Building Data Path for the Custom Instruction

Yong ZHU*

School of Computer Engineering, Jinling Institute of Technology, Hongjing Road Jiangning District of Nanjing No. 99, 211169

*Corresponding author

Keywords: Custom instruction, Data path, Function unit, SoC.

Abstract. Compared with a sequence execution programs of commonly used processor, the configurable processor uses custom instructions and builds user data paths for a speedup of tens to hundreds of times aimed at special and critical programs in the algorithm. The real-time processing and display IR SoC (infrared System on Chip) based on custom instruction is realized by designing user logic function unit and data path of pseudo-color thermal image processing.

Introduction

The implementation of image processing algorithm is mainly focused on pure hardware [1], FPGA [2] or DSP [3] mode. These ways are outstanding in performance, but less flexible. Or it’s easy to upgrade, but can not meet the real-time processing requirements of high resolution images. A configurable processor is used to extend the instruction set so that it can be implemented in a single instruction to a large number of modules requiring repeated computations, greatly improving the performance of the design [4]. There are good examples to use Altera Nios with configurable custom instructions to implement IR thermal image correction [5] [6].

In order to visually represent IR thermal image data, pseudo-color representation is usually used. The data value D is represented by the RGB color, namely F (R, G, B) = f (D, x, y), where x and y are two coordinates of thermal image. Since the color and the data do not have an objective one-to-one relation, it is a subjective representation. Moreover, a thermal image can correspond to a plurality of color palettes, and the color palette can be dynamically changed and adjusted depending on the display effect. The thermal image display provides the objective characteristic data of the object under test, and also reveals the visual temperature distribution information to the user to intuitively and comprehensively understands the object, which is helpful for field diagnosis.

The "custom instruction" technology commendably integrates the processor instruction function and its execution units on data path to accelerate the critical tasks, and develops the system speedup. The design is usually to construct user logic, which has the advantage of hard real-time processing of signals and integration with high-speed channels on SoC. With the development of SoC’s process fundamentals, design methods and verification simulation, microprocessors are integrated on many chips for complex system functions. In general, the processor executes microcode in a sequential and serial manner to implement a function; as for hardware circuit, all function modules operate at the same time under the driving of clock. It is a real parallel processing with higher execution speed.

Custom Instruction

Altera Nios is a cost-effective soft core. It is easy to add user-defined logic to the system by SoC technology. Nios also provides a custom instruction function, that’s easy to define complex functions as user instructions added to the Nios instruction set in a very user-friendly manner for application specific customization. It is a good method of development to quickly produce the final product and effectively shorten the time-to-market for prototype verification [7].

Systems designers can take advantage of Nios custom instruction sets to speed up the time-critical algorithms. The real-time software algorithm and time-consuming calculation software are added to
the Nios instruction set as custom instructions to improve the performance, and complete complex
data processing tasks and enhance the real-time processing ability of the system[8][9]. Custom
instructions may need only one clock cycle (combinational logic) or several clock cycles (sequential
logic) to optimize the software processing task, such as loop program, data packing, intensive
operations etc, and can access the external memory and user logic.

Implementation of Nios custom instruction includes two basic elements: (1) User custom logic
module, namely the hardware that implements the operation will become part of the Nios ALU; (2)
Software macro, That is, the system designer accesses user custom logic in the software code through
it.

The user custom logic maps the sampled data D to pseudo-color code RGB, and the pseudo-color
display is implemented by executing the custom instruction. Since the custom instructions are
implemented by the user hard-wired logic, the execution of the instructions can be guaranteed to have
hard real-time synchronization to the hard-wired logic pulse.

**Data Path**
The Nios ALU data path of pseudo-color display with user custom instructions is shown in Figure 1:

![Figure 1. The Nios ALU data path.](image)

The Altera SOPC Builder tool can be made use of to configure Nios with user custom instructions.
The key is to design the user logic. The connection and glue logic of bus system (on chip and off chip)
is completed by Altera Quartus II integrated environment.

There are two implementations according to the different data sources: (1) Normal operands A and
B provide source data, which are read into the user register of the Nios and supplied to the ALU; (2)
The operands are sampled and directly provide by user defined ports. This method is more close to
user logic interface and less operation overhead, but it needs synchronization mechanism to ensure the
correctness of data feed in.

**Function Unit**
User custom instructions are executed on the user logic unit of on-chip processor, which are accessed
by user instructions to perform special functions such as real-time processing. Data processing using
hard-wired logic is usually much faster than software operation, especially for HDL description
algorithm, such as XOR operation in CODEC, video synchronization timing and look-up palette. A
performance comparison of typical FPU (floating-point processing unit) between hardware and
software implementation is shown in Table 1:

|   | Hardware | Software |
|---|----------|----------|
| 1 | 1000      | 990      |
| 2 | 1001      | 991      |

With the growing scale of SoC, it is important to use proven and reusable IP modules as function
units for custom instruction execution. The IP and OEM from the professional manufacturer are
properly utilized in the design process, together with the special processing module of the HDL
algorithm designed by the user, so that the whole system has high performance and high reliability.
Table 1. Performance comparison table between HW. and SW. Implementation.

| FP. OP. | CPU Cycle | speed-up ratio |
|---------|-----------|----------------|
|         | SW. Lib.  | CI             |                |
| a*b     | 2874      | 19             | 151.3          |
| -(a+b)  | 3147      | 19             | 165.6          |
| |a|      | 1769           | 18             | 98.3           |
| -a      | 284       | 19             | 14.9           |

User logic can be described in HDL, and the basic logic signals include: Two 32 bit input operands dataa and datab, 32 bit results, clock signal, reset and start signal, and 11 bit prefix used for transmission, parameter and control purposes; The most critical is the user defined port, which allows the user logic to interface with the external components of the Nios system.

Implementation on SoC

The entire design process is completed in the Altera Quartus II: First, the project file is built, and then the 32 bit Nios is loaded with the SOPC Builder tool. In the Nios configuration page, the user logic is introduced, that is, the pseudo-color encoder HDL module. After that, the user logic module is scanned automatically and associated with the corresponding Nios custom instruction code, and the C/C++ and assembly language are generated in the header file to provide the calling macro. After compiling the entire engineering file, the programming file is generated and downloaded to the FPGA or its configuration chip. Details are as follows:

As a typical HW./ SW. Co-design system, the application software information is generated by SOPC Builder as follows: Software components, C and assembly header files (nios.h and nios.s), generic C drivers, middleware libraries used for software programming interfaces. The transmission and control of pseudo-color Avalon bus module with DMA are achieved through the macro provided by SOPC Builder as programming interface. That is, the custom instructions are executed by the macro ALT_CI_LUT call and implemented by user hard-wired logic. The code is as follows:

```c
IOWR_ALTERA_AVALON_ENABLE_REG(RGB_LUT_0_BASE,1);  //Timing start
alt_timestamp_start();
for(i=0;i<=255;i++)
{
    IOWR_ALTERA_AVALON_DATAIN_REG(RGB_LUT_0_BASE,i);  //Look-up table
    ci_result=IORD_ALTERA_AVALON_DATAIN_REG(RGB_LUT_0_BASE);  //Read results
    ALT_CI_FIFO_PORT(ci_result,FIFO_WRITE);      //Display output
}
timer=alt_timestamp();  //Timing finish
IOWR_ALTERA_AVALONPIO_DATA(LED_BASE, 0x0);
```

After the completion of software programming, the code is compiled into Nios executable machine code files through GNU compiler GCC for Nios in NiosII IDE, and downloaded to the user target board running.

The simulation results are shown in Figure 2:

(a)
Figure 2. (a) (b) The simulation of pseudo-color output in line.

NA_DATA and RGB are raw data and pseudo-color code respectively in Figure 2.

Test Result

The test results of CI (Custom Instruction) mode and I (normal Instruction) mode are shown in Table 2:

Table 2. Comparison table between instruction mode and Custom Instruction mode @120MHZ.

| Mode          | Look-Up Tab. (us) | Output (us) | Read (us) | Total (us) |
|---------------|-------------------|-------------|-----------|------------|
| CI@256 Pixels | 44                | 44          | 0         | 88         |
| I@256 Pixels  | 44                | 44          | 20        | 108        |
| CI@512 Pixels | 88                | 88          | 0         | 176        |
| I@256 Pixels  | 88                | 88          | 38        | 214        |

The test results of DMA mode are shown in Table 3:

Table 3. Test table of DMA @120MHZ.

| Pixels | Cycle | Time (us) | BandWidth (MB/S) |
|--------|-------|-----------|------------------|
| 128x8  | 4087  | 34.0      | 3.76             |
| 256x8  | 4336  | 36.1      | 7.09             |
| 512x8  | 4909  | 40.9      | 12.50            |

The CI pattern in form is similar to the processor instruction execution mode, but the essence is that it's executed on hardware supported by the user logic. It can be seen clearly that the running time is determined by user logic, and zero cycle overhead can be achieved from the test results. The key to CI technology is to build user functional units on data path. As a result, very high speedup is achieved when critical paths and time-consuming operations are performed in CI mode!

Conclusion

Pseudo-color image enhancement, as an effective method of thermal image visualization, plays an important role in IR camera detection and diagnosis [10]. The custom instructions embed the user's logical implementation into the Nios CPU as a special instructions that extends the CPU's ability to process signals in real time; The hard real-time processing and display of the image is realized by using the high performance transmission of Avalon bus and the hard-wired logic acceleration technology on the Avalon bus as I/O, which provides the guarantee for the on-line detection of the IR camera.

Since the implementation of the "custom instruction" and "Avalon module" technologies have hard real-time characteristics, it can be achieved immediate results by improving the user hard-wired logic and speeding up the clock frequency. As the amount of data increases, the proportion of DMA control overhead becomes smaller, and the transmission efficiency is improved. The pseudo-color SoC based on custom instruction technology has the advantages of fast transmission and high efficiency. It is easy to fuse with the user logic to protect the reuse of the original IP Core design.
Acknowledgement

This research was financially supported by the JIT fund No. 219/40620025.

Reference

[1] Raja G, Khan S, Mirza M J. VLSI architecture & implementation of H.264 integer transform microelectronics [C], 2005. ICM 2005. The 17th International Conference on 13-15 Dec.2005, Digital Object Identify, 218-223.

[2] Lindroth T, Avessta N Teuhola, et al. Complexity analysis of H.264 decoder for FPGA design [C]. Multimedia and Expo, 2006 IEEE International Conference on 9-12 July 2006, 1253-1256.

[3] Vun Nicholas, Nguyen T N A. Development of H.264 encoder for a DSP based embedded system [C]. Consumer Electronics, ISCE 2007. IEEE International Symposium on 20-23 June 2007, 1-4.

[4] Pozzi L, Atasu K, Ienne P. Exact and approximate algorithms for the extension of embedded processor instruction sets [J]. Computer Aided Design of Integrated Circuits and Systems, IEEE Transactions, 2006, 25(7): 1209-1229.

[5] Zhang Feng, Liu Shang-qian, Li Fan. Real-time Nonuniformity Correction for IRFPA Image Based on NIOSII [J]. Opto-Electronic Engineering. Vol. 35, No. 2, Feb, 2008: 71-74

[6] Kong Lingbin, Wang Chuan, Xu Hongwen, Qi Jianhan. Correction of infrared image with real-time and nonuniformity [J]. J. Huazhong Univ. of Sci. & Tech. (Nature Science Edition). Vol. 33. No. 12: 70-72.

[7] Fang Zhuo, Peng Cheng-lian, Chen Ze wen. NIO based SOPC design [J]. Computer Engineering and Design. Vol. 25 No. 4, Apr. 2004: 504-507.

[8] Lu Deliang, Zhou Xuegong, Peng Chengliang. Design and Implementation of Custom Instructions In Nios II Processor [J]. Computer Applications and Software. Vol. 24, No. 12: 3-9.

[9] Guan Lixin, Shen Baosuo. Custom FFT Instructions for Nios Embedded Processor [J]. Microcomputer Information, 2006, 22(32): 10-12.

[10] Zhu Yong. Design and Research on High-speed Channel of Infrared Image Based on SoC [D]. Huazhong University of Science and Technology. 2007.