Virtual Communication Bus with Hardware and Software Tasks in Real-Time System

Peter Nygren and Lennart Lindh
Mälardalens University, MRTC (Mälardalens Real Time Research Center), {peter.nygren, lennart.lindh}@mdh.se

Abstract:
The FPGA (Field Programmable Gate Array) of recent years has opened newer design possibilities of moving software into hardware. This paper is studied at two cases of transferring functionality from software into hardware. The paper describes the VCB (Virtual Communication Bus) concept and hardware tasks. The approach with VCB is focused today on existing systems with a main processor and slave processors or DSP (Digital Signal Processors). The first approach is to reduce the system load from the VCB bus and the second phase will be to eliminate the slave processors and move them to hardware tasks implemented in FPGA. This means that a hardware task can consist of 1000 pages of C code. The hardware tasks will reduce the response time and make the system more time-deterministic.

Keywords: FPGA, VCB, Task, hardware tasks, Real-Time System

I. INTRODUCTION

Already today an FPGA have 10 million gates it will not be long before FPGA's have hundreds of millions of logic gates on-chip [3]. A FPGA can be programmed by a subset of C[1] or a hardware language such as VHDL [2]. A rule of thumb is that about 100 pages of C code fit into 30 000 gates. The cost of a 30K device is to day under 10 US $ [3]. Compilers today translate ordinary software from code into serial machine code. The hardware compilers translate the source code into concurrent gates, fliplops and memories by means of a synthesizer. The synthesizer during the last 10 years has developed from a simple state machine to behavioral translation.
Today the designer can design in hardware design tools or ordinary software tools. With the help of new tools as CoWare N2C™ Design System [1] are dealing with this problem.

The first phase will be to eliminate the operating system (OS) load generated from the VCB bus. The system load generated from the bus is an ordinary software task and this functionality could be moved from software into hardware. This will reduce the system load and make more execution time available for application tasks.
The approach with VCB is focused today on an existing system (see figure 1) with a main processor and slave processors or DSP (Digital Signal Processors). The goal of this approach is to eliminate the slave processors and move the software functionality in to hardware tasks implemented in FPGA. This means that a hardware task can consist thousand pages of C code.

II. MOTIVATION AND OVERVIEW

The objective is to remove the functionality from the VCB bus, implement this functionality in software and move it into a hardware task. The purpose of this is to utilize the true parallelism in the hardware and their bye making available more execution time for the application tasks in the system. The second phase of the project will be to reduce the number of CPU's in a multiprocessor system with mixed architecture of generally micro-controllers and signal processors. The purpose of this is to decrease the response time of external units and to reduce the overhead for the ordinary system to be able to handle communication with external devices. Industry today can reduce the cost of system architecture design if the system architecture uses hardware task instead of CPU's. The cost of a system could thereby be reduced. An example of a commercial system is shown in *figure 1*. In this type of system it is possible to eliminate the slave CPU's and move the functionality into hardware tasks.

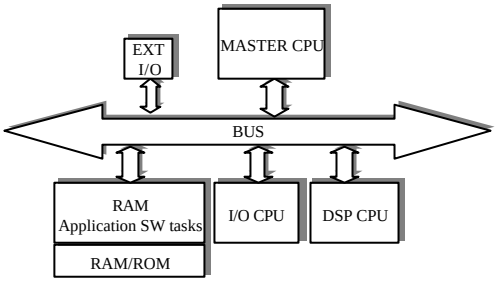

Figure 1: Logical architecture of a common system.

If we move the functionality from the CPU's and replace them with an FPGA and implement the functionality in hardware tasks could we reduce the necessity of mixed achitecture in many types of systems *(see figure 2)*. An advantage of using hardware instead of CPU's is that the external I/O can be connected directly into the FPGA. This reduces the communication on the shared bus. The communication possibilities on the VCB bus permit communication between hardware and software tasks in the system.

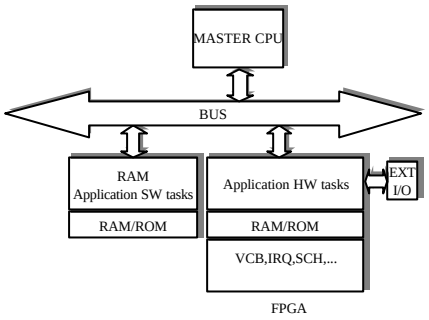

Figure:2 Logical architecture of the proposed system.

Moving common software functionality into hardware tasks results in greater predictability and increase system speed. This new design method gives less complexity and reduces the cost of the whole system.

III. PHASE ONE: VCB IMPLEMENTATION:

The Virtual Communication Bus is used for inter process communication and for synchronization tasks in the system. Communication between different tasks in a system usually consists of some kind of message passing mechanism such as mailboxes. The VCB bus is such a message passing mechanism and the bus allowed task to task communication locally on one CPU and between several different CPUs in the system. The tasks could be either an ordinary software task or it a hardware task, the interface for the tasks being the same.

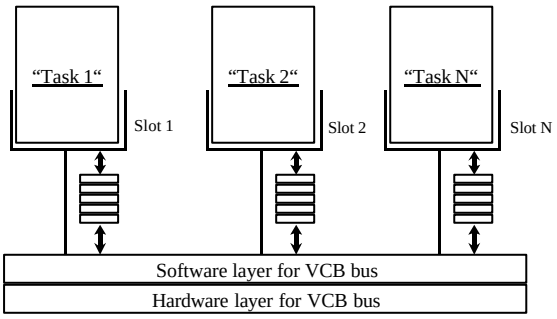

Figure 3 Logical architecture of a system with the VCB bus

The VCB bus is divided into two layers, the upper layer being the software implementation of the bus. In this layer the system provides support for different types of functionality from the bus. The other part of the bus consists of the base primitives. These primitives are implemented and integrated in the FPGA. When tasks want to communicate on the VCB the task hade to allocate one VCB-slot and their bye bee connected to the virtual bus. The VCB connects the system and makes possible communication in two different ways, *synchronous* or *asynchronous*. When a task is connected to a slot it can communicate with all the other tasks in the system.
Send and receive communication is the type of functionality mostly frequently used in the VCB bus. Other functions the bus could support are, for example *broadcast, send and wait, multicast, subscribe*. To reduce the system load from the OS (Operating System), the first step will be to reduce the execution time of the subscribe function. The subscribe function is a *Server<->Client* concept in which the *"Server"* is the functionality which handles all mail requests from other *"Clients"* in the system. The server running each time *T* to decrement each individual timer *T*. When the time has expired for one or more *"Clients"* the *"Server"* sends a mail to every *"Client"* in the request list (see *figure 3)*.

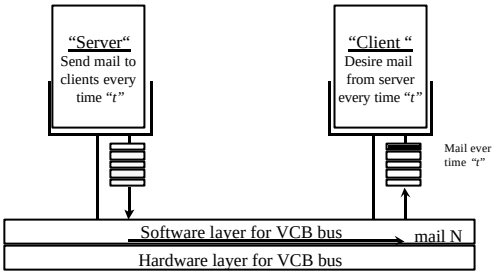

Figure: 4 "Server" "Client" concept for subscribe function

The mail system uses some shared memory area either globally or locally. In some cases the local location is a better solution because it minimizes the accesses on the global common system bus and improves the performance of the mail system.

IV. PHASE TWO HARDWARE TASK:

The purpose of removing CPU's from a system is to increase the performance and predictability of some functionality. This type of implementation gives many advantages. One of the biggest differences is the performance as a hardware task runs in true parallel mode giving higher speeds as compared with a corresponding software implementation. Another advantage is the predictability of the hardware, the *max/min* execution time for the function being definable on the clock cycle level. The complexity and size of the code are reduced when hardware tasks are used. The design space increased if the designer used hardware tasks instead of ordinary software implementation.
Hardware tasks can reduce the numbers of CPU's because of the possibility of using the VCB bus for communication and when functionality moved from the CPU into hardware tasks. The hardware task could be one instance or it could be divided into many small units in the same task.

When using CPU's in the system *(see figure:1)* it is difficult to handle the high frequency of the external I/O. The overhead for the interrupts reduces the CPU performance. The hardware task could handle the external I/O interrupts directly from the external I/O units without any overhead from the OS. The hardware task could also handle concurrent processing of the information in true parallelism. Results from the hardware tasks are sent directly to the software task at the master CPU via VCB bus.
Device drivers for handling external I/O arenot necessary

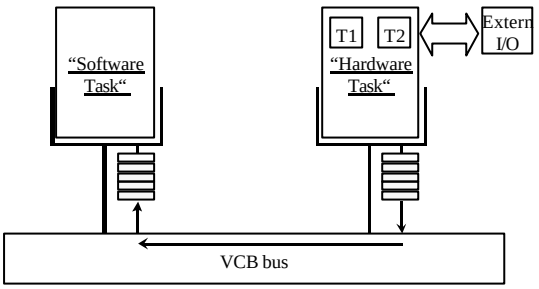

Figure 5: Logical picture of a hardware task

Higher capacity FPGA's give the possibility of using memory on chip instead of common system RAM. This gives a improved performance of the task in the FPGA.

V. ADVANTAGES WITH HARDWARE TASKS

The implementation of the same function code in hardware is considerable different from its implementation in software

It is easier to predict real-time behavior in hardware. Min and max times can be verified with tools. In software it is difficult or almost impossible. The background is that software uses the shared resources such as CPU, ALU but in the case of hardware the task use it's own gates. The hardware can use the same resource for different "tasks", but this can be scheduled offline.
In hardware it is also easier to deal with asynchronous events (such as interrupts). In a software solution an interrupt interrupts the entire system, in a hardware task it interrupts only the interrupt function in the system.

Performance of a hardware task is much higher than of a software task, if the function can be held inside the same chip. The parallelism in hardware is very high, in a 30000 gates FPGA it is 30000 concurrent elements.

Hardware tasks needs no overhead, such as operating system, device drivers for interrupt. The response time from a hardware task is much shorter than from a common software task.

FPGA hardware gives flexible hardware architecture. For example the number of ALU units, interrupt pins, I/O is configurable.

In a processor there are nearly always overhead and restriction resources. For example if you need an floating point unit, it will be provided, or if you need more interrupt lines there must be added. When you must access a unit outside the CPU the response time will increase.

VI. CONCLUSION

?? New hardware design methods [1] and higher capacity FPGA's create new possibilities of removing CPU's from ordinary system design and replacing the CPU's.
?? The integration of heavy regulates algorithms into hardware tasks and the possibility of transparent communication between soft and hardware tasks gives new dimensions in this new type of architecture.
?? New FPGA types with more internal memory need less external memory and will increase the speed of the function. In many cases the time behavior of hardware tasks will be much more predictable.
?? As the price of FPGA's is reduced by 50% each 18 month, the cost of architecture the same functionality in a system is almost the same.

VII. REFERENCE

[1] CoWare N2C Design System www.CoWare.com
[2] **VHSIC** **H**ardware **D**escription **L**anguage VHSIC stands for Very High Speed Integrated Circuit
[3] http://www.xilinx.com