RAPID PROTOTYPING OF DEDICATED SYSTEMS BASED ON BIT-Serial UNIVERSAL ARCHITECTURE

G. Rubin, A. Petrovsky

Bialystok Technical University, Computer Science Department
Wiejska 45A, Bialystok, Poland
e-mail: gregori@wi.pb.edu.pl; palex@bsuir.by

The aim of this paper is to present the method of rapid prototyping for reducing development cost of dedicated systems. Prototypes are build to assess whether proposed system will be acceptable to its user and whether, a proposed design will provide adequate functionality and performance. In this paper the designing method is proposed, for real-time embedded systems. This approach is based on modeling using modification of Petri nets called hardware Petri nets. Implementation is made using special hardware bit-serial universal architecture. The main advantage is that, there is no need to change hardware, even all used algorithms must be changed. There is a possibility to make all design process to be almost fully automated from modeling to working prototype.

Keywords – Rapid prototyping, FPGA, bit-serial architecture

1 INTRODUCTION

Rapid prototyping aims to reducing development cost via prototyping[1]. A prototype is constructed prior to the system's production version to gain information that guides analysis and design. This paper presents the method for design real-time dedicated systems. This approach is based on modeling using modification of Petri nets called Hardware Petri Nets (HPN)[4]. Implementation is made using special bit-serial universal hardware architecture. The main advantage of proposed method is that, there is constant hardware platform. Algorithm implementation requires only to find time scheduling. There is a possibility to make all design process to be almost fully automated from modeling to working prototype. According to the figure 1, rapid prototyping stages consists class of task, based on that the hardware architecture is designed. The next step is design of HPN model depended on task. The time schedule is a result of HPN model simulation and will be used for control unit of the hardware platform. At the end of prototyping process working application of prototype according to specification should be achieved, if not, then HPN model needs to redesign. Redesign of HPN model is better way than redesign of hardware because of time. This paper presents an example of rapid prototyping, from concept to application.

Fig. 1. Stages of Rapid prototyping.

2 BIT-Serial UNIVERSAL ARCHITECTURE

Shared-memory architecture (SMA) approach is detailed in [2]. Figure 2 shows one of that architecture type. The idea is very simple. In order to simultaneously provide the PEs (Processing Elements) with input data, the shared-memory is partitioned into blocks. PEs usually perform simple memoryless mapping of the input values to a single output values. Using a rotating access scheme, each processor gets access to the memories once per \( N \) (\( N \) - number of PEs') cycles. During this time processor either writes or reads data from memory. All processors have the same duration time slot to access to the memories and access conflict is completely avoided. Examples of usage and algorithm implementations can be found in [3]. In this paper some modifications of architecture were made. Separate input and output registers were replaced by registers with input/output interface. There is possible to save result of serial calculation into free cells of input registers during shifting data into processing elements. The input/output registers are 16-bit width of data. Proposed architecture is universal for dedicated system, so specialized algorithms for multimedia systems could be implemented. The hardware of architecture is constant and functionality depends on input vectors according to demanded tasks or instructions. For easy and fast of input vectors generation special toolbox should be used. One of possible solutions, for parallel processes, is modified Petri Nets theory appliance. An example can be found in the next chapter.
Moreover there is possibility to design hierarchical structure of the net. Every transition and place can be designed as the subnet. Simulation of hierarchical structure is also possible. The control vectors of simulation result can be written as a text file, then control vectors can be loaded to Xilinx simulator for FPGA device simulation. Such approach allows for better scheduling, because of running every elements of architecture as fast as it's possible, when proper data are ready for processing.

Fig. 3. Example of time scheduling for shared memory architecture.

Fig. 4. Hardware Petri Net model for universal architecture.
4 HARDWARE IMPLEMENTATION

The hardware platform of universal architecture was synthesized with Xilinx ISE 10.1 for Virtex II xc2v3000. Top design is schematic form and realized according with proposition on fig. 2. Table 1 shows device utilization summary.

Implementation of processing elements (fig. 5.a) consists bit-serial multipliers, which are in practice based on the shift-and-add algorithm, where several bit-products are added in each time slot. Bit-serial addition (fig. 5.b) is realized by equation below:

\[
\begin{align*}
S = \text{XOR}(A, B, C_i) = A \oplus B \oplus C_i \\
C_i = \text{Majority}(A, B, C_i) = \ \\
\ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ A_i \cdot B_i + A_i \cdot C_i + B_i \cdot C_i \\
\ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ D_i = C_i, \text{ for } i = W_d - 1, W_d - 2, \ldots, 0 \\
D_{W_d} = 0
\end{align*}
\]

where: \( W_d \) - data word length.

![Diagram of processing element architecture](image)

![Diagram of bit-serial adder](image)

The \( n \)-bit serial multiplier is depicted on figure 6. At the first step \( m \)-Register (shift register with serial input and parallel output) takes less significant bit of multiplier, and first delay from element \( D_1 \) (less significant bit of multiplicand). Unit delay \( D_1 \) holds bits for one clock cycle and \( D_2 \) unit two cycles respectively.

![Diagram of bit-serial multiplier](image)

The first bit (less significant) of output appears after \( 2n \) clock cycles. For example 8-bit serial multiplication takes 16 clock cycles. Presented parts of SMA unit was connected to control unit block, which loads control vectors from the text file. The vectors of time scheduling were generated by simulation, using Petri Nets toolbox. The control unit reads line by line and puts the 46-bit vector into “CU vector” bus. There is only one main clock signal for SMA architecture.

### DEVICES UTILIZATION SUMMARY

<table>
<thead>
<tr>
<th>Device Type</th>
<th>Utilization</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of Slices</td>
<td>123 out of 14336 8%</td>
</tr>
<tr>
<td>Number of Slices Flip Flops</td>
<td>578 out of 28672  7%</td>
</tr>
<tr>
<td>Number of 4 input LUTs</td>
<td>2192 out of 28672 7%</td>
</tr>
<tr>
<td>Number of bowed I/Os</td>
<td>173 out of 684  25%</td>
</tr>
<tr>
<td>Number of 4 BUF MUUXs</td>
<td>1 out of 16 6%</td>
</tr>
</tbody>
</table>

5 RAPID PROTOTYPING OF 4X4 PARALLEL TRANSFORM FOR H.264/AVC

The H.264/AVC standard is known as: ISOMPEG4 Part 10, ITU-T H.264, and the Advanced Video Coding (AVC)[5]. The transform block (Residual Transform) in H.264/AVC is one of the key components and there are several aspects of its design that are considered[6]. The 4x4 residual data transform takes up majority of computation, therefore it will be used as an example. The authors of H.264/AVC start with a well known two dimensional Discrete Cosine Transform (DCT). This transform can be represented by:

\[
Y = AX \cdot \mathcal{F} = \begin{bmatrix}
a & b & a & c \\
-\frac{1}{\sqrt{2}} \cos \left( \frac{\pi}{8} \right) & \frac{1}{\sqrt{2}} \cos \left( \frac{3\pi}{8} \right) & c & a - \frac{1}{\sqrt{2}} \cos \left( \frac{\pi}{8} \right)
\end{bmatrix} X
\]

where:

\[
a = \frac{1}{2}, \quad b = \frac{1}{\sqrt{2}} \cos \left( \frac{\pi}{8} \right), \quad c = \frac{1}{\sqrt{2}} \cos \left( \frac{3\pi}{8} \right)
\]

The final forward transform based on [9] and looks as follows:

\[
Y = (x \cdot x^T) \cdot \mathcal{F} = \begin{bmatrix}
1 & 1 & 1 & 1 \\
2 & 1 & -1 & 2 \\
1 & -1 & -1 & 1 \\
1 & -2 & -2 & 1
\end{bmatrix} X
\]

\[
C \cdot X^T \text{ part is the core of transform and can be carried out with integer arithmetic using only additions, subtractions and shifts. The last operation } \oplus E \text{ requires multiplication, which can be done into the quantization process.}
\]

Several hardware design methods for the implementation of the 2-D integer transform have been developed in recent years. In this paper an architecture which performs one dimensional transform on a column of input data by a matrix multiplication architecture was used [9]. Figure 7 and formula (7) show 1-D transform. That operation needs to be performed four times along the vertical dimension, and four times along the horizontal dimension on X. Each of these eight 1-D column/row transforms requires four adders and four subtractors.
This example is based on that transform, but hardware implementation uses shared memory architecture approach. Moreover, proposed approach uses bit-serial arithmetic and synchronous calculations.

\[ \begin{align*}
    x_i &= (x_i + x_i) + (x_i + x_i), \\
    x_i' &= (x_i - x_i) - (x_i - x_i), \\
    x_i &= (x_i - x_i) + (x_i - x_i) - 2,
\end{align*} \]

(7)

The transformations were implemented on universal architecture (Fig. 2). Two blocks of data

\[ X_1 = \begin{bmatrix}
    1 & 6 & 11 & 16 \\
    2 & 7 & 12 & 17 \\
    5 & 10 & 15 & 20 \\
    8 & 13 & 18 & 23
\end{bmatrix}, \quad X_2 = \begin{bmatrix}
    255 & 226 & 225 & 226 \\
    224 & 225 & 226 & 226 \\
    224 & 226 & 226 & 226 \\
    224 & 225 & 228
\end{bmatrix} \]

(8)

were set as input and the result was purchased:

\[ Y_1 = \begin{bmatrix}
    164 & -140 & 0 & -20 \\
    -28 & 0 & 0 & 0 \\
    -12 & 0 & 0 & 0 \\
    16 & 0 & 0 & 0
\end{bmatrix}, \quad Y_2 = \begin{bmatrix}
    3605 & -18 & 1 & -9 \\
    -1 & 15 & -3 & 0 \\
    3 & 0 & 3 & -5 \\
    2 & 5 & -4 & 5
\end{bmatrix} \]

(9)

6 SUMMARY

In this paper, was presented the method of rapid prototyping for reducing development cost of dedicated systems. As an example parallel 4x4 integer transform was used. Based on bit-serial shared memory architecture, universal structure was design to improve processing rate for H.264/AVC and reduce power consumption. Reducing the power consumption was achieved by bit-serial communications as an alternative to bit-parallel interconnects. Simulations results of proposed rapid prototyping method shows accurate approach for design dedicated systems using bit-serial universal architecture.

REFERENCES