Design Approach for a Low Power Video Decoder


Center of Electrical Engineering and Computing, Campina Grande, PB, 58102-012, Brazil
{elmar,henrique,maria,fabricio,matheus}@lad.dsc.ufcg.edu.br

Abstract — This paper presents a detailed description of the design of a Low Power, SystemC-RTL, OCP-IP compliant MPEG-4 decoder IP-core, named Terpsicore. The Terpsicore was designed for use in embedded systems where low cost, high performance and power efficiency are important. This design supports simple profile L0 with two external memories, was manufactured using a 0.35um CMOS 4ML technology and has a die of 49mm² comprising 48095 gates. It consumes 115mW at 25MHz when video encoding of QCIF at 15fps with a voltage supply of 3.3V. Compared with other implementations of the MPEG-4 standard available in the market, the Terpsicore presents better energy efficiency and this paper shows how this was achieved.

I. INTRODUCTION

Recently, markets for mobile electronic devices, such as cellular phones and portable PC, are currently growing quickly. Multimedia processing is an essential functionality in such mobile-device applications. The key technologies to success in mobile multimedia markets are low power dissipation and high cost effectiveness. To achieve these requirements, both algorithm and chip implementations have to be improved for these applications.

MPEG-4 is an open standard that was created by the Motion Picture Experts Group (MPEG) which came to replace the MPEG-2 standard. It provides that motion video can be manipulated as a form of computer data and can be stored, transmitted and received over existing and future networks or broadcast channels [3].

The MPEG-4 coding can be done in various profiles and levels to fit a variety of applications. Some of these applications can take advantage from a VLSI implementation with optimization in cost and power dissipation. Terpsicore (this name is a reference of the muse of music and dance from the Greek mythology) implements Simple Profile Level 0 with QCIF size and 15fps.

Terpsicore is an implementation of the MPEG-4 decoder standard that was designed for use in mobile applications where low cost, real time performance and low power dissipation are needed. In 0.35μm technology, its design consumes 115mW at 25MHz.

Terpsicore uses an interface according to the Open Core Protocol International Partnership (OCP-IP) [1]. Terpsicore is implemented using SystemC [2], a language that provides hardware-oriented constructs within the context of C++ as a class library implemented in standard C++. These constructs are powerful enough to let users describe their models in RTL or system-level. SystemC also supports verification through the SystemC Verification (SCV) library [2].

The next section contains a description of the state-of-the-art implementations of the MPEG-4 decoding standard so that a comparison with Terpsicore can be made, in the third it is shown Terpsicore’s architecture, and the last one presents the conclusions achieved in section fourth it is shown a comparison of Terpsicore with other implementations in terms of normalized power consumption and normalized area which indicates that Terpsicore has a very optimized design.

II. OTHER IMPLEMENTATIONS OF MPEG-4 STANDARD

There are many implementations of the MPEG-4 standard available in the market. Here we’ll discuss some of them to compare their functionality and performance with Terpsicore’s. We’ll divide the implementations into two categories: the programmable and the dedicated/hybrid approaches.

A. Programmable Approaches

The programmable approaches can be divided into four categories [6]:

1. Based on general-purpose processors (GPP) with media instruction-set architecture;
2. Based on embedded RISC processors;
3. Based on traditional DSP architectures;
4. Based on special-mostly very-long-instruction-word-media-processors

Although these approaches introduces a high-level programmability and generally a low cost, they generally
lack on performance, which is a major feature for more complex applications in video processing.

Among the known decoders available, there is an implementation using the ARM7TDMI that comprises simple profile at level 1 operating in a 20MHz clock [6]. An example of implementation using DSP architecture is NEC’s uPD77110 for a L1 QCIF 15 fps codec at 75 MHz [6].

B. Dedicated and Hybrid Approaches

There are a number of implementations that fits in this kind of approach and it must be emphasized that the majority of them has a hybrid architecture which are based on the combination of RISC or a DSP for control tasks and a number of dedicated hardware accelerators for encoding/decoding. Most of the architectures aim for mobile applications where low cost, high performance and low power dissipation are strongly recommended. The drawback of this kind of implementation is the low-level of programmability.

A fully dedicated approach is Fujitsu’s MPEG-4 video LSI introduced in 2002, which supports simple profile L3 with simultaneous encoding and decoding. Its video codec core is announced to have a power dissipation of 9mW for QCIF at 15 fps and 29mW for CIF at also 15 fps with a voltage supply of 1.5 V and a clock frequency of 13.5 MHz. The MPEG-4 video LSI was manufactured in a 0.18μm CMOS 4 ML technology and has a die size of 28mm² comprising 700k gates [6].

Toshiba’s TC35273 is a hybrid codec solution that comprises a RISC processor and other 6 accelerators. It works with simple profile at level 1 and operates at a 70MHz clock. From Toshiba there’s also the TC35274, which performs MPEG4 decoding 15 fps in QCIF at 30MHz [6].

Matsushita’s developed a hybrid decoder for the simple profile at level 1. This one is a combination of a DSP and other 4 coprocessors. All units run at a 27MHz clock and the power dissipation is stated to be 11 mW at 1.5V. The chip was manufactured using a 0.18μm CMOS 4 ML technology and has a die size of 37 mm² [5].

Arakida’s MPEG-4 audiovisual LSI, which supports simple profile L2 with 16Mb embedded DRAM, was manufactured using a 0.13μm CMOS triple well 5ML technology and has a die size of 6.56mm x 6.56mm comprising 3M gates. This chip integrates 4 16b RISC processors dedicated hardware accelerators including a 5GOPS adaptative filter engine. It consumes 160mW at 125MHz when video encoding of CIF at 15fps with a voltage supply of 1.5V [4].

Takahashi’s MPEG-4 video codec has been developed for mobile multimedia applications and using clustered voltage scaling with variable internal supply voltage of 2.5 and 1.75V. This codec works with simple profile level 1 with QCIF resolution at 10fps and operates at a 30MHz clock with power dissipation of 60mW. This chip was fabricated in a 0.3μm CMOS double well 3ML technology and has a die size of 81mm² [8].

III. TERPSICORE DESIGN ARCHITECTURE

The MPEG standard specifies the functions to be implemented in order to decode video in this format, but it doesn’t impose a structure for the architecture. Terpsicore’s architecture is divided in four major blocks: Bitstream processing, texture decoding, motion estimation and image composition as shown in Figure 1.

The bitstream demultiplexer receives the compressed video stream in the MPEG-4 format and feeds the other blocks with the proper data and/or configuration parameters so that each one is able execute its function. In fact, this block is a 16-bit dedicated processor (called 3MBIP that stands for Mobile Multimedia Bitstream Processor) implemented along with Terpsicore to decode the video header that contains the configuration parameters of all other blocks and to decode the VLC (Variable Length Coding) data. The architecture for 3MBIP is shown in Figure 2. And its operation is as follows: The DIS block is a 32-bit register responsible for receiving the video streaming and discarding each bit until he identifies that the current 32 bits corresponds to a “start code”.

After this code is found the DIS block continues receiving the following bits and the header decoding can go on. The firmware memory contains the algorithm used for demultiplexing the MPEG-4 video stream. This
algorithm is implemented in only 16KB of code for a specially designed CISC processor.

Fig. 2. 3MBIP architecture.

Normally, processors spend a lot of energy to search instructions on the memory, however in this design each instruction from the firmware is decoded into microinstructions inside the Microcode ROM which decrease the memory access. The addressing modes for the instructions are described below:

- \textit{imm}: Immediate argument of the instruction
- \textit{f}: Number of flag
- \textit{n}: Number of bits
- \textit{addb}: 8 bits address
- \textit{addw}: 16 bits address for Aw register
- \textit{addx}: 16 bits address for Ax register

These are used in 3MBIP instructions set, that consists of:

- Conditional instruction for addressing modes \textit{f} and \textit{addx}
- Comparison instruction for mode \textit{n} and an instruction combining compare jump
- Output instruction for all modes
- Store instruction for all modes
- Load instruction for modes \textit{addb} and \textit{addx}
- Increment and decrement instructions for modes \textit{addw} and \textit{addx}

Except the bitstream block all the remaining modules were designed with dedicated logic which allows high energy efficiency in the operations.

The texture decoding block is divided in 5 other blocks as shown in Figure 1 each one has its function summarized below:

- \textbf{DCTCD}: Performs the variable length decoding, reconstructing the DCT coefficients. This method is based on the fact that a typical coefficients block has a small number of non-zero coefficients if compared to the number of zeros of the same block.
- \textbf{IS}: Converts an one-dimensional array of DCT coefficients into a two-dimensional array of the same type, accordingly to the scan-pattern defined by the bitstream demultiplexer. This technique is very useful because in a block, the non-zero coefficients tend to be by the top-left corner, or around the low-frequencies. These significant coefficients can be grouped together by reordering the block using, for example, a zig-zag scan pattern [7]. There are other scan patterns that can be selected accordingly to the characteristics of each block [3]. To select the scan pattern, among other parameters it is necessary to use the prediction direction of the current block. This data can only be calculated after the dequantization, so it has been made necessary to feedback this information after the IQ (Inverse Quantization) block has performed its function, as shown in Figure 1.
- \textbf{ACDCIP}: Executes the inverse prediction function, in which the current DCT coefficients are updated based on the prediction information of the previously decoded neighbor macroblocks. More precisely, it is needed information about the DC coefficients of the previous dequantized blocks to calculate the prediction direction and the DC prediction value [3]. This was implemented introducing a feedback connection between the IQ and ACDCIP blocks as we can see in Figure 1. This was done in a way that the IQ block calculates the prediction direction and DC prediction value and sends it back, since this block holds all the information necessary to perform the DC prediction operation. To calculate the AC prediction values, besides the prediction direction, some of the previous predicted blocks are necessary. So all that is needed is a way to store data into the ACDCIP block.
- \textbf{IQ}: The IQ transmits the inverse quantized DCT coefficients to the IDCT. This step is mostly a multiplication by a scaling factor. In Terpsicore’s implementation the block responsible for the inverse quantization aggregates another function: to calculate the prediction direction and prediction value and send them back to the IS and ACDCIP blocks.
- \textbf{IDCT}: The inverse discrete cosine transform maps the DCT domain into the color-space domain.

The motion estimation involves two steps:

- \textbf{MVD}: Performs the motion vector decoding.
- \textbf{PBC}: The prediction blocks copier retrieves the previous decoded blocks stored in the VOP memory.

At last, the image composition is performed in two steps:

- \textbf{SUM}: Sums the output of the texture block and the motion estimation block and delivers the data in the correct order to be converted into RGB values. The data
sent from the SUM module are luminance/chrominance (YUV) coefficients. They are also written in the VOP memory to store the current frame, so that it can be used later as a reference frame in motion estimation. Only the luminance (Y) components are stored in the memory.

**RGB:** Converts from YUV luminance/chrominance color-space into RGB color-space. When this block receives a chrominance coefficient (U or V), it retrieves the associated luminance from the VOP memory and converts these parameters to RGB values. This was implemented this way to prevent RGB of storing luminance values again, since they are already in the VOP memory.

The Terpsicore design has been manufactured in AMS 0.35μm CMOS 4ML technology, the area occupied by the core and pads is 49mm² comprising 48095 gates and the power consumption measured is 115mW at 25MHz. It is able to decode simple profile level 0 with QCIF size at 15fps with a voltage supply of 3.3V.

### IV. Results

Although there are many implementations of the MPEG-4 standard, Terpsicore has come to be one of them that requires a smaller silicon area and lowest power if compared to other decoders. In the Table 1 it is possible to see this difference. Since different processes, supply voltages and video specification are used, the area and power were normalized according to the supply voltage, the frame rate, the cifsize and the dimension for the comparison. Equations 1 and 2 were used to calculate normalized power and normalized area, respectively [9].

\[
\text{Normalized Power} = \frac{\text{Power}}{\text{Process} \times \text{Voltage} \times \text{fps} \times \text{cifsize}}^{0.35}
\]

\[
\text{Normalized Area} = \frac{\text{Area}}{\text{Process} \times \text{cifsize}}^{0.35}
\]

The results show that if all these designs would have been fabricated in the same process technology and would work in the same frame size and frame rate, Tepsicore would occupy the least area and consume less power.

### V. Conclusion

The key point in this project is the architecture, where fully dedicated blocks do all texture decoding and motion decoding and also image composition, and on the top of it a dedicated processor only for demultiplex media streams. This approach leads to a more energy efficient decoder design.

### Table I. Comparison Between Terpsicore and Other Implementations

<table>
<thead>
<tr>
<th>Process (μm)</th>
<th>Matsushita’s decoder</th>
<th>Fujitsu’s decoder</th>
<th>Arakida’s decoder</th>
<th>Takahashi’s codec</th>
<th>Terpsicore</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.18</td>
<td>0.18</td>
<td>0.13</td>
<td>0.3</td>
<td>0.35</td>
<td></td>
</tr>
<tr>
<td>Voltage (V)</td>
<td>1.5</td>
<td>1.5</td>
<td>1.5</td>
<td>2.5 or 1.75</td>
<td>3.3</td>
</tr>
<tr>
<td>Area (mm²)</td>
<td>37</td>
<td>28</td>
<td>43</td>
<td>81</td>
<td>49</td>
</tr>
<tr>
<td>Video Spec.</td>
<td>QCIF 15fps</td>
<td>QCIF 15fps</td>
<td>CIF 15fps</td>
<td>QCIF 10fps</td>
<td>QCIF 15fps</td>
</tr>
<tr>
<td>CIF size</td>
<td>1</td>
<td>1</td>
<td>4</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>Power (mW)</td>
<td>11</td>
<td>9</td>
<td>160</td>
<td>60</td>
<td>115</td>
</tr>
<tr>
<td>Normalized Power</td>
<td>201</td>
<td>165</td>
<td>1403</td>
<td>213 or 436</td>
<td>115</td>
</tr>
<tr>
<td>Normalized Area</td>
<td>140</td>
<td>106</td>
<td>78</td>
<td>110</td>
<td>49</td>
</tr>
</tbody>
</table>

### References