CHAPTER 6

MAPPING OF REAL-TIME APPLICATIONS ON DRESPA AND ITS RESULTS

6.1 INTRODUCTION

The most common digital signal processing and digital image processing algorithms such as filtering, image transforms and motion estimation are considered for implementation on the proposed DRESPA architecture; specifically, IIR filter, 2-Dimensional Discrete Cosine Transforms and MPEG Motion Estimation. These algorithms have a high degree of data-parallelism and tight real-time constraints. Manual mapping of these algorithms on DRESPA is performed. The performance estimates of these implementations in terms of number of computation elements, throughput and speedup are also provided based on HDL simulations. The HDL simulation results are compared with MATLAB simulation.

6.2 IIR FILTERING ON DRESPA

IIR filters are used in many areas of technology such as sound and music enhancement, telecommunications, video and image processing, biomedical instrumentation etc.

The $N$th order IIR filter is defined as in Equation (6.1).
\[ H(Z) = \frac{\sum_{k=0}^{N} b_k Z^{-k}}{1 + \sum_{k=1}^{N} a_k Z^k} \]  \hspace{1cm} (6.1)

where, \( a_k \) and \( b_k \) are filter coefficients.

The transposed direct form II realization can be described by the set of difference equations represented by Equations (6.2) and (6.3).

\[ y(n) = w_1(n-1) + b_0 \times x(n) \]  \hspace{1cm} (6.2)

\[ w(n) = w_{k-1}(n-1) - a_k \times y(n) + b_k \times x(n) \]  \hspace{1cm} (6.3)

where, \( k = 1, 2, ..., N - 1 \)

Here, a fourth-order low pass IIR filter is considered for implementation. The direct form II structure for fourth-order IIR filter is shown in Figure 6.1.

The computation requires nine reconfigurable cells, out of which four are reused. The coefficients \( b_k \) and \( a_k \) are read in as constants from context word. The input data \( X_n \) is read through input unit and processed data \( Y_n \) is sent through output unit at each clock cycle. The latency time is two clock cycles. Initially the context with constants \( b_k \) and \( a_k \) are loaded into nine RCs. The input sample \( X_n \) is fed into five RCs as A input and B input of the RC is set to zero. The coefficients \( b_k \) are fed in as constants in context word. Similarly, next four RCs are fed with \( Y_n \) as A input and B input of the RC is set to zero. The coefficients \( a_k \) are fed in as constants in context word. These nine RCs are configured as multipliers. In the next clock cycle, the five RCs
are reconfigured as adders. This implementation takes one clock cycle to read in the data, two clock cycles to compute and one clock cycle for output. Input and output are pipelined. So, effectively two clock cycles are taken for computation. This mapping process is shown in Figure 6.2.

Figure 6.1 Fourth-order IIR filter
With 16-bit data, the throughput of fourth-order IIR filter is given by Equation (6.4).

\[
\text{Throughput} = \frac{16}{2} \text{ bits/cycle}
\]  

(6.4)

In this implementation, fourteen operations (nine multiplications and five additions) are performed as vector computation and four scalar operations (one input, one output and two context switches) are performed by the processor. Therefore, speedup is given by Equation (6.5).

\[
\frac{4}{18} \times S \leq 1
\]  

(6.5)

or \[ S \leq 4.5 \]  

(6.6)

The implementation results are summarized in Table 6.1.

**Table 6.1 DRESPA implementation of fourth-order IIR filter**

<table>
<thead>
<tr>
<th>Clock Period</th>
<th>13 ns</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of RCs</td>
<td>9</td>
</tr>
<tr>
<td>Latency</td>
<td>26 ns</td>
</tr>
<tr>
<td>Throughput</td>
<td>615 M bits per sec</td>
</tr>
<tr>
<td>Number of Clock cycles</td>
<td>2</td>
</tr>
<tr>
<td>Number of operations for RC</td>
<td>14</td>
</tr>
<tr>
<td>Number of operations for Input</td>
<td>1</td>
</tr>
<tr>
<td>Number of operations for Output</td>
<td>1</td>
</tr>
<tr>
<td>Number of Context Switches</td>
<td>2</td>
</tr>
<tr>
<td>Maximum Speedup</td>
<td>4.5</td>
</tr>
</tbody>
</table>
A fourth-order IIR low pass filter is simulated using MATLAB and the output for a stream of data is shown in Figure 6.3. The same IIR structure is mapped on to DRESPA. The simulation waveform is shown in Figure 6.4 and the results are identical with those obtained using MATLAB.

```
Command Window

Numerator Coefficients:

b =

2 -3 4 1 2

Denominator Coefficients:

c =

-1 9 4 -1

Input Data

x =

1 2 3 4 5 6 7 8 9 10 11 12 13 14

Output Data

y =

0 2 1 2 13 26 19 41 125 138 65 323 744 316 114

>>
```

Figure 6.3 MATLAB simulation of fourth-order LPF IIR
6.3 2-D DISCRETE COSINE TRANSFORM ON DRESPA

Two-Dimensional Discrete Cosine Transforms (2-D DCT) is most widely used in video and image compression. The algorithm used for this implementation is an integer DCT algorithm which is based on separable transforms. The mathematical expression for 1-D DCT computation of N-point data is given by Equation (6.7).

\[ y(k) = w(k) \sum_{n=0}^{N-1} x(n) \cdot \cos \left( \frac{(2n+1)k\pi}{2N} \right), \quad k = 0 \text{ to } N - 1 \]  

(6.7)

where,

\[ w(k) = \begin{cases} 
1, & k = 0 \\
\frac{1}{\sqrt{2}}, & k = 1 \text{ to } N - 1 
\end{cases} \]

Let \( c(k, n) \) represent the elements of cosine matrix \( C \), where

\[ c(k, n) = \cos \left( \frac{(2n+1)k\pi}{2N} \right), \quad 0 \leq k, n \leq N - 1 \]  

(6.8)

The 2-D DCT transform coefficients \( y_{ij} \) for 2-D image \( x_{ij} \), \( 0 \leq i, j \leq N - 1 \) are computed using the cosine matrix, \( C \) as shown in Equations (6.9) and (6.10).

\[ M = C \times X \]  

(6.9)

\[ Y = M \times C^T \]  

(6.10)

where \( Y \) is a \( N \times N \) matrix containing the 2-D DCT transform coefficients. For \( N = 4 \), the required \( C \) matrix is shown in Equation (6.11).
The image is divided into non-overlapping blocks each of size $4 \times 4$. Using properties of separable transforms, 1-D DCT for each row in the $4 \times 4$ block is computed with $N = 4$. Then, 1-D DCT is computed for each column of the resulting matrix to get 2-D DCT.

The 1-D $4 \times 4$ integer DCT requires twenty eight reconfigurable cells for computation within two clock cycles. The $4 \times 4$ pixel block $P_i$ is first preloaded into data buffer. The cosine coefficients $c(k, n)$ are read in as constants form context word. Sixteen reconfigurable cells are configured as buffer. The 1-D transformed output is stored in the data buffer after three clock cycles. During first clock cycle, data buffering is done in sixteen reconfigurable cells. During the second clock cycle, the other eight reconfigurable cells are configured as adders. During the third clock cycle the remaining four reconfigurable cells are configured as shifters (dividers). This mapping process is shown in Figure 6.5.

For a $256 \times 256$ grayscale image, there are 4096 blocks each of size $4 \times 4$. In three clock cycles, one $4 \times 4$ block is processed along the row. In the next three clock cycles, one $4 \times 4$ block is processed along the column. To process 4096 blocks, 24576 clock cycles are required. With 16-bit data, the throughput of 2-D DCT algorithm is given by Equation (6.12).

$$\text{Throughput} = \frac{8 \times 256 \times 256}{24576} \text{ bits/cycle} \quad (6.12)$$
Figure 6.5 Mapping of integer 2-D DCT on DRESPA

In this implementation, sixteen operations (sixteen multiplications) are performed as vector computation and one context switch operation is performed by the processor. So, speedup is given by Equation (6.13).

\[
\frac{1}{17} \times S \leq 1 \tag{6.13}
\]

or

\[
S \leq 17 \tag{6.14}
\]

The implementation results are summarized in Table 6.2.
Table 6.2 DRESPA implementation of 2-D DCT algorithm

<table>
<thead>
<tr>
<th>DESCRIPTION</th>
<th>VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Clock Period</td>
<td>13 ns</td>
</tr>
<tr>
<td>Number of RCs</td>
<td>112</td>
</tr>
<tr>
<td>Throughput</td>
<td>1.6 G bits per sec</td>
</tr>
<tr>
<td>Number of Clock cycles</td>
<td>24576</td>
</tr>
<tr>
<td>Number of operations for RC</td>
<td>16</td>
</tr>
<tr>
<td>Number of operations for Input</td>
<td>0</td>
</tr>
<tr>
<td>Number of operations for Output</td>
<td>0</td>
</tr>
<tr>
<td>Number of Context Switches</td>
<td>1</td>
</tr>
<tr>
<td>Maximum Speedup</td>
<td>17</td>
</tr>
</tbody>
</table>

A 2-D $4 \times 4$ integer DCT is simulated using MATLAB and the output for a stream of data is shown in Figure 6.6. The same is mapped onto DRESPA. The simulation waveform is shown in Figure 6.7 and the results are found to be identical with those obtained using MATLAB.
Figure 6.6  MATLAB simulation of 2-D integer DCT (a) Input data sequence (b) Output data sequence
6.4 MPEG MOTION ESTIMATION ON DRESPA

In MPEG encoder for video compression, after conversion from RGB to YCbCr followed by preprocessing and subsampling, the next step is motion estimation. Motion estimation of image pixels is done to remove temporal redundancies in successive P-type and B-type frames. Block matching algorithm is used for motion estimation.

The algorithm is formulated using absolute difference criteria given by Equation (6.15).

\[
e(m,n) = \sum_{i=1}^{M} \sum_{j=1}^{N} |R(i,j) - C(i+m,j+n)|
\]  

(6.15)

where, \(R(i,j)\) is the reference frame block of size \(M \times N\) pixels at coordinates \((i,j)\) and \(C(i+m, j+n)\) is the candidate frame block within the search area in the previous frame. The motion vector is determined by the least \(e(m,n)\) value among all the \(M \times N\) possible displacements within the search area. Usually \(M = N\).

For each reference frame block, five consecutive candidate frame blocks are matched concurrently. Initially, one reference block (256 Bytes) and all the first rows of five candidate blocks (80 Bytes) from the search area associated with it are loaded into one set of data buffer. The RC starts the matching process for this reference block. During this computation, another reference frame block and the search area associated with it are loaded into another set of data buffer. Thus, data loading and computation are done simultaneously. Typically, \(M = N = 16\) is chosen, so that block size is \(16 \times 16\) pixels. Each sixteen RCs in the first, fourth, seventh, tenth and thirteenth rows
perform the subtraction and find the absolute value in Equation (6.15) during first clock cycle.

6.4.1 Clock Cycle 1 Computation

Row 1 RCs performs subtraction
\[ RC(1,c) = |R(1, j) - C(1, j)|, \ c, j = 1 \text{ to } 16 \] (6.16)

Row 4 RCs performs subtraction
\[ RC(4,c) = |R(1, j) - C(2, j)|, \ c, j = 1 \text{ to } 16 \] (6.17)

Row 7 RCs performs subtraction
\[ RC(7,c) = |R(1, j) - C(3, j)|, \ c, j = 1 \text{ to } 16 \] (6.18)

Row 10 RCs performs subtraction
\[ RC(10,c) = |R(1, j) - C(4, j)|, \ c, j = 1 \text{ to } 16 \] (6.19)

Row 13 RCs performs subtraction
\[ RC(13,c) = |R(1, j) - C(5, j)|, \ c, j = 1 \text{ to } 16 \] (6.20)

Each sixteen RCs in second, fifth, eighth, eleventh and fourteenth row performs the addition of all sixteen absolute values from previous row RCs in fifteen clock cycles.

6.4.2 Clock Cycle 2 to 16 Computation

Row 2 RCs performs addition
\[ RC(2,c) = \sum_{c=1}^{16} RC(1, c) \text{ outputs }, \ c = 1 \text{ to } 16 \] (6.21)
Row 5 RCs performs addition

\[ RC(5, c) = \sum_{c=1}^{16} RC(4, c) \text{ outputs}, \ c = 1 \text{ to } 16 \quad (6.22) \]

Row 8 RCs performs addition

\[ RC(8, c) = \sum_{c=1}^{16} RC(7, c) \text{ outputs}, \ c = 1 \text{ to } 16 \quad (6.23) \]

Row 11 RCs performs addition

\[ RC(11, c) = \sum_{c=1}^{16} RC(10, c) \text{ outputs}, \ c = 1 \text{ to } 16 \quad (6.24) \]

Row 14 RCs performs addition

\[ RC(14, c) = \sum_{c=1}^{16} RC(13, c) \text{ outputs}, \ c = 1 \text{ to } 16 \quad (6.25) \]

The comparison of error values is done in four RCs of sixteenth row in four clock cycle.

### 6.4.3 Clock Cycle 17 to 20 Computation

Row 16 Column 1 RC performs comparison

\[ RC(16,1) = \text{MAX}(\text{RC}(2, 16) \text{ output}, \text{RC}(5, 16) \text{ output}) \quad (6.26) \]

Row 16 Column 2 RC performs comparison

\[ RC(16,2) = \text{MAX}(\text{RC}(16, 1) \text{ output}, \text{RC}(8, 16) \text{ output}) \quad (6.27) \]

Row 16 Column 3 RC performs comparison

\[ RC(16,3) = \text{MAX}(\text{RC}(16, 2) \text{ output}, \text{RC}(11, 16) \text{ output}) \quad (6.28) \]
Row 16 Column 4 RC performs comparison

\[ RC(16,4) = \text{MAX}( RC(16, 3) \text{ output}, RC(14, 16) \text{ output} ) \]  \hspace{1cm} (6.29)

It takes twenty clock cycles to finish matching of five consecutive blocks. Sixteen clock cycles are needed for comparing sixteen \(e(m,n)\) values to update motion vectors for the best match. One extra clock cycle is needed for buffering pixel values in data buffer. So, thirty seven clock cycles are required for matching one candidate frame block with five reference frame blocks. For a frame size of \(352 \times 240\), there are 330 blocks. Therefore, for matching one candidate frame block with 330 reference frame blocks, 2442 clock cycles are required. For matching 330 such candidate frame blocks, 805860 clock cycles are required.

With 16-bit data, the throughput of motion estimation algorithm is given by Equation (6.30).

\[
\text{Throughput} = \frac{16 \times 330 \times 16 \times 16 \times 5}{805860} \text{ bits/cycle} \hspace{1cm} (6.30)
\]

In this algorithm, 256 operations are performed as vector computation and seventeen scalar operations (one input, one output and fifteen context switches) are performed by the processor. Therefore, speedup, \(S\), is given by Equation (6.31).

\[
\frac{17}{273} \times S \leq 1 \hspace{1cm} (6.31)
\]

or

\[
S \leq 16.06 \hspace{1cm} (6.32)
\]

The implementation results are summarized in Table 6.3.
Table 6.3 DRESPA implementation of motion estimation algorithm

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Clock Period</td>
<td>13 ns</td>
</tr>
<tr>
<td>Number of RCs</td>
<td>164</td>
</tr>
<tr>
<td>Throughput</td>
<td>645 M bits per sec</td>
</tr>
<tr>
<td>Number of Clock cycles</td>
<td>805860</td>
</tr>
<tr>
<td>Number of operations for RC</td>
<td>256</td>
</tr>
<tr>
<td>Number of operations for Input</td>
<td>1</td>
</tr>
<tr>
<td>Number of operations for Output</td>
<td>1</td>
</tr>
<tr>
<td>Number of Context Switches</td>
<td>15</td>
</tr>
<tr>
<td>Maximum Speedup</td>
<td>16.06</td>
</tr>
</tbody>
</table>

6.5 CONCLUSION

In this chapter, the performance of the proposed DRESPA architecture is evaluated by executing most common digital signal processing and digital image processing algorithms such as IIR filter, 2-D DCT and MPEG Motion Estimation on DRESPA. The software required to support the architecture is developed for programming the system. These algorithms are also simulated using MATLAB.

DRESPA simulation results are compared with those obtained using MATLAB for the same input data. The results obtained confirm the performance improvement achieved with the proposed architecture in real-time signal processing applications.

The performance of DRESPA is assessed in terms of number of RCs, throughput, computation time, number of context switches and Speedup.
The performance analysis of the applications implemented on DRESPA is summarized in Table 6.4.

Table 6.4  DRESPA performance analysis

<table>
<thead>
<tr>
<th>Parameters</th>
<th>4th order LPF IIR</th>
<th>4×4 2D DCT</th>
<th>MPEG Motion Estimation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Clock Period</td>
<td>13 ns</td>
<td>13 ns</td>
<td>13 ns</td>
</tr>
<tr>
<td>Number of RCs</td>
<td>9</td>
<td>112</td>
<td>164</td>
</tr>
<tr>
<td>Latency</td>
<td>26 ns</td>
<td>26 ns</td>
<td>26 ns</td>
</tr>
<tr>
<td>Throughput</td>
<td>615 M bits/ sec</td>
<td>1.6 G bits/ sec</td>
<td>645 M bits/ sec</td>
</tr>
<tr>
<td>Number of Clock cycles</td>
<td>2</td>
<td>24576</td>
<td>805860</td>
</tr>
<tr>
<td>Number of operations for RC</td>
<td>14</td>
<td>16</td>
<td>256</td>
</tr>
<tr>
<td>Number of operations for Input</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Number of operations for Output</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Number of Context Switches</td>
<td>2</td>
<td>1</td>
<td>15</td>
</tr>
<tr>
<td>Maximum Speedup</td>
<td>4.5</td>
<td>17</td>
<td>16.06</td>
</tr>
</tbody>
</table>