# Communication and Interface Synthesis on a Rapid Prototyping Hardware/Software Codesign System

Yin-Tsung Hwang  Yuan-Hung Wang
Department of Electronic Engineering
National Yunlin University of Science & Technology
Touliu, Yunlin 64045, Taiwan, R.O.C
E-Mail: hwangyt@cad.el.yuntech.edu.tw

### *Abstract*

*In this paper, we propose the target board architecture of a rapid prototyping embedded system based on hardware software codesign. The target board contains a TMS320C30 DSP processor and up to four Xilinx XC4025E FPGAs. Various communication channels between the C30 and the FPGAs are provided and a master-master computing paradigm is supported. HW/SW communication protocols, ranging from handshaking, batch to queue controlled, as well as the corresponding interfaces are described in VHDL and C codes respectively and can be easily augmented to the mapped design. A codesign implementation example based on G.728 LD-CELP speech decoder shows the proposed communication protocols and interfaces lead to very small time and circuitry overhead.*

## 1. Introduction

HW/SW codesign has been a common practice adopted in embedded system designs for applications such as personal communication, automotive control and consumer electronics. Since various processing blocks in an embedded system must interact with one another, they cannot be designed independently. An important issue in HW/SW codesign is the synthesis of HW/SW communication interfaces. Efficient interface will reduce the communication overheads across the HW/SW boundary and thus enhance the entire system performance. In this paper, we first propose a rapid prototyping target board as the co-design platform. The target board architecture is flexible enough to accommodate the computing and communication needs for various embedded signal processing applications, in particular, in the signal compression areas. We next propose a communication interface synthesis scheme which maps various communication primitives into the target board.

Numerous work [1-8] on codesign system architectural templates and the associated communication interfaces have been presented. The architectural templates usually consist of a microprocessor for the software section, plus some programmable logic devices or co-processor for the hardware section. The communication between the HW and SW sections is achieved by shared memory, FIFO or handshaking protocols. They can support either master-master (where both sections can work concurrently) or master-slave (where only one of the two sections is active at a time) computing paradigm. Table 1 summarizes previous approaches as well as our proposed architecture.

| | Target architecture | Com pstyle | Host Interf | Batch com | HS com | FIFO com |
|---|---|---|---|---|---|---|
| [1] | generic template | - | - | - | Yes | Yes |
| [2] | i860+XC4008 | M-S | AT bus | Yes | No | No |
| [3] | Sparc+XC4025s | M-S | S bus | No | No | No |
| [4] | 68000+XC4005 | M-M | - | No | Yes | No |
| [5,6] | Sparc+co-proc | M-S | S bus | No | Yes | No |
| [7,8] | generic template | - | - | No | No | Yes |
| Ours | C30+XC4025s | M-M | PCI bus | Yes | Yes | Yes |

Table 1. Comparison of different target architectures

## 2. The target board architecture

As shown in Figure 1, our target board architecture contains 5 basic modules, i.e.

- a TI TMS320C30 *DSP processor* which implements as much as possible of the signal processing and control functions of the system;
- an *FPGA array* which is a programmable hardware accelerator consisting of four Xilinx XC4025Es to implement time critical functions of the system;
- a *peripheral block* which is for communicating with the data acquisition and playback devices;
- a 16M *shared memory module* which holds the program/data accessible to the DSP processor, the FPGA array and the host machine;

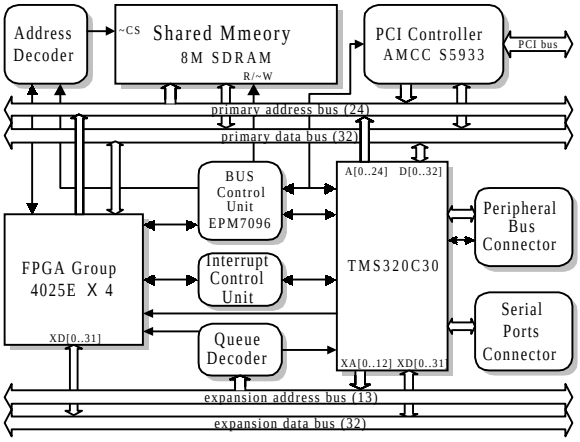


Fig 1. Block diagram of the target board

- a PCI bus based *host communication interface* through which the target board can be initialized and communicate data with the host machine.

To ensure the target board adaptable to a wide range of applications, a universal shared bus architecture is adopted. The 32-bit primary bus facilitates a basic shared memory communication model between the hardware and the software sections. The C30 processor is assigned higher priority than the FPGAs. The 16-bit expansion bus aims to provide a streamline port-to-port communication between the C30 and the FPGA array. With an addressing mechanism, multiple FIFO communication channels can be implemented over it. Inside the FPGA array, four 4025E FPGAs are connected as a ring structure as shown in Figure 2. This bus is employed for the point-to-point direct connection across FPGAs. Each FPGA is also associated with a 2K local memory module which can also be accessed by the C30's DMA controller

## 3. Communication protocols and interfaces

In our target board design, five types of communications are supported, they are:

1. *asynchronous communication by handshaking*: This is performed over the primary data bus. It is suitable for small amount of data exchange and for simple control transfer between the C30 and the FPGAs.
2. *asynchronous communication by queue*: This is performed over the expansion data bus. The queue itself and the access control are implemented in the FPGA hardware. This is suitable for in order, constant data rate communication.
3. *batch communication*: The batch communication is carried out by the C30's DAM controller to move a block of data between the shared memory and the local memory of an FPGA. It is thus suitable for burst mode data communication.
4. *synchronous communication*: It is supported only in intra-HW communication via the local interconnection bus between FPGAs. The send and receive signals are both latched. The scheme is suitable for pipelining the data path implementation across multiple FPGAs.
5. *direct communication*: This is similar to the case of synchronous communication except the signals are not latched.

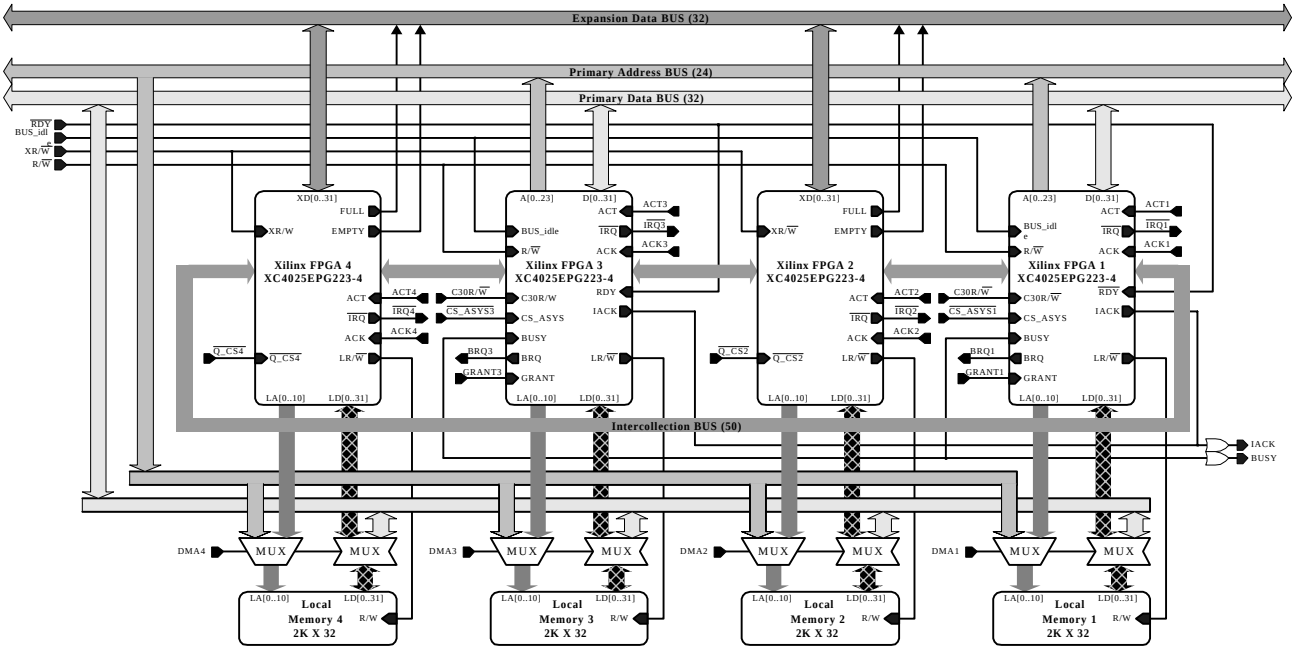


Fig 2. Block diagram of the FPGA array

Specific communication protocols must be observed for handshaking, queue and batch type communications. Basically, software protocols are coded in C routines while hardware protocols are described in synthesizable VHDL codes which contain both control FSM and interface circuitry. Table 2 outlines the features of these protocols. The readers are referred to [9] for the details. Figure 3 shows the FPGA implementation model which consists of a data path, communication interfaces (e.g. queues, shared memory access port, hardware send/receive ports), and their associated FSM controllers. For the data path FSM controller, the state diagram is shown in Figure 4. The original FSM of the data path controller is encapsulated in the *execution* state. Note that since these interfaces are implemented in FPGAs, they are included only when the corresponding communication schemes are employed.

| Comm types | protocols | Basic operations | Interface ckt |
|---|---|---|---|
| Hand-shaking | SW snd<br>SW rcv | Mem mapped I/O + polling | |
| | HW snd | interrupt | FSM+I/O port |
| | HW rcv | polling | FSM+I/O port |
| Queue control | SW read<br>SW write | Flag polling + mem mapped I/O | |
| | HW read<br>HW write | FSM control | FSM + queue |
| batch | SW init | SW snd + DMA | |
| | HW init | HW snd + DMA | FSM+I/O port |

Table 2. List of communication protocols

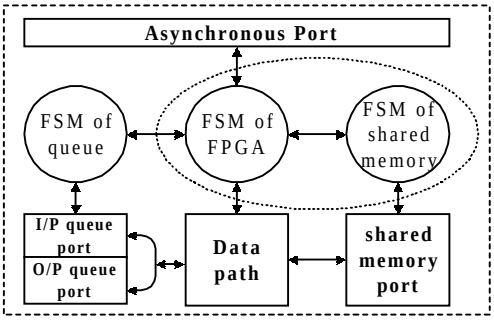


Fig 3. The FPGA implementation model

In Table 3, we list the estimated communication delays of the proposed communication protocols. The numbers are derived from the FPGA implementation and from counting the C30 execution cycles. (each cycle is 60 ns) They, however, do not include the wait delay incurred from the mismatch between the send and receive operations. Among all, the FPGA local memory access has the least communication overhead while the handshaking scheme takes the longest time. In Table 4, the FPGA interface circuitry overheads are compiled. The interface circuitry occupies only about 7% of the CLB resources.

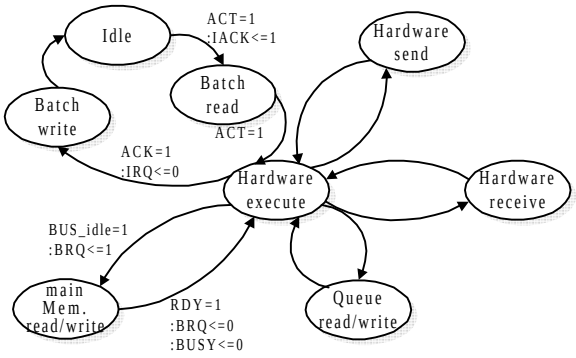


Fig 4. The state diagram of FPGA implementation model

| | Handshake | | C30 S. Mem | | FPGA S. Mem | | FPGA L. Mem | |
|---|---|---|---|---|---|---|---|---|
| type | H→S | S→H | read | Wr | read | write | read | write |
| cycle | 8 | 4 | 2 | 3 | 3 | 4 | 1 | 1 |
| | SW Queue | | HW Queue | | Batch | | | |
| type | read | write | read | write | | setup | delay/w | |
| cycle | 3 | 4 | 1 | 1 | | 4 | 2 | |

Table 3. The estimated communication delays

| | DP FSM | Queue /FSM | async port | s. mem port | local mem | avail -able | % used |
|---|---|---|---|---|---|---|---|
| CLBs | 10 | 24/8 | 32 | 0 | 2 | 1024 | 7.4 |
| I/Os | 0 | 32/0 | 0 | 32+24 | 32+11 | 192 | 52, 39 |

Table 4. FPGA interface circuitry overheads

## 4. LD-CELP decoder example and summary

To demonstrate the proposed communication protocols and interfaces, a large and practical codesign example for the LD-CELP speech decoder based on CCITT G.728 recommendation is investigated. Figure 5 shows the simplified block diagram of the speech decoder system. The system specification is described in VHDL as a collection of concurrent processes. To model the communication among processes, a process communication graph (PCG) is first constructed. The PCG of the LD-CELP speech decoder is shown in Figure 6. A system profiling is next conducted to extract each process's hardware and software implementation attributes. Each edge is also labeled as either collective or discrete mode communication. A PCG edge is then mapped to a particular communication subject to the

partitioning results and the rules given in Table 5.

| communication | Assignment conditions | | |
|---|---|---|---|
| batch | HW-SW | collective | |
| queue | HW-SW | discrete | invoc. freq > 1 |
| handshake | HW-SW | discrete | invoc. freq ≤ 1 |
| sync | intra-HW | | inter-iteration |
| direct | intra-HW | | intra-iteration |

Table 5. Assignment of communications

The total communication time overheads are compiled in Table 4. For both processes, the communication overheads consume less then 1% of the computing iteration time. The average delays for transferring one word of data are 180 ns and 127 ns, respectively. These figures are approximately only equal to the delays for the C30 processor to perform one off-chip memory access. The proposed communication protocols are therefore considered very efficient. Both the target board architecture and the LD-CELP speech decoder are currently still under development. Further refinements are expected before its real implementation. We have written the interface codes in both VHDL and C codes. For the LD-CELP decoder example, we have completed the entire system simulation and successfully implemented the Levinson-Durbin Recursion module on three XC4025E FPGAs. In summary, in this paper, we have presented a novel embedded prototyping system based on hardware/software co-design. The target board is carefully designed so that various communication protocols can be implemented efficiently with very little time and circuitry overhead. The communication interfaces are described in VHDL code and C communication routines and can be easily augmented to the HW and SW section designs, respectively. Our experiment with an LD-CELP speech decoder system fully exhibits the efficiency of the proposed communication protocols and interfaces.

## References


[1] M. Theiβinger, P. Stravers, and H. Veit, "Castle: An Interactive Environment for HW-SW Co-design. "*Proc. 3rd Int'l Workshop on HW / SW Codesign*, Sep. 1994, pp 203-9.

[2] M. D. Edwards, J. Forrest, " Software acceleration using programmable hardware devices.", *IEE Proc. Computer Digital Tech.*, Vol: 143, Jan. 1996, pp.55-63.

[3] G. Koch and others, "A prototyping environment for hardware/software codesign in the CORBA project," *Proc. 3rd Int'l Workshop HW / SW Codesign*, Apr 1994, pp 10-6.

[4] J. P. Calvez, D.Isidoro, and D. Jeuland, "A CoDesign experience with the MCSE Methodology." *Proc. 3rd Int'l Workshop on HW / SW Codesign*, Sept.1994, pp 140-147.

[5] R. Ernst, J. Henkel, T. Benner, "Hardware-Software Cosynthesis for Microcontrollers.", *IEEE Design & Test of Computers*, Vol. 10, Issue 4, Dec. 1993, pp. 64-75.

[6] D. Herrmann and others, "An Approach to the Adaptation of Estimated Cost Parameters in the COSYMA System," *Proc. 3rd Int'l Workshop HW/SW Codesign*, Apr 1994, pp 100-7

[7] R. K. Gupta, G. De Micheli, "Hardware-Software Cosynthesis for Digital System.", *IEEE Design & Test of Computer*, Vol. 10, Issue 3, Jan 1994, pp.29-41.

[8] R. K. Gupta, C. N. Coelho Jr., G. De Micheli, "Program Implementation Schemes for Hardware-Software Systems.", *Computer*, Jan. 1994, pp. 48-55.

[9] Y. Hwang, Y. Wang, and J. Hwang, "Rapid Prototyping of HW/SW Codesign for Embedded Signal Processing," *JISE*, vol. 14, No. 3, Sep. 1998, pp. 605-631.


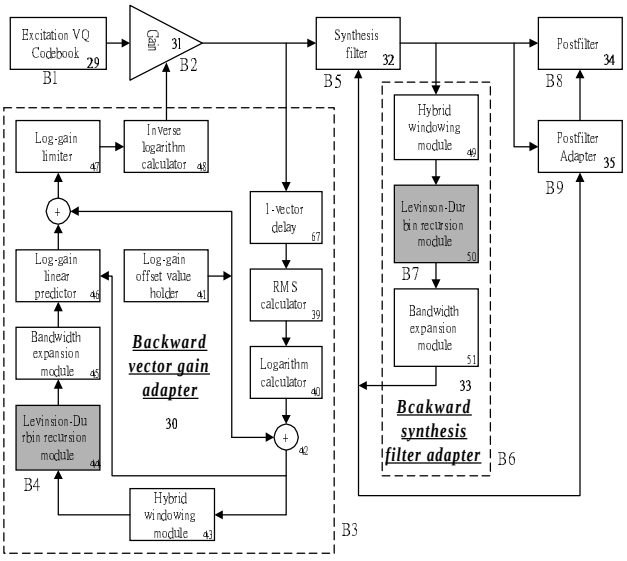


Fig 5. The block diagram of an LD-CELP speech decoder

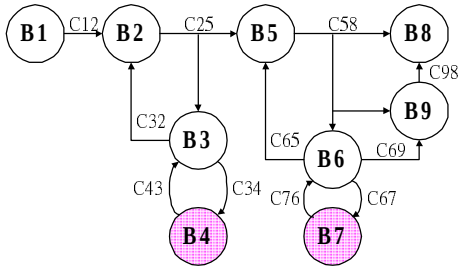


Fig 6. PCG of the LD-CELP speech decoder

| Proc | iteration cycle | assign type | # data comm. | Over-head | avg delay per word |
|---|---|---|---|---|---|
| B4 | 2.5ms | batch | 10 | 3.6us (0.14%) | 180ns |
| B7 | 2.5ms | batch | 50 | 12.7us (0.51%) | 127ns |

Table 6. Communication time overhead for the HW sections in LD-CELP decoder