POWER EFFICIENT SEMI-AUTOMATIC INSTRUCTION ENCODING FOR APPLICATION SPECIFIC INSTRUCTION SET PROCESSORS

Tilman Glökler and Stefan Bitterlich

Institute for Integrated Signal Processing Systems (ISS)
Aachen University of Technology
{gloekler, bitterli}@ert.rwth-aachen.de

ABSTRACT

A novel design methodology for the implementation of control units for application specific instruction set processors (ASIPS) is described. This methodology uses automatic instruction encoding and semi-automatic generation of the hardware instruction decoder to speed up the ASIP design. Significant power savings due to optimized instruction encoding are achieved. Results for ICORE (ISS-Core), which is an ASIP for digital video broadcasting algorithms of Infineon Technologies, demonstrate the efficiency and applicability of this approach.

1. INTRODUCTION

Application specific instruction set processors using ISA oriented internal structures combine several advantages compared to dedicated hardware. The most important advantage is the combination of flexibility and programmability of a processor with the scalability and computational performance of more dedicated hardware.

A drawback of this flexibility is an increase in energy consumption over dedicated hardware for the same computational task. Furthermore, the design time for ASIPs tends to be higher due to the fact, that the designer has to implement the hardware *and* the ASIP program.

This paper describes a methodology to speed up the design of the ASIP hardware using automatic instruction encoding and semi-automatic generation of the instruction decoder. In addition, it is possible to significantly reduce the power dissipation of the on-chip instruction memory with optimized encoding techniques.

1.1. Why ASIPs?

Programmability is useful for the implementation of highly configurable tasks like control flow or mixed data flow/control flow tasks. This class of applications is often prone to specification changes late in the design flow. A bug fix in the ASIP software decreases the design effort in these cases - even post-netlist-freeze or post-silicon mask changes at reduced costs are possible.

On the other hand, if the application requires high computational DSP performance for a subtask, the ASIP concept is able to provide this performance by increasing instruction parallelism or by using additional hardware resources. If the performance of an ISA oriented ASIP itself is still insufficient, the coupling of a dedicated coprocessor is easily possible.

The choice which kind of computational core is to be used for a specific design clearly is a trade-off between power-efficiency and flexibility in case of a semi-custom ASIP and a dedicated semi-custom core. The deployment of a full-custom DSP core (often available as a hard-macro block) may also be an option for power-critical applications with the drawback of a more heterogeneous, expensive and risky design flow and the loss of technology independence.

1.2. ASIP Control Power Minimization

This paper deals with ASIP power reduction using advanced instruction encoding schemes to reduce the power distribution in the ASIP instruction memory. Results for ICORE (ISS-core) [1] [2] clearly indicate that the instruction memory can have a significant contribution to the overall ASIP power as depicted in figure 1.

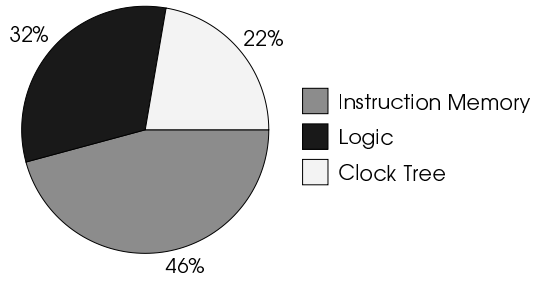

Figure 1: ICORE power distribution (unoptim.)

The following two options to reduce memory power consumption have been used in this work:

- reduction of the word width and the number of words

in the memory

- minimization of the toggle activity in the memory

The reduction of the memory word width is well known and has been exploited e. g. by the ARM thumb extension [7] and by many others reducing the size of the instruction ROM. This reduction is commonly referred to as “increase of the code density”, which means reduction in redundancy in instruction encoding. This concept has been used within our methodology to design an appropriate instruction set encoding with minimum instruction word width using the encoding techniques for fixed size instruction words described in chapter 3.

Furthermore, our methodology supports reduction of the internal switching activity in the memory itself. For this reason we have to use more detailed power models for the memories which are described in chapter 2. In chapter 3 the tool used for this optimization and the encoding techniques are described. Finally, the application of this design methodology together with quantitative results are presented in chapter 4.

2. MEMORY MODELS

The following discussion is limited to the read operation of simple read only memories (ROMs) or simple random access memories (RAMs), but similar models could also be used for the write operation of RAMs. The adaption to enhanced low power memories using divided-word-line or segmented bit-lines is also possible but not covered within this work. Furthermore, it is assumed that the main source of power consumption is the switching activity of the nodes and not the static leakage current (which is significant for very low Vdd-memories).

Memory models used for RTL and gate-level power tools like Synopsys’ DesignPower or Sente’s Wattwatcher use extremely simple power dissipation models. They only consider the logic activity on the primary I/Os and whether the memory is active (chips select asserted) or not. The internal logic activity is ignored by this simple approach.

The proposed optimizations in chapter 3 optimizes the internal memory power consumption. Thus, a more detailed model of the memory is required. Figure 2 shows a block diagram of a ROM with one transistor in the bit cell [3]: the row decoder uses one part of the memory address to activate one row of the ROM. This row is connected to bit cells - each of them contains one transistor in this case. If the gate contact of this transistor is connected (logic “0” - unconnected gates correspond to a logic “1”) to the word line, the corresponding bit line, which has been a priori precharged to logic high, is discharged by this transistor. Finally, using the remaining address bits, the correct set of bit lines is selected by the column decoder. Other technology options

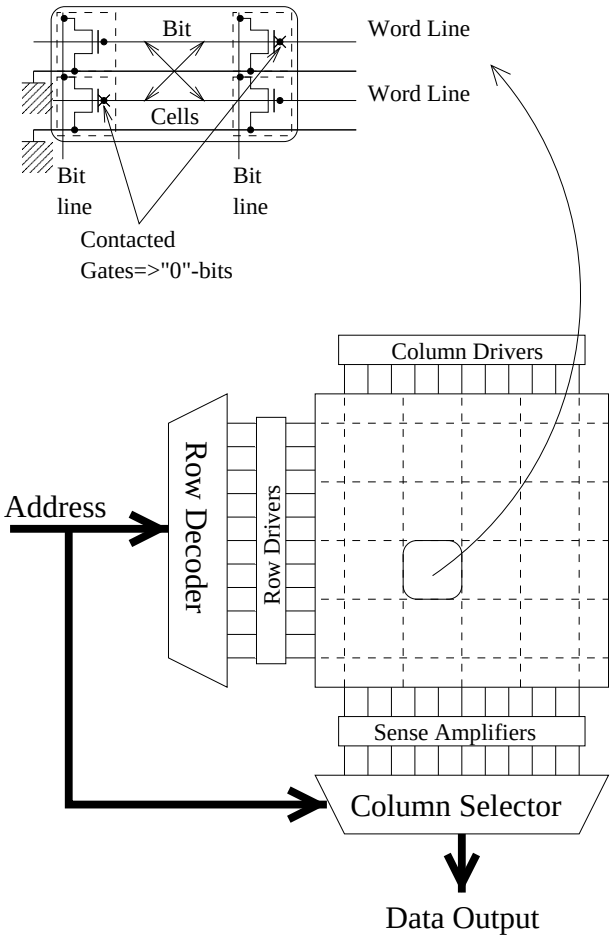

Figure 2: ROM Structure

are possible but the principle of this read operation stays the same. During read operation of a memory line, a “0” bit in memory requires one charging and one discharging action whereas a “1” bit requires no toggle activity on the bit lines. For large memories the memory matrix uses a lot of silicon area, thus, the bit and word lines and all of the connected nodes represent huge capacitances in the design. According to a case study by Chang [4], about 70% of the SRAM power is consumed by the bit lines, the sense amplifiers and the bit cells themselves. For ROMs these values might be slightly different, because they are using different cells. The proposed model for the current work assumes, that the remaining power (30% for the previously mentioned example) is not a function of the information stored in the memory and, thus, is not affected by the encoding optimizations.

The power depending on the activity of the word lines and bit lines is modelled using overall capacitances corresponding to the sum of all affected physical capacitances. For a bit line this overall capacitance C_{bit_line} can be sub-

divided into the capactitance of the bit line itself, the capacitance of the connected bit cell nodes and the input capacitance of the connected sense amplifier. The proposed model is depicted in figure 3. In this model the word lines control ideal switches which are able to discharge the bit lines in case of a “0” bit in the cell.

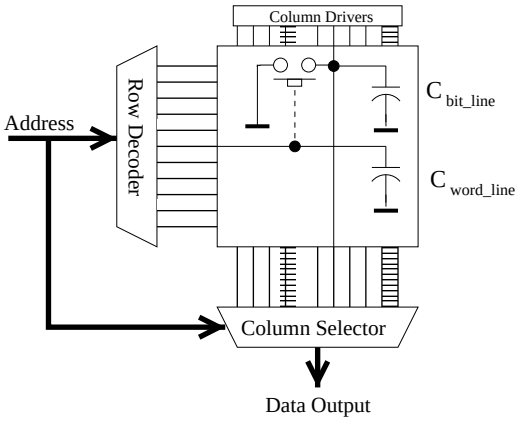

Figure 3: Model with internal capacitances

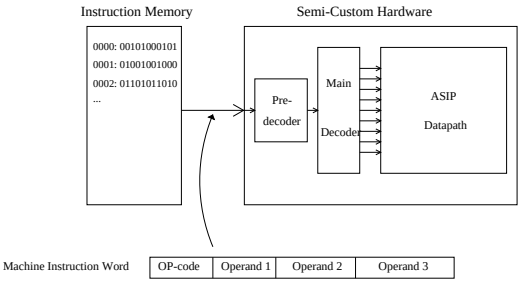

Figure 4: ASIP with Instruction Memory

The described model has been implemented to be used within Sysnopsys’ DesignPower. It supports arbitrary memory organizations in terms of $word\_lines \times bit\_lines$ with adjustable capacitances C_{bit_line} and C_{word_line}.

3. AUTOMATIC INSTRUCTION ENCODING

The proposed methodology includes a software tool, which generates an instruction coding table for the ASIP as a function of the instructions, that have been used in a given program. Afterwards, the information stored in the instruction memory and parts of the hardware decoder (namely the “predecoder” of figure 4) are generated.

3.1. Minimum Instruction Word Width

The instruction word of an ISA oriented ASIP can be subdivided into operation code field (op-code field) and operand fields as shown in figure 4.

In this project we used fixed length binary encoding and variable length hierarchical encoding for the op-code field. Hierarchical encoding can take advantage of shorter op-code fields in combination with long operand fields to minimize the overall instruction word width. On the other hand, there are cases, where the normal binary encoding yields better results than hierarchical encoding. Our automatic encoding program first analyzes the instruction set, which might be subject to frequent modifications during the ASIP definition. The optimum op-code field encoding style is then automaticly selected resulting in the shortest possible instruction word length.

The position of the operand fields in the instruction word is determined heuristicly reducing the hardware effort of demuxing the operand values. The challenge is to find an assignment, where operand fields of the same type (e. g. register fields) use the same position in the instruction word for different instruction formats, if this is possible.

Additional operand field encoding optimization are possible using histograms of the operand values from the programe trace files obtained using the fast LISA simulator [5]. For instance, if most of the relatively long immediate values use small non-negative values (binary encoded) with a lot of “0”es, it is advantageous to store these immediate values bit inversed.

An even more aggressive optimization method uses operand compression. Starting from a histogram of the used values of one operand in the program, the encoder program implements a hardware look up table for the used operand values only. If not all of the possible operand values are used in the program (which usually happens for large operand fields), it is possible to use a much shorter field in the instruction word. This methodology is very effective to reduce the width of the instruction word, but it reduces the flexibility of the design, because late design changes requiring the use of non-implemented values in the hardware look up table are harder to realize. Furthermore, the maximum size of the look up table is limited in hardware depending on the constraints of the design in terms of speed and area and the capabilities of the synthesis tool.

3.2. Reducing the Toggle Activity

As a function of the primary source of power dissipation the reduction of the bit line toggle activity seemed to be the best candidate for thorough optimization. By using as many “1” bits as possible in the memory the toggle activity on the bit lines is minimized, because the bit line does not have to be discharged after the precharge phase. This feature has been implemented in the automatic encoding tool for this work. For the binary fixed length op-code encoding, the tool assigns the code words as a function of their Hamming weight starting with the most frequently occuring instruction (maximum weight code word) to the least frequently occuring one (minimum weigth code word).

4. OPTIMIZATION RESULTS

The proposed methodology has been applied to ICORE, which is an ASIP implementing essential control and processing tasks of the next generation of INFINEON Technologies AG (Munich/Germany) single-chip DVB-T receiver. This chip integrates new features together with enhanced algorithms compared to the last generation receiver chip [6]. The implementation of ICORE has a typical load/store harvard DSP-architecture. The complete datapath excluding the 16x16-bit multiplier has 32 bits. The core implements 56 DSP instructions. These instructions are subdivided into 18 arithmetic instructions and 14 instructions for program flow control including zero overhead loop instructions. The remaining instructions are used for memory and interface I/O operations, bit manipulations and logical operations etc. Furthermore, some highly optimized instructions support a fast CORDIC angle calculation. The computational tasks of ICORE required about 1500 assembler instructions, thus, a 2k word instruction memory has been used, leaving enough capacity for further extensions.

Table 1 shows the toggle rates of the memory bit lines using an internal instruction memory organizations of 4 columns á 20 bit with 512 word lines and a typical ICORE program for DVB-T acquisition as benchmark. Three encoding techniques have been evaluated:

- conventional binary encoding with hierarchical opcode field
- the same encoding bit-inverted
- optimized encoding using the proposed instruction encoding tool with automatic operand field inversion, maximizing the "1" bits in memory

encoding technique	bit line toggles /1E6	savings (BL toggle count)
unopt. binary	2.93	0%
inverted binary	1.06	63,6%
optimized encoding	0.829	71,7%

Table 1: Results of Toggle Rate Optimization

It is obvious from table 1 that the conventional binary encoding yields a very high bit line toggle activity. This is due to the fact, that this encoding implicitly tends to prefer "0" bits instead of "1" bits e.g. for small non-negative operand values like register and immediate fields. Simple bit inversion already yields a significant improvement, but the explicitly optimized encoding is still better. The corresponding power savings are estimated assuming a similar (but for our approach more conservative) SRAM power distribution analogous to the one of Chang where about 60% of the overall power is due to the bit line toggle activity. This assumption yields a power reduction in the memory of approximatley 43%.

For the ICORE overall power consumption, this means a reduction of 19%. This is obviously not the bigger part of the overall power, but as this saving was achieved using a completely automatic design methodology without sacrificying any other design parameter, this saving is extremely valuable.

These results may vary using different memory organizations and different assumptions concerning the memory power. It may be argued, that the instruction decoder complexity in the ASIP could significantly increase and undo the savings in terms of power consumption using this approach. This is not the case in this example, as the ratio of the complete decoder power to the overall power is less than 2% for all encodings.

5. SUMMARY

A novel approach to the design of the decoder unit of ISA oriented ASIPs has been described. For this methodology, the decoder has been subdivided into predecoder connected to the instruction memory and main decoder, generating all the needed control signals for the datapath. The instruction encoding and the hardware predecoder were automaticly generated. Using this approach and appropriate instruction encoding techniques for ICORE - an ASIP for DVB-T algorithms - power savings of typically 20% in the instruction memory were achieved. The physical design of this DVB-T chip is currently performed at Infineon Technologies.

6. REFERENCES

[1] Tilman Glökler, S. Bitterlich, H. Meyr, ICORE: A Low-Power Application Specific Instruction Set Processor for DVB-T Acquisition and Tracking IEEE ASIC-SOC Conference, Washington, DC, September 2000.

[2] Tilman Glökler, S. Bitterlich, H. Meyr, Increasing the Power Efficiency of Application Specific Instruction Set Processors Using Datapath Optimization IEEE-SIPS, Lafayette, LA, October 2000.

[3] N. Weste, K. Eshraghian Principles of CMOS VLSI Design - A Systems Perspective, Addison Wesley

[4] Chang Power-Area Trade-Offs in Divided Word Line Memory Arrays Proc. of SLPED, August 1997.

[5] S. Pees, A. Hoffmann, V. Zivojnovic, and H. Meyr, LISA - Machine Description Language for Cycle-Accurate Models of Programmable DSP Architectures, 36th Design Automation Conference, New Orleans, June 1999.

[6] Product Brief SQC 6100 - Terrestrial Receiver for DVB-T, INFINEON AG, Germany, www.infineon.com /products/ics/pdf/sqc_10b.pdf, 1999

[7] http://www.arm.com/sitearchitek/armtech.ns4/html/Thumb, November 2000