Subpixel Interpolation Architecture for Multistandard Video Motion Estimation

Liang Lu, John V. McCanny, Fellow, IEEE, and Sakir Sezer

Abstract—A new reconfigurable subpixel interpolation architecture for multistandard (e.g., MPEG-2, MPEG-4, H.264, and AVS) video motion estimation (ME) is presented. This exploits the mixed use of parallel and serial-input FIR filters to achieve high throughput rate and efficient silicon utilization. Silicon design studies show that this can be implemented using 34.8 × 10⁶ gates with area and performance that compares very favorably with specific fixed solutions, e.g., for the H.264 standard alone. This can support SDTV and HDTV applications when implemented in 0.18-μm CMOS technology, with further performance enhancements achievable at 0.13-μm and below.

Index Terms—Motion estimation, reconfigurable architectures, system-on-a-chip (SoC), video compression.

I. INTRODUCTION

The increasing need to process real-time video streams captured using different coding standards (e.g., MPEG-2, MPEG-4, H.264, AVS) is usually accommodated by having separate processors within a single system-on-a-chip (SoC). In a recent publication [1], we have described a generic, template architecture for multimedia SoC design that can be electronically reconfigured to cover the common video compression standards and to handle the variable search ranges required. This incurs a relatively small penalty in terms of power dissipation and silicon area when compared with implementations for specific standards. Another important and related function is motion compensation (MC). The majority of early MC predictors only used integer-pixel accuracy. However, since image frame-to-frame displacements are unrelated to the sampling grid, improved prediction can be obtained when subpixel accuracy is employed, 2–4 dB improvements in Peak Signal Noise Ratio being achievable [2]. MPEG-2 and H.263, therefore, use displacement vectors with half-pixel resolution, with this having been extended to quarter-pixel resolution in MPEG-4, H.264, and AVS. Unfortunately, all these require different subpixel interpolation filters and this must be accommodated in any generalized approach. The purpose of this letter, therefore, is to present a new run-time reconfigurable architecture, capable of doing so and operating up to HDTV data rates.

II. INTERPOLATION FOR VARIOUS STANDARDS

All common video standards, typically, use 1-D finite impulse response (FIR) filters to perform subpixel interpolation on integer-pixels. For earlier standards (e.g., MPEG-1, MPEG-2, and H.263), where half-pixel accuracy is required interpolated subpixel samples are derived by applying a bilinear filter to both the vertical and horizontal directions as illustrated in Fig. 1. In H.264, the fractional-pixel MC interpolation is performed using two successive filtering operations, i.e., half-pixel then quarter-pixel interpolation. Here, half-pixel samples are derived using a 6-tap separable FIR filter, with a group of symmetrical coefficients (1, −5, 20, 20, −5, 1) [3], with this being applied to nearest integer-pixel samples in the horizontal and vertical directions (Fig. 2). Central half-pixel (CHP) samples, e.g., j, are then derived by applying the 6-tap filter to the intermediate values of the nearest half-pixel samples in either the horizontal or vertical direction, since either yields the same result. The intermediate results are then rounded and clipped to a correct luminance value between 0 and 255 for 8-bit color depth. Quarter-pixel samples are then computed by averaging two adjacent samples. In other standards, half-pixel sample computation processes are similar, but with different FIR filters used. For example, in MPEG-4, an 8-tap filter with coefficients (−1, 1, 3, −6, 20, 20, −6, 3, −1) is used, whilst in AVS a 4-tap filter with coefficients (−1, 5, 5, −1) is required. These symmetrical fixed coefficient filters are all modified versions of a Wiener filter prototype and chosen to reduce computational complexity (see Table I).

Fractional-pixel motion estimation (FME) represents a demanding real-time computational load and requires efficient hardware architectures, e.g., to meet HDTV requirements. Consideration needs also to be given to the interface of such circuits with circuits for real-time sum of absolute transformed difference (SATD) computation. An efficient interpolation module must typically produce and transfer data continuously since, as discussed previously [1], SATD computation units
are usually pipelined to achieve high throughput rates. Several recent publications on full-search FME architectures have also focused on subpixel interpolation [5]–[7], [9], [10]. In [5], [9], [10], a 4 × 4 block size interpolation module is described for H.264. Here, a group of 6-tap 1-D interpolation FIR filters are applied to a 10 × 10 integer sample block, with 5 filters used for the horizontal direction and 11 filters used for the vertical direction. All these have been designed with parallel-inputs and use adder and shifter operations for coefficient multiplication. This produces outputs every clock cycle and achieves SDTV performance with a relatively small silicon area. However, the 4-level adder structures employed result in long delay times, limiting operation frequency to below HDTV data rates. This delay path is reduced by a factor of 3 in the well known data broadcast/systolic like FIR architecture, although with the serial-input scheme leading to a proportional reduction in overall throughput. In [7], single input FIR filters, including the vertical and diagonal filters, are designed to process one input data value per clock cycle. This reduces the critical path and achieves a higher throughput rate than previously. However, a large number of costly initialization registers are required. All these architectures have been designed for H.264 alone, whilst our research has been motivated by the need to derive more flexible, reconfigurable circuits covering multiple standards.

### III. RECONFIGURABLE PARALLEL-INPUT FIR FILTER

The architecture proposed is based on a generic scheme that combines parallel and serial-input FIR filters. In a parallel-input FIR filter, all data is input to the filter on the same clock cycle. Fig. 3. Moreover, for the filters considered, coefficient symmetry allows the number of multipliers to be halved as shown below for an 8-tap filter. Here, \(x_i\), is the input pixel value, \(a, b, c, \) and \(d\) are the filter coefficients and \(y\) the output

\[
y = ax_0 + bx_1 + cx_3 + dx_4 + cx_5 + bx_6 + ax_7
\]

(1)

The predetermined nature of the coefficients can also be exploited to reduce hardware complexity as has been extensively discussed in the literature. For example, a fixed coefficient multiplier requires considerably less silicon area than a traditional ripple carry array [8]. It is also attractive in terms of implementing a design that can handle the small number of groups (e.g., 4) of coefficients needed to cover existing standards with these selected using a multiplexer, as shown in Fig. 3. Here, the output \(y\) is produced as an 8-bit value following normalization. Four registers are used to reduce the critical path delay and to enhance performance to a level commensurate with achieving HDTV data rates.

### IV. RECONFIGURABLE SERIAL-INPUT FIR FILTER

A generic serial-input filter covering all the video compression standards considered is shown in Fig. 4. This is based on the well known data broadcast/systolic like FIR architecture, in which data values are input and filter values output every cycle, the latter with an \(N\) (in this case 8) cycle latency, for an \(N\)-tap (8-tap) filter. Again, this exploits the symmetry of the filter coefficients, with a multiplexer also used to select these. For standards where the number of taps (i.e., multipliers) required is less than eight an enable signal can be used to switch these out to reduce power dissipation. A multiplexer can also be used to select the location in the delay line from which to derive the required output. For H.264 (6-tap filter), the output is derived from register \(D_0\) whilst in the case of a 2-tap filter this is \(D_{-1}\).
V. RECONFIGURABLE SUBPIXEL INTERPOLATION ARCHITECTURE

The serial-input and parallel-input filter architectures described each have their advantages. In this section, we describe a new reconfigurable subpixel interpolation architecture that combines the best attributes of both. For the purposes of explanation, a 4 × 4 luminance sample block is considered. We also firstly consider the most complex case, namely the 8-tap MPEG-4 filter. The generation of half-pixel sample values requires integer-pixel samples from both sides of the half-pixel location. Therefore, integer-pixel samples outside of the 4 × 4 block are required when the targeted half-pixel is located at the edge (Fig. 5). In this, the gray squares represent the 4 × 4 integer-pixel samples, the white squares represent the integer-pixel samples required in the interpolation process and the circles represent the half-pixel samples generated by the interpolation filters. Since the filter has eight taps, eight samples are required. For instance, to generate the gray circle (half-pixel sample) in the figure, the four white squares on its left and the four gray squares on its right in the same row are needed. The same approach also applies to the other half-pixel samples located on the right-hand boundary of the 4 × 4 block. Thus, the number of integer-pixel samples involved in half-pixel interpolation is (4+4+4) × (4+4+4) = 144. Therefore, for MPEG-4 half-pixel interpolation, a 12 × 12 sample block is required. Similarly, interpolation of a 4 × 4 block using 6-tap, 4-tap and 2-tap filters requires a 10 × 10, 8 × 8, and 6 × 6 sample blocks, respectively.

Before interpolation, the derivation of half-pixel samples is necessary with these being computed from integer-pixel samples at different locations. Half-pixels derived from integer-pixel samples in vertical direction are referred to as vertical half-pixel (VHP) samples, while half-pixels derived using integer-pixel samples in horizontal direction are referred to as horizontal half-pixel (HHP) samples. Correspondingly, half-pixels located in the center of four integer-pixel samples are referred to as CHP samples. These different sets of half-pixel samples are derived by different filters in our combined interpolation architecture, Fig. 6. This comprises five parallel-input FIR filters for VHP sample interpolation, six serial-input FIR filters for HHP samples interpolation and five serial-input FIR filters for CHP samples interpolation. Note that, in this figure, all buffers are used to store pixel values. For example, on each cycle an integer-pixel buffer receives an integer-pixel value that it then passes to the next integer-pixel buffer on the following clock cycle. For MPEG-4 (8-tap filter), 12 × 12 integer-pixel data values are input into and used for half-pixel sample computation. Therefore, on each clock cycle, a column of 12 pixels is input into the five parallel-input FIR filters for VHP sample interpolation. For example, in clock cycle 0, integer-pixel samples \( I_{0,0}, I_{0,1}, \ldots, I_{0,3} \) from Row 0 (R0), Row 1 (R1), \ldots, Row 7 (R7) respectively are transferred to FIR1 with the VHP sample being generated 2 cycles later and stored in a vertical half-pixel buffer. This is then used at a later
stage for CHP sample computation. Once the VHP samples are available, these are input sequentially through the vertical half-pixel buffers (right shifted in Fig. 6) into the five serial-input FIR filters to perform CHP sample interpolation, with the first CHP sample becoming available eight cycles later. At the same time, integer-pixel samples from Row3 (R3) to Row8 (R8) are also input into these six serial-input FIR filters for HHP sample generation. Similarly, it takes 8 clock cycles to generate the first HHP sample. Note that the serial-input FIR filters for data from R3 and R8 are used to generate half-pixel samples for quarter-pixel computation, if required. Thus, if only half-pixel accuracy is required, these two filters can be shut down to reduce power. The interpolation architecture is fully pipelined, with the generation of all the half-pixel samples requiring 14 clock cycles for MPEG-4. The overall data flow is summarized in Table II. Here, the gray blocks represent the pixels that are input or generated. Similarly, the number of cycles required for \(4 \times 4\) MPEG-2, A VS and H.264 blocks is 8, 10 and 12 respectively.

In this approach the use of parallel-input FIR filters avoids using integer-pixel initial buffers for the serial-input diagonal filter, as is the case of Yang et al. [7]. Not only does this save silicon area, it also avoids the initialization latency for these filters. Having one delay register in the parallel-input FIR filters avoids using integer-pixel initial buffers for serial-input FIR filters, thus extending the flexibility of these filters. The interpolation architecture is fully pipelined, with the generation of all the half-pixel samples requiring 14 clock cycles for MPEG-4. The overall data flow is summarized in Table II. Here, the gray blocks represent the pixels that are input or generated. Similarly, the number of cycles required for \(4 \times 4\) MPEG-2, A VS and H.264 blocks is 8, 10 and 12 respectively.

In this approach the use of parallel-input FIR filters avoids using integer-pixel initial buffers for the serial-input diagonal filter, as is the case of Yang et al. [7]. Not only does this save silicon area, it also avoids the initialization latency for these filters. Having one delay register in the parallel-input FIR filters avoids using integer-pixel initial buffers for serial-input FIR filters, thus extending the flexibility of these filters. The interpolation architecture is fully pipelined, with the generation of all the half-pixel samples requiring 14 clock cycles for MPEG-4. The overall data flow is summarized in Table II. Here, the gray blocks represent the pixels that are input or generated. Similarly, the number of cycles required for \(4 \times 4\) MPEG-2, A VS and H.264 blocks is 8, 10 and 12 respectively.

VI. Design Studies and Comparisons

The architecture in Fig. 6 has been captured in VHDL [VH-SIC (very high speed integrated circuits) hardware description language] and synthesized using Synopsis. The results obtained (Table III), are based on UMC’s 0.18 and 0.13 \(\mu\)m Complementary Metal-Oxide-Semiconductor (CMOS) Standard Cell libraries. The use of 0.18 \(\mu\)m CMOS enables direct comparison with previous research based on specific architectures (namely H.264). Since references [9], [10], uses the same fractional interpolation module architecture described in reference [5], only results for this have been included. As discussed, our target application is 720 pixels @ 30 MHz. In terms of previous publications, only [7] presents results suitable for this processing rate. The multistandard architecture proposed has to cover the case of an 8-tap MPEG-4 filter, whereas previous research has focused specifically on a simpler 6-tap H.264 filter. Thus, despite needing 33% additional registers, adders and multipliers the silicon area implementation overhead is only 6.6% when compared with Yang et al. [7]. As a result, when compared with previous H.264 architectures, our approach achieves the best throughput/area.

More importantly, it is the only one that can be configured for a wide range of video coding standards. The

<table>
<thead>
<tr>
<th>Clock cycle</th>
<th>Input</th>
<th>VHP</th>
<th>HHP</th>
<th>CHP</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Authorized licensed use limited to: Rensselaer Polytechnic Institute. Downloaded on February 20,2010 at 11:38:04 EST from IEEE Xplore. Restrictions apply.
0.13 µm CMOS results in Table III indicate that the maximum working frequency increases from 200 to 270 MHz allowing application to much higher resolution video sequences or higher frame rates and thus increased picture quality. This can be exploited in multiple channel applications, for example, the encoding or decoding of two SDTV real-time sequences simultaneously.

A detailed comparison of power dissipation/energy consumption between these architectures is not possible due to a lack of specific published data. However, for a fixed throughput rate the above comparisons indicate a lower area and/or frequency requirement in those circuits that have the highest throughput/unit area (fixed voltage). This, in turn, indicates that such circuits should be expected to have the lowest power dissipation/energy consumption, i.e., the circuits we present in this letter.

Whilst the focus of this letter has been on a subpixel interpolation architecture, the approach described can also be extended to FME systems. As discussed in references [5] and [7], it is the critical path of the interpolation unit that determines the throughput rate of an FME architecture. For example, in [7] a reduction from 10 to 4.04 ns in the filter critical path delay leads to an increase by a factor of over two in the block throughput rate. The analysis presented in Table III indicates higher values again are obtainable using circuits based on the architecture described.

In addition, ME systems typically comprise pipelined IME (integer-pixel motion estimation) and FME circuits operating in parallel [11]. The latter unit has the higher complexity and thus the use of an improved performance interpolation module is attractive in order to match FME and IME throughput rates. Thus in turn is attractive in terms of enhancing the performance of ME systems in general for different video standards.

### VII. Conclusion

In this letter, a new domain-specific reconfigurable subpixel interpolation architecture for multistandard video motion estimation or motion compensation has been presented. The mixed use of parallel-input and serial-input FIR filters achieves high throughput rate and efficient silicon area utilization. Flexibility has been achieved by a detailed examination of the interpolation filters used in these standards and by using a multiplexed reconfigurable data-path controlled by a selection signal. This compares very favorably with existing fixed solutions that are based solely on the H.264 standard. A detailed design study has shown that this can be implemented using \(3.8 \times 10^3\) gates. It can support SDTV and HDTV applications when implemented using 0.18 µm CMOS technology, with further performance enhancements achievable at 0.13 µm and below.

### REFERENCES


