline{+ f_{2,(m-1)}(-w_{2,i})\}2^{i-1}}}\\
&\quad + \sum_{i=0}^{m-2}\sum_{j=0}^{m-1} \left(f_{0,i}\overline{w_{1,j}} + f_{1,i}w_{1,j} + f_{0,i}\overline{w_{2,j}} + f_{2,i}w_{2,j} \right) 2^{i+j-m}.
\end{aligned} \tag{21}$$

First, we propose a method that halves partial products by utilizing selector logics (Section 3.2). Second, we will focus on the negative term with the wavy line in Eq. (21) and propose a weights-range limit method that will reduce the partial products furthermore (Section 3.3).

For example, a weighted adder for reconstruction with $n = 3$ and $m = 4$ originally generates 88 partial products, which corresponds to Eq. (24) and **Fig. 4** (a). By using our proposed approaches in Section 3.2 and Section 3.3, they will be reduced to 33 as in Fig. 4 (c). Overall, we can realize extremely fast weighted adders.

3.2 Reducing Partial Products by Using Selector Logics

In this subsection, we propose a bit-level transformation technique such that a selector logic can be applied to a weighted adder.

First, the double-underlined term in Eq. (21) can be transformed into:

$$\begin{aligned}
\sum_{i=0}^{m-1} f_{0,(m-1)}\left(-\overline{w_{1,i}}\right)2^{i-1} &= f_{0,(m-1)} \sum_{i=0}^{m-1} \left(1 - \overline{w_{1,i}} - 1\right) 2^{i-1}\\
&= f_{0,(m-1)} \left(-1 \cdot 2^{m-1} + \sum_{i=0}^{m-1} w_{1,i}2^{i-1} + 2^{-1} \right)\\
\sum_{i=0}^{m-1} f_{1,(m-1)}\left(-w_{1,i}\right)2^{i-1} &= f_{1,(m-1)} \sum_{i=0}^{m-1} \left(1 - w_{1,i} - 1\right) 2^{i-1}\\
&= f_{1,(m-1)} \left(-1 \cdot 2^{m-1} + \sum_{i=0}^{m-1} \overline{w_{1,i}}2^{i-1} + 2^{-1} \right)\\
\sum_{i=0}^{m-1} f_{0,(m-1)}\left(-\overline{w_{2,i}}\right)2^{i-1} &= f_{0,(m-1)} \sum_{i=0}^{m-1} \left(1 - \overline{w_{2,i}} - 1\right) 2^{i-1}\\
&= f_{0,(m-1)} \left(-1 \cdot 2^{m-1} + \sum_{i=0}^{m-1} \overline{w_{2,i}}2^{i-1} + 2^{-1} \right)\\
\sum_{i=0}^{m-1} f_{2,(m-1)}\left(-w_{2,i}\right)2^{i-1} &= f_{2,(m-1)} \sum_{i=0}^{m-1} \left(1 - w_{2,i} - 1\right) 2^{i-1}\\
&= f_{2,(m-1)} \left(-1 \cdot 2^{m-1} + \sum_{i=0}^{m-1} \overline{w_{2,i}}2^{i-1} + 2^{-1} \right).
\end{aligned} \tag{22}$$

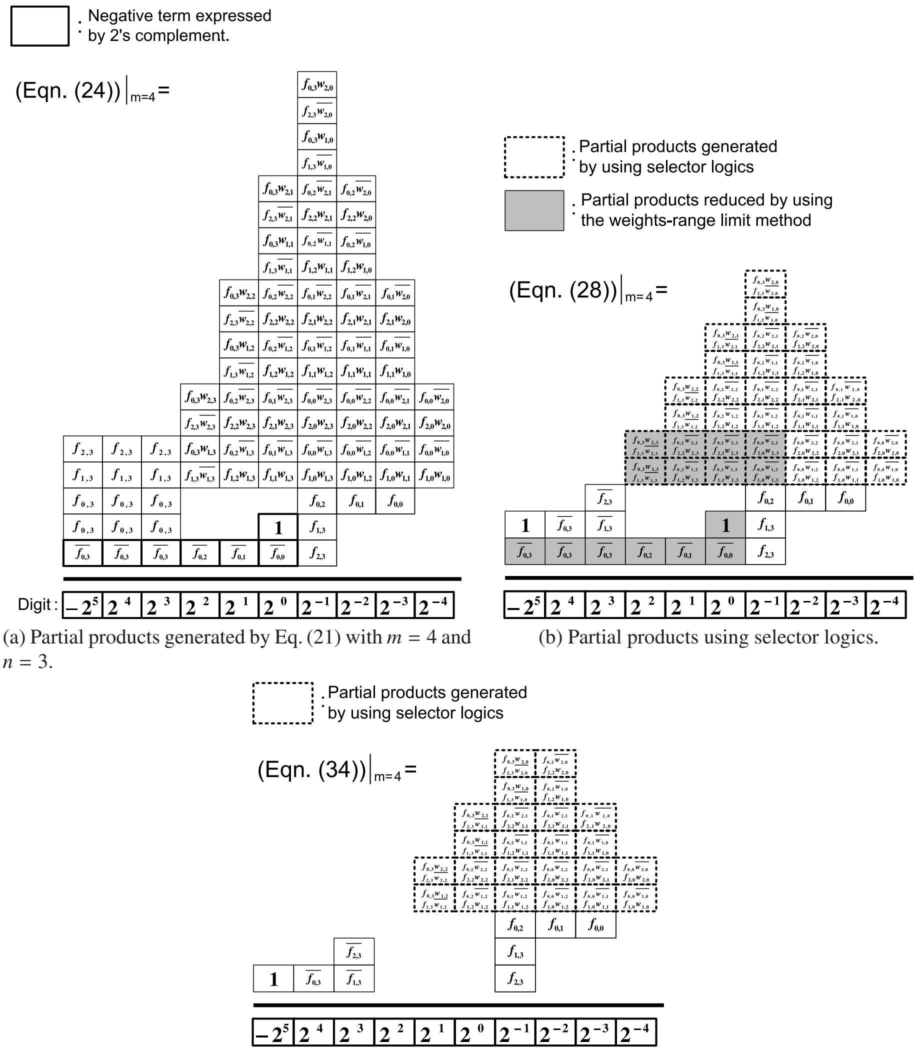
Using Eq. (14), $f_0 2^{-m+1}$ in Eq. (21) can be transformed into:

$$f_0 2^{-m+1} = f_0 \times 2^{-m+1} = -f_{0,(m-1)} + \sum_{i=0}^{m-2} f_{0,i}2^{i-m+1}. \tag{23}$$

Using Eqs. (22) and (23), Eq. (21) can be transformed into:

$$\text{Eq. }(21) = -f_0 - \left(2f_{0,(m-1)} + f_{1,(m-1)} + f_{2,(m-1)}\right)2^{m-1}$$

IPSJ Transactions on System LSI Design Methodology Vol.5 96–105 (Aug. 2012)



(a) Partial products generated by Eq. (21) with $m = 4$ and $n = 3$.

(b) Partial products using selector logics.

(c) Partial products using the weights-range limit method.

Fig. 4 Partial products generated for reconstruction using weighted adders with $m = 4$ and $n = 3$ using selector logics.

$$
\begin{aligned}
&+ \sum_{i=0}^{m-1}(f_{0,(m-1)}w_{1,i} + f_{1,(m-1)}\overline{w_{1,i}} + f_{0,(m-1)}w_{2,i} \\
&\qquad + f_{2,(m-1)}\overline{w_{2,i}})2^{i-1} \\
&+ (2f_{0,(m-1)} + f_{1,(m-1)} + f_{2,(m-1)})2^{-1} - f_{0,(m-1)} \\
&+ \sum_{i=0}^{m-2} f_{0,i}2^{i-m+1} \\
&+ \sum_{i=0}^{m-2}\sum_{j=0}^{m-1}(f_{0,i}\overline{w_{1,j}} + f_{1,i}w_{1,j} + f_{0,i}\overline{w_{2,j}} \\
&\qquad + f_{2,i}w_{2,j})2^{i+j-m} \\
= &-f_0 - \left(2f_{0,(m-1)} + f_{1,(m-1)} + f_{2,(m-1)}\right)2^{m-1} \\
&+ \sum_{i=0}^{m-1}(f_{0,(m-1)}w_{1,i} + f_{1,(m-1)}\overline{w_{1,i}} + f_{0,(m-1)}w_{2,i} \\
&\qquad + f_{2,(m-1)}\overline{w_{2,i}})2^{i-1} \\
&+ f_{0,(m-1)}2^{0} + (f_{1,(m-1)} + f_{2,(m-1)})2^{-1} - f_{0,(m-1)} \\
&+ \sum_{i=0}^{m-2} f_{0,i}2^{i-m+1} \\
&+ \sum_{i=0}^{m-2}\sum_{j=0}^{m-1}(f_{0,i}\overline{w_{1,j}} + f_{1,i}w_{1,j} + f_{0,i}\overline{w_{2,j}} \\
&\qquad + f_{2,i}w_{2,j})2^{i+j-m} \\
= &-f_0 - \left(2f_{0,(m-1)} + f_{1,(m-1)} + f_{2,(m-1)}\right)2^{m-1} \\
&+ \sum_{i=0}^{m-1}\Bigl(\underline{f_{0,(m-1)}w_{1,i} + f_{1,(m-1)}\overline{w_{1,i}}} \\
&\qquad + \underline{f_{0,(m-1)}w_{2,i} + f_{2,(m-1)}\overline{w_{2,i}}}\Bigr)2^{i-1} \\
&+ \left(f_{1,(m-1)} + f_{2,(m-1)}\right)2^{-1} + \sum_{i=0}^{m-2} f_{0,i}2^{i-m+1} \\
&+ \sum_{i=0}^{m-2}\sum_{j=0}^{m-1}\Bigl(\underline{f_{0,i}\overline{w_{1,j}} + f_{1,i}w_{1,j}} \\
&\qquad + \underline{f_{0,i}\overline{w_{2,j}} + f_{2,i}w_{2,j}}\Bigr)2^{i+j-m}.
\end{aligned} \tag{24}
$$

Selector Logics:

Let us focus on underlined terms in Eq. (24). They have a form which is completely the same as the selector logic represented by an expression below:

$$d = a\overline{c} + bc, \tag{25}$$

where a, b, c and d are 1-bit variables. The output value d is set to be a or b by using the select signal c. This expression can be implemented by two logical ANDs and a logical OR but it can be also implemented by a "selector."

The *output range* of the selector logic is expressed by

$$0 \leq |a\overline{c} + bc| \leq 1. \tag{26}$$

In other words, it generates no carry-out, since the output of the selector logic becomes 0 or 1. Generally speaking, any arithmetic operation which has two or three 1-bit inputs and whose output range is greater than one generates a carry-out. For example, a full adder and a half adder used very often in arithmetic units must generate a sum and a carry-out. This means that, if we can apply selector logics to the underlined terms in Eq. (24), carry propagation must be reduced. As shown in Fig. 4 (b), a selector logic directly computes each of the underlined terms in Eq. (24). We can reduce partial products by pre-computating them. Since selector logics generate no carry-outs, this precomputation can be done very fast.

How to Reduce Negative Terms:

Additionally, let us focus on the negative terms in Eq. (24) whose coefficient is 2^{m-1}. A weighted-sum of any m-bit input signals results in a $(2m + 1)$-bit signed fixed-point value. Each of the negative terms in Eq. (24) whose coefficient is 2^{m-1} affects not only 2^{m-1}-th digit but 2^m-th digit and 2^{m+1}-th digit. For example, we require 12 partial products for $(-(2f_{0,(m-1)} + f_{1,(m-1)} + f_{2,(m-1)})\,2^m)$. Its overhead may be too large.

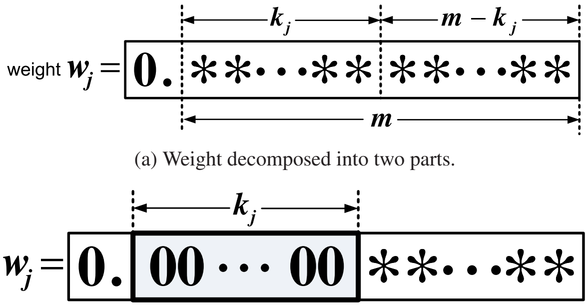
Thus we perform a bit-level transformation to them as follows:

$$\begin{aligned}
&-(2f_{0,(m-1)} + f_{1,(m-1)} + f_{2,(m-1)})\,2^{m-1} \\
&\quad = \left\{-2\left(1 - \overline{f_{0,(m-1)}}\right) - \left(1 - \overline{f_{1,(m-1)}}\right) - \left(1 - \overline{f_{2,(m-1)}}\right)\right\} 2^{m-1} \\
&\quad = -2^{m+1} + \overline{f_{0,(m-1)}}2^m + \left(\overline{f_{1,(m-1)}} + \overline{f_{2,(m-1)}}\right) 2^{m-1}.
\end{aligned} \tag{27}$$

By applying this transformation, the number of partial products required for $(-(2f_{0,(m-1)} + f_{1,(m-1)} + f_{2,(m-1)})\,2^m)$ is reduced from 12 to 4.

Assigning Eq. (27) to Eq. (24) leads to:

$$\begin{aligned}
&\text{Eq. (24)} \\
&= -f_0 - 2^{m+1} + \overline{f_{0,(m-1)}}2^m + \left(\overline{f_{1,(m-1)}} + \overline{f_{2,(m-1)}}\right) 2^{m-1} \\
&\quad + \sum_{i=0}^{m-1} \left(\underline{f_{0,(m-1)}w_{1,i} + f_{1,(m-1)}\overline{w_{1,i}}} + \underline{f_{0,(m-1)}w_{2,i} + f_{2,(m-1)}\overline{w_{2,i}}}\right) 2^{i-1} \\
&\quad + (f_{1,(m-1)} + f_{2,(m-1)})\,2^{-1} + \sum_{i=0}^{m-2} f_{0,i}2^{i-m+1} \\
&\quad + \sum_{i=0}^{m-2}\sum_{j=0}^{m-1} \left(\underline{f_{0,i}\overline{w_{1,j}} + f_{1,i}w_{1,j}} + \underline{f_{0,i}\overline{w_{2,j}} + f_{2,i}w_{2,j}}\right) 2^{i+j-m}.
\end{aligned} \tag{28}$$



(a) Weight decomposed into two parts.

(b) Limiting the first k_j bits as zero.

Fig. 5 A weights-range limit method.

For example, Eq. (24) with $n = 3$ and $m = 4$ originally generates 88 partial products. By using selector logics, they are reduced to 49 as in Figs. 4 (a) and 4 (b).

3.3 Reducing Partial Products by Using a Weights-range Limit Method

In this subsection, we propose a weights-range limit method that can reduce partial products by focusing on the negative term underlined with the wavy line in Eq. (28). First, we decompose the weight w_j defined by Eq. (15) as in **Fig. 5** (a) into the following two parts:

$$w_j = \sum_{i=0}^{m-1} w_{j,i}2^{i-m} = \sum_{i=m-k_j}^{m-1} w_{j,i}2^{i-m} + \sum_{i=0}^{m-1-k_j} w_{j,i}2^{i-m}. \tag{29}$$

Since Eq. (28) is transformed from Eq. (20), Eq. (28) can be transformed into:

$$\begin{aligned}
\text{Eq. (28)} &= \text{Eq. (20)} \\
&= (-1 + 2^{-m+1})f_0 \\
&\quad + \left\{ f_0 \sum_{i=0}^{m-1} \overline{w_{1,i}}2^{i-m} + f_1 \sum_{i=0}^{m-1} w_{1,i}2^{i-m} \right\} \\
&\quad + \left\{ f_0 \sum_{i=0}^{m-1} \overline{w_{2,i}}2^{i-m} + f_2 \sum_{i=0}^{m-1} w_{2,i}2^{i-m} \right\}.
\end{aligned} \tag{30}$$

Assigning Eq. (29) to Eq. (30) leads to:

$$\begin{aligned}
\text{Eq. (30)} &= f_0 2^{-m+1} + \underline{\left(-1 + \sum_{i=m-k_1}^{m-1} \overline{w_{1,i}}2^{i-m} + \sum_{i=m-k_2}^{m-1} \overline{w_{2,i}}2^{i-m}\right) f_0} \\
&\quad \underline{+ f_1 \sum_{i=m-k_1}^{m-1} w_{1,i}2^{i-m} + f_2 \sum_{i=m-k_2}^{m-1} w_{2,i}2^{i-m}} \\
&\quad + \left\{ f_0 \sum_{i=0}^{m-1-k_1} \overline{w_{1,i}}2^{i-m} + f_1 \sum_{i=0}^{m-1-k_1} w_{1,i}2^{i-m} \right\} \\
&\quad + \left\{ f_0 \sum_{i=0}^{m-1-k_2} \overline{w_{2,i}}2^{i-m} + f_2 \sum_{i=0}^{m-1-k_2} w_{2,i}2^{i-m} \right\}.
\end{aligned} \tag{31}$$

Let us focus on the underlined terms in Eq. (31). Assume that

$w_{j,i}$ $(m-k_j \le i \le m-1)$ for the weight w_j is zero as in Fig. 5 (b). In Fig. 5 (b), * shows 0 or 1 and the first k_j bits are zero. Then the underlined terms in Eq. (31) will be transformed into:

$$\begin{aligned}
&\left(-1+\sum_{i=m-k_1}^{m-1}\overline{w_{1,i}}2^{i-m}+\sum_{i=m-k_2}^{m-1}\overline{w_{2,i}}2^{i-m}\right)f_0\\
&\quad+f_1\sum_{i=m-k_1}^{m-1}w_{1,i}2^{i-m}+f_2\sum_{i=m-k_2}^{m-1}w_{2,i}2^{i-m}\\
&\quad=\left(-1+\sum_{i=m-k_1}^{m-1}2^{i-m}+\sum_{i=m-k_2}^{m-1}2^{i-m}\right)f_0\\
&\quad=(-1+1-2^{-k_1}+1-2^{-k_2})f_0\\
&\quad=(1-2^{-k_1}-2^{-k_2})f_0. \qquad (32)
\end{aligned}$$

How to Define k_j:

Now let us focus on the value k_j. Consider the case that w_0 is the maximum among w_0, w_1 and w_2[*6].

In this case, we have $w_1 < 1/2$ and $w_2 < 1/2$ since $w_0 + w_1 + w_2 = 1$. This means that the first bit of w_1 and w_2 must be zero, i.e., $w_{1,(m-1)} = w_{2,(m-1)} = 0$ in Eq. (15). Overall, this discussion leads to $k_1 = 1$ and $k_2 = 1$. Assigning $k_1 = k_2 = 1$, Eq. (32) = 0. Then we have

$$\begin{aligned}
\text{Eq. (30)} = f_0 2^{-m+1}&\\
+\left\{f_0\sum_{i=0}^{m-2}\overline{w_{1,i}}2^{i-m}+f_1\sum_{i=0}^{m-2}w_{1,i}2^{i-m}\right\}&\\
+\left\{f_0\sum_{i=0}^{m-2}\overline{w_{2,i}}2^{i-m}+f_2\sum_{i=0}^{m-2}w_{2,i}2^{i-m}\right\}&. \qquad (33)
\end{aligned}$$

Using Eqs. (22), (23) and (27) in the same way, (33) can be transformed into:

$$\begin{aligned}
&\text{Eq. (24)}\\
&=-f_{0,(m-1)}+\sum_{i=0}^{m-2}f_{0,i}2^{i-m+1}\\
&\quad+\sum_{i=0}^{m-1}\{f_{0,(m-1)}(-\overline{w_{1,i}})\\
&\quad+f_{1,(m-1)}(-w_{1,i})+f_{0,(m-1)}(-\overline{w_{2,i}})+f_{2,(m-1)}(-w_{2,i})\}2^{i-1}\\
&\quad+\sum_{i=0}^{m-2}\sum_{j=0}^{m-2}(f_{0,i}\overline{w_{1,j}}+f_{1,i}w_{1,j}+f_{0,i}\overline{w_{2,j}}+f_{2,i}w_{2,j})2^{i+j-m}\\
&=-\left(2f_{0,(m-1)}+f_{1,(m-1)}+f_{2,(m-1)}\right)2^{m-1}\\
&\quad+\sum_{i=0}^{m-2}(f_{0,(m-1)}w_{1,i}+f_{1,(m-1)}\overline{w_{1,i}}+f_{0,(m-1)}w_{2,i}\\
&\quad+f_{2,(m-1)}\overline{w_{2,i}})2^{i-1}\\
&\quad+\left(2f_{0,(m-1)}+f_{1,(m-1)}+f_{2,(m-1)}\right)2^{-1}-f_{0,(m-1)}\\
&\quad+\sum_{i=0}^{m-2}f_{0,i}2^{i-m+1}\\
&\quad+\sum_{i=0}^{m-2}\sum_{j=0}^{m-2}(f_{0,i}\overline{w_{1,j}}+f_{1,i}w_{1,j}+f_{0,i}\overline{w_{2,j}}+f_{2,i}w_{2,j})2^{i+j-m}\\
&=-2^{m+1}+\overline{f_{0,(m-1)}}2^m+\left(\overline{f_{1,(m-1)}}+\overline{f_{2,(m-1)}}\right)2^{m-1}\\
&\quad+\sum_{i=0}^{m-1}\Big(\underline{f_{0,(m-1)}w_{1,i}+f_{1,(m-1)}\overline{w_{1,i}}}\\
&\qquad\quad+\underline{f_{0,(m-1)}w_{2,i}+f_{2,(m-1)}\overline{w_{2,i}}}\Big)2^{i-1}\\
&\quad+\left(f_{1,(m-1)}+f_{2,(m-1)}\right)2^{-1}+\sum_{i=0}^{m-2}f_{0,i}2^{i-m+1}\\
&\quad+\sum_{i=0}^{m-2}\sum_{j=0}^{m-2}\left(\underline{f_{0,i}\overline{w_{1,j}}+f_{1,i}w_{1,j}}+\underline{f_{0,i}\overline{w_{2,j}}+f_{2,i}w_{2,j}}\right)2^{i+j-m}. \qquad (34)
\end{aligned}$$

Let us focus on underlined terms in Eq. (34). The terms have a form which we can apply selector logics to. Thus not only the weights-range limit method but also selector logics can be applied to weighted addition.

For example, Fig. 4 (b) has 49 partial products. Using the weights-range limit method above, they will be reduced to 33 as in Fig. 4 (c). We can expect that a weighted adder in super-resolution will be much faster than the one realized by a conventional method.

Weighted Adder for Reconstruction with n input images:

The discussion above can be applied similarly to weighted adders in super-resolution which require n input images. If the value k_j meets the condition below:

$$0 = 1-(2^{-k_1}+\cdots+2^{-k_{n-1}}), \qquad (35)$$

an weighted adder for reconstruction which requires n input images can be expressed as follows:

$$\begin{aligned}
\text{Eq. (9)} &= \left\{-(n-1)f_{0,(m-1)}-\sum_{i=1}^{n-1}f_{i,(m-1)}\right\}2^{m-1}\\
&\quad+\sum_{i=0}^{m-2}\sum_{j=0}^{n-1}\left\{f_{0,(m-1)}w_{j,i}+f_{j,(m-1)}\overline{w_{j,i}}\right\}2^{i-1}\\
&\quad+\sum_{i=0}^{m-2}f_{0,i}2^{i-m+1}+\sum_{i=0}^{n-1}f_{i,(m-1)}2^{-1}\\
&\quad+\sum_{i=1}^{n-1}\sum_{j=0}^{m-1-k_i}\sum_{l=0}^{m-1}\left(\overline{w_{i,j}}f_{0,l}+w_{i,j}f_{j,l}\right)2^{j+l-m}. \qquad (36)
\end{aligned}$$

4. Experimental Results

In this section, we demonstrate experimental evaluations. In this experiment, we assume that one pixel has a bit-length of eight ($m = 8$) for input images and the number of required input images for super-resolution based on a frequency-domain approach is seven ($n = 7$) according to [14]. We have compared our

*6 We cannot say that w_0 is always the maximum. If w_0 is not the maximum, we have to re-arrange the weights and signals accordingly so that w_0 is the maximum. If we know the maximum weight beforehand such as in the weight calculation, it is very easy to re-arrange them. But, if we do not have the maximum weight beforehand, we have to add an extra process to obtain the maximum weight. In our experiments in Section 4, we have obtained the weights of $w_0 = 0.359$, $w_1 = 0.186$, $w_2 = 0.185$, $w_3 = 0.092$, $w_4 = 0.079$, $w_5 = 0.053$, and $w_6 = 0.046$, and w_0 was the maximum of them.
Note that Section 3.2 can be applied to any case of weighted addition whatever weights we have.

Table 2 Experimental results.

Method	Delay time [ns]	Area [μm²]
Arithmetic operators	2.28 (100%)	5,718 (100%)
BLF	2.30 (101%)	8,879 (155%)
Redundant binary method 1	2.04 (89%)	7,258 (127%)
Redundant binary method 2	2.61 (114%)	9,025 (158%)
Proposed approach 1 (BLF+SL)	2.06 (90%)	4,942 (86%)
Proposed approach 2 (BLF+SL+WRL)	1.95 (86%)	3,316 (58%)

weighted adders with the ones as follows:

Arithmetic operators: In this method, we use conventional *arithmetic operators* such as plus (+), minus (−) and multiply (∗). In our experiences, Design Compiler using *arithmetic operators* synthesizes very much fast arithmetic circuits based on many optimizing techniques.

Bit-level transformation (BLF): In BLF method, the partial products generated by bit-level transformation in Eq. (24) are added up by Design Compiler.

Redundant binary method 1: In this method, the partial products in in Eq. (24) are added up by redundant binary adders (RBA) [6]. A redundant binary method generates carry propagation at most once by introducing redundant representations; $x \in \{-1, 0, 1\}$ and speeds up repeated arithmetic operations. By using a redundant binary addition tree, the weighted-sum can be faster because it includes many addition.

Redundant binary method 2: The result of Redundant binary method 1 is expressed by redundant binary form. It is necessary to decode redundant binary values into normal binary values. In addition to Redundant binary method 1 above, we add the decoder converting the result of Redundant binary method 1 into normal binary values.

Proposed method 1 (Bit-level transformation + Selector-logics (BLF+SL)):

In BLF+SL method, selector logics are applied to the partial products generated by Eq. (28) and they are added up by Design Compiler.

Proposed method 2 (Bit-level transformation + Selector-logics + Weights-range limit method (BLF+SL+WRL)):

In BLF+SL+WRL method, selector logics and the weights-range limit method are applied to the partial products generated by Eq. (34) and they are added up by Design Compiler*7.

We used Design Compiler Version B-2008.09-SP4 with the cell libraries in STARC CMOS 90nm to synthesize them where its objective function is to minimize their delays with no area constraints. Experimental results are shown in **Table 2**. Our proposed weighted adder (BLF+SL+WRL) has smaller delays than the ones using other designing methods. Comparing Our proposed method 2 (BLF+SL+WRL) with BLF and Proposed method 1 (BLF+SL), using selector logics improves the performance by 11.65% and using the weights-range limit method improves it by 5.641%.

Redundant binary method 1 has smaller delays than arithmetic operators, BLF and Proposed method 1 (BLF+SL). But Redundant binary method 2 has larger delays than the ones using other designing methods. Then we have compared our proposed methods 1 and 2 with the Redundant binary methods 1 and 2. Our proposed method 1 (BLF+SL) can improve the performance by 21.07% compared with Redundant binary method 2. Our proposed method 2 (BLF+SL+WRL) can improve the performance by 4.612% furthermore compared with Redundant binary method 1 and by a maximum of 25.29% compared with Redundant binary method 2.

5. Conclusions

In this paper, we proposed a fast weighted adder for reconstruction-based super-resolution. From the viewpoint of reducing partial products, we propose two approaches to speed up a weighted adder. First, we use selector logics to halve its partial products. Second, we propose a weights- range limit method utilizing negative term. Applying our proposed approaches to a weighted adder, we can reduce carry propagations and our weighted adder can be designed very fast compared to conventional ones.

Experimental results show that our proposed weighted adder (BLF+SL+WRL) improves its performance by a maximum of 25.29% and reduces its area to up to 1/3, compared to conventional ones.

In the future, we will design overall super-resolution hardware using our proposed weighted adder and demonstrate its effectiveness.

Acknowledgments This research was supported in part by KAKENHI (22300019).

References

[1] Ando, S.: A velocity vector field measurement system based on spatio-temporal image derivative, *Trans. Society of Instrument and Control Engineers*, Vol.22, No.12, pp.1330–1336 (1986).

[2] Aoki, S.: Super resolution processing by plural number of lower resolution images, *Ricoh Technical Report*, No.10, pp.19–25 (1998).

[3] Carlson, R.E. and Fritsch, F.N.: Monotone piecewise bicubic interpolation, *SIAM J. on Numerical Analysis*, Vol.22, No.2, pp.386–400 (1985).

[4] Keys, R.G.: Cubic convolution interpolation for digital image processing, *IEEE Trans. Acoust. Speech Signal Processing*, Vol.29, No.6, pp.1153–1160 (1981).

[5] Koga, H., Muta, K., Yokoyama, T. and Watanabe, T.h: Motion-blurred image restoration method using a super-resolution process, *Trans. Institute of Electronics, Information and Communication Engineers*, Vol. J90–D, No.6, pp.1532–1541 (2007).

[6] Kudou, T., Tsunekawa, Y. and Suzuki, M.: Structure of high-speed

*7 As discussed in Section 3.3, the weights-range limit method can be applied to the case where the weight w_0 is the maximum. In this sense, Proposed method 1 (BLF+SL) is more general than Proposed method 2 (BLF+SL+WRL).

modulo multiplier suitable for repeated operations, *Trans. Institute of Electrical Engineers of Japan C*, Vol.128, No.6, pp.933–942 (2008).

[7] Krithivasan, S., Schulte, M.J. and Glossner, J.: A subword-parallel multiplication and sum-of-squares unit, *Proc. ISVLSI*, pp.273–274 (2004).

[8] Lu, N.: *Fractal blowup*, pp.217–222, Fractal Imaging Academic Press San Diego (1997).

[9] Matsumoto, N. and Ida, T.: Reconstruction-based super-Resolution using self-congruency around image edges, *Trans. Institute of Electronics, Information and Communication Engineers D*, Vol.J93-D, No.2, pp.118–126 (2010).

[10] Octave, available from ⟨http://www.gnu.org/software/octave/⟩.

[11] Park, M.K., Kang, M.G. and Park, S.C.: Super-resolution image reconstruction: a technical overview, *IEEE Signal Process*, Vol.20, No.3, pp.19–36 (2003).

[12] Pasztor, E.C., Jones, T.R. and Freeman, W.T.: Example-based super-resolution, *IEEE Comput. Graph. Appl.*, Vol.22, No.2, pp.56–65 (2002).

[13] Tanaka, M. and Okutomi, M.: A fast algorithm for reconstruction-based super-resolution and its accuracy evaluation, *Trans. Institute of Electronics, Information and Communication Engineers D*, Vol.J88-D2, No.11, pp.2200–2209 (2005).

[14] Veterli, M., Vadewalle, P. and Süsstrunk, S.: A frequency domain approach to registration of aliased images with application to super-resolution, *EURASIP Journal on Applied Signal Processing*, Vol.20, No.3, pp.1–14 (2006).

Hiromine Yosihara received his B.Eng. from Waseda University in 2011. He is presently working toward M.Eng. degree there. His research interests include VLSI design, selector-logic applications and super-resolution. He is a student member of IEICE.

Masao Yanagisawa received his B.Eng., M.Eng., and Dr.Eng. degrees from Waseda University in 1981, 1983 and 1986, respectively, all in electrical engineering. He was with University of California, Berkeley from 1986 through 1987. In 1987, he joined Takushoku University. In 1991, he left Takushoku University and joined Waseda University, where he is presently a Professor in the Department of Electronic and Photonic Systems. His research interests are combinatorics and graph theory, computational geometry, VLSI design and verification, and network analysis and design. He is a fellow of IEICE and a member of IEEE and ACM.

Nozomu Togawa received his B.Eng., M.Eng., and Dr.Eng. degrees from Waseda University in 1992, 1994 and 1997, respectively, all in electrical engineering. He is presently a Professor in the Department of Computer Science and Engineering, Waseda University. His research interests are VLSI design, graph theory, and computational geometry. He is a member of IEEE and IEICE.

(Recommended by Associate Editor: *Shigetoshi Nakatake*)