DESIGN CONSIDERATIONS FOR REAL-TIME SYSTEMS WITH DSP AND RISC ARCHITECTURES

İskender Serhat KOÇ

Nortel NETAŞ
Ümraniye,İstanbul,Turkey
phone: + (90216)5222953 , fax: + (90216) 5222222, email: iskoc@netas.com.tr
web: www.netas.com.tr

ABSTRACT

The challenges a real-time embedded designer faces in a new product design can be summarized as : Time-to-Market, Real-Time Performance, Quality, Cost, Scalability and Development Tools. This paper discusses Real-Time Performance challenges and presents some practical and important techniques used in Nortel Netaş products to overcome these challenges. The focus will be on DSP and RISC architectures.

This paper is organized as follows :

- Real-Time System Definition
- Customer and Technology Requirements
- Choosing The Right Architecture For The Solution
- Design Considerations For DSP and RISC
- Applications In Nortel Netaş Products

1. REAL-TIME SYSTEM DEFINITION

When we say real-time system, in fact we are emphasizing a system with deadlines, some of which are critical while some are really vital. So the main and very simple difference between a real-time system and a non-real-time system can be phrased as "The correctness of the result of a computation is also a function of time in real-time systems", where the same is not always necessary for non-real time systems. In other words, the correct result of a computation at wrong time is accepted as incorrect or useless for a real-time system. An example : If the signal sent by the park sensor of a car is processed later then required, then the car will crash the wall. No matter how accurate the sensor information in meters is, it is useless.

2. CUSTOMER AND TECHNOLOGY REQUIREMENTS

The first step for a designer is to clearly understand and determine the requirements of the customer. Also there may be missing requirements which customer can forget or can not declare. These must be informed. After clarifying all the requirements, then the technology for the solution should be decided. Sometimes the customer can dictate the technology like, "ATM in the backbone or ADSL at the subscriber line" etc. When the customer does not announce the technology, the designer should do it instead. And at last, the designer should also consider possible potential features the customer most probably will request later in the future. The design should be able to meet some of these future requests, at least without requiring any change in the existing H/W.

Now we can cover all the requested features and some potential future features. It is time to choose the right architecture.

3. CHOOSING THE RIGHT ARHITECTURE FOR THE SOLUTION

Today, there are different alternatives for the embedded designer to choose from. The most popular ones are : ASIC, FPGA, DSP, RISC, CISC and Network Processors. All have advantages and disadvantages depending on the solution. We will not discuss those in detail in this paper. But in summary, the designer must take the facts listed below, into consideration :

Common to all types :

- Cost / Performance
- Time-to-market : Instant availability or in the roadmap
- Scalability

ASIC and FPGA :

- Size
- Internal constant and variable delays
- Internal usable blocks : ram, timer, PLL

For DSP, CISC, RISC :

- Maximum addressable Program and Data Memory size.
- Memory Management : Memory Protection & Translation.
- Program and Data Cache availability and the size.
- Speed : Operating frequencies.
- Architecture : Multi core, Super Scalar, Pipeline Stage, Floating Point Unit etc.
- Internal Memory : Zero wait cycle ram accesses , size
- Exception Reporting : Very important for bug tracing
- Interrupt controller : Interrupt latching, priority resolving, interrupt nesting.
- DMA : Efficient data move between peripherals and/or memory.
- Timer : For generating periodic interrupts & events and precise measurements of critical code.
- External I/F : Bus width, bus clock.
- External I/F : Supported devices(SRAM, SDRAM, SBSRAM, DPRAM)
- External I/F : Additional wait state insertion, external ready logic for slow devices and multi master shared ram configurations.
- Internal units like ATM, Ethernet, HDLC, DES Encryption, I2C, SPI, UART etc.

Determining the right architecture is not an easy task actually. For that, simulators or more costly but more realistic, an EVM(Evaluation Module) will probably help a lot. If it is seen that the processor suffers, then the architecture can be changed to a dif-

ferent or faster version immediately or it is totally changed. The earlier the deficiency of the architecture is seen, the better. After a moment in the design phase, it will most probably be impossible to change the processor or architecture.

4. DESIGN CONSIDERATIONS FOR DSP AND RISC

During the phase of architecture selection, the hardware design of the card, backplane or maybe the whole system must also be taken into consideration. Surely, just considering the processor only is not enough for the selection. All the interfaces, communication paths with other cards and processors, function specific chipsets on the card, H/W limitations, bottlenecks in the system and specific requirements of the H/W must also be taken into consideration. To make it clear, let's give an example: We design a board having 400MIPS DSP or RISC processor and a slow device of 1µs access time. Assume that, this slow device will be accessed every 5µs which ends up with 20% real time performance spending for this process. And if this periodic process is not the main process of the processor, then we must conclude that, we have a 320MIPS processor for the main process instead not 400MIPS as we selected.

The considerations during design and implementation phases are presented below.

4.1 Processor Data Bus

Unless otherwise needed, use the full width of the processor data bus and connect the fastest RAM chips the processor interface supports. Sometimes, power consumption, size and layout of the board, reflection and crosstalk issues brings some limitations for these interfaces. This is an important decision for the design which directly affects performance. So it must be strictly decided in the beginning phase. If this can not be determined exactly, then the real-time assumptions must be done for the worst case.

4.2 Shared / Multi Ported RAM

If there are more than one processor on the card, their access to shared resources must be minimized. RAM is one of the most common resources between processors in multi processor designs. RAM is frequently used for data flow between the processors. The RAM between these processors can be designed in two ways : *Shared Ram or Dual Ported Ram*.

If shared ram design is used, most probably an arbitration logic should be included to control the accesses of processors. One will access the shared ram while the others will be frozen by external READY delayed cycles. So, the other processors are hang during the winner accesses the shared ram. DPRAM or Multi Ported RAM is better, but a little bit expensive way to overcome this problem. The arbitration logic is embedded in the RAM device. Unless the same location in the DPRAM is accessed, no requester's cycle is delayed. So, both sides access DPRAM in parallel without wait states.

4.3 Internal RAM usage

Most of the DSPs today have internal memory available. But the size of this memory is not sufficient for most of the applications. So the most critical codes, like interrupt routines or signal processing algorithms must be located in internal RAM region while less critical code like configuration, management or performance monitoring can be located in external RAM regions.

4.4 H/W interrupts

H/W interrupts are presented to the processor by external signals. Some processors have internal units like timers or DMAs which also generate signals to the core internally. When designing with H/W interrupts, the following details must be considered :

- How frequent will this signal be generated.
- Should this signal be edge or level sensed.
- If an interrupt is missed somehow, how will this be recovered.
- What should be the maximum tolerable interrupt latency.
- What should be the maximum interrupt service routine execution time.
- Every interrupt means context save & restore overhead. According to the architecture, the size of the saved context(registers, MMU tables) differs.
- Can this overhead be reduced by grouping the signal of this interrupt by other interrupt signals.
- Is the interrupt code as small as it can be locked to cache when extra performance becomes vital.
- Is there an interrupt controller for priority resolving or will it be implemented in S/W.
- Is it needed to nest interrupts according to priority.
- Is it needed for the interrupt to be able to preempt itself. This is rarely needed and applied for catching very critical interrupts.
- Is NMI needed. NMI can be very useful to signal just before the H/W generates a hard reset caused by watchdog period expiration.

4.5 Multi Bus Processors

Some DSPs and RISC processors have more than one bus for accessing memory and peripheral devices. The processors today, mainly the ones called Communication Processors mostly have more than one execution unit that can access external memory or peripherals. These processors can have more than one core in the same package, an additional RISC responsible for implementing communication protocols like ATM, Ethernet, TDM and additionally a powerful multi channel DMA engine. These units can mostly operate parallel unless a resource conflict occurs. So they can request access to external memory at the same time. In order to utilize best performance of multi bus structures, the connected devices to the busses should be organized carefully. As an example, a very slow device should not be connected to a bus with high speed SDRAMs. Otherwise, for example, when the DMA transfers data from/to this slow device, the core will be blocked. So this will not be a good practice. Better way is to connect it to the other bus and organize the other peripherals around the same philosophy.

Also the data structures should be organized in such a way, as to minimize resource conflict between multi units in the processor. An example : If you have a processor with an ATM controller inside, then locate the ATM cell buffers to the bus other than the bus the program and data are located at. By this way high rate ATM traffic will not stall core execution.

4.6 Slow Device Interface

Although the technology today brings GHz speeds into the embedded market, still there are and will always be slow devices like external UARTs or Flash EPROM and we will always be obliged to access them. So if it is possible, we should design in such a way as to pipeline the operations. As an example, when the processor wants to program flash EPROM according to EPROM's FSM(Finite State Machine), it is not a good design to wait for the EPROM to acknowledge the operation. It is better to do other jobs during this

process and then check it periodically or be invoked by an interrupt generated based on EPROM condition signals. (moderate flash word programming is about 100µs, flash erase is about 2-5s which is a deadly period for 500Mhz processor having 2ns cycle time).

4.7 Maximum Cache Utilization

Most of the processors today include internal program and data cache of approximately 16-32K each. Some implement direct mapped caches while others implement 2 or 4 way set associative caches. And almost all of them implement LRU(Least Recent Used) algorithm for aging cache lines. The theoretical best practice is to lock all the code to program cache and all the data to data cache and operate at the clock speed. But unfortunately, caches are too small for the whole code and data to fit in. So what a software designer must do, is to minimize the cache misses as much as possible. The caches operate using two main principles : *spatial locality & temporal locality.*

Spatial locality means, if a memory location is accessed, then most probably a neighboring location will also be accessed. So it is wise to cache these extra locations also. This is accomplished by caching the whole line size of 32 or 64 bytes. Temporal locality means, if a memory location is accessed once, then it is most probably that this memory location and its neighboring will again be accessed in a period of time, which leads to LRU aging algorithm.

These principles in mind, some methods for cache utilization are listed below :

- Try to use arrays instead of separate variables for utilizing spatial locality. Separate variables can easily be mapped to different memory locations for alignment and performance criterions by today's highly optimized compilers.
- Try to use loops as much as possible for temporal locality.
- If appropriate, disable interrupts around loops. Otherwise when the interrupt occurs, interrupt service routine code will be cached and loop code will be invalidated.
- If there are function calls from the loops, this will most probably invalidate the loop code. So try to divide loops into smaller loops. Accumulate results in a stack array and call function once with this array pointer. Organize the called function also as a loop, executing on the array passed as function parameter.
- Dividing loops also helps. Because the smaller the loop is, the more possibility it will fit into cache and achieve high cache hit rate.

4.8 Maximum DMA Utilization

Most of the processors have DMA engines inside as separate units. Some has one to six independent channels while some others can have 64 independent DMA channels. Try to utilize as much as possible from these engines. DMA is very useful for three reasons :

- Moving bulk of data between memory or peripheral in the background. (Highly utilized in Nortel Netaş products)
- Read/Write data to/from peripheral on synchronous or asynchronous events in the background while the core is busy with other jobs. (Highly utilized in Nortel Netaş products)
- Highly complex DMA engines can organize data in different formats which is very useful for serving multi channel TDMs for speech codecs.

4.9 Maximum Register Utilization

When writing in 'C', try to use stack variables instead of accessing directly to variables or structures in RAM. This way the RAM variables are read only once and then referenced as stack variables, so no more memory accesses. Most of the execution will go through registers. The compilers today try to allocate core architecture registers for stack variables as much as possible. If there is lack of registers, then actual stack memory is allocated. Although some of the compilers today are so intelligent that they can manage this, sometimes a little help to the compiler is will be helpful.

5. APPLICATIONS IN NORTEL NETAŞ PRODUCTS

The design considerations and techniques described above are highly used in products we design. Below, some of our products are given as examples, emphasizing these techniques.

5.1 ATM Cell Mapping to 64Kbit/s TDM :

We designed an APCO-25 compatible switch, which implements the RFC(Radio Frequency Controller) and RFG(Radio Frequency Gateway) functional blocks of the standard definitions. Switch interfaces to the base stations with 2Mbit/s ATM E1 links. And over these links, ATM cells carry 4.4Kbit/s IMBE(Improved Multi-Band Excitation) coded speech parameters, encryption parameters and some other radio channel and base station related parameters. The switch should transport these ATM cells to a black-box interfacing with BRI(2B+D). As the PCM is synchronous and the ATM is asynchronous, we had to map these ATM cells to TDM using X.30 framing. We designed a card interfacing to the ATM Bus(155Mbit UTOPIA-II) and the PCM Bus(2Mbit/s TDM) of the switch backplane. Motorola MPC8260-200/166/66 is used as the host processor and 5xTMS320vc5416-160 are used as the X.30 mappers. One DSP is capable of mapping 8 channels. ATM controller of the 8260 is interfaced to the ATM Bus of the switch. DSP side and the host side are interfaced to each other through high speed dual ported RAMs. The ATM traffic coming into 8260 is routed to DPRAMs. The DSPs read the cells from DPRAMs and map them to X.30 framing. We used the same DMA technique described in section 5.4 . The period of mapping an ATM cell to X.30(10byte frame) is 11ms. We did not used 11msec samples for DMA but 3.750ms samples. This decision was a result of maximum CDV(Cell Delay Variation) requirements.

5.2 Radio Signal Receiver Voter – Comparator

The APCO-25 Switch referenced in 5.1 is also responsible for Inter-BS handover. The radio speech signals received are encapsulated in ATM cells and sent to the switch. According to the cellular coverage principles, there are receivers operating at the same frequency. As a result of this principle and implementation, the switch receives replicas of ATM cells from different BSs, carrying the same speech samples. Makes a decision based on RSSI(Received Signal Strength Indicator) value which is measured and inserted into the cell at the receiver, then clones and distributes these replicas to the transmitters of the same channel back. We designed a card having very powerful DSPs dedicated to this job. We selected Texas TMS320c6416-500Mhz which has UTOPIA-II direct interface to our ATM Bus. One DSP is responsible for 160 channel receiver voting and distribution.

The challenges according to system design were :

- Around 6Mbit/s ingress & egress traffic to/from DSP should be handled at ATM layer.
- Selected cell should be cloned and modified according to destinations.
- 160 channel processing with CDV constraints.

Our solutions were :

- We did not want UTOPIA peripheral send interrupt to the core for every cell received or cell transmitted. The core

would already be busy with 160 receiver voting channels, trying to minimize ATM traffic reshaping and optimizing CDV. So we had to find another way for driving UTOPIA cell traffic. We utilized 6416 DMA. 6416 has a very efficient and well designed DMA engine, which Texas call it EDMA(Enhanced DMA). EDMA executes 64 independent channels at the half clock rate of the core clock, in our case 250Mhz. And these channels run over a dedicated parameter structure located in the parameter RAM, which can be modified on the fly by both the core and the EDMA itself. We utilized this ability and designed an EDMA micro engine consisted of 7 EDMA channels for receiver and 7 EDMA channels for transmitter. What upper layer S/W does is, just to supply ATM cell buffers to a pointer queue and poll the status from a status queue. When the UTOPIA peripheral generates an event of a cell arrival, this micro engine invokes, reads the current ATM cell buffer pointer from the queue and moves the cell to this buffer and at the end signs the status queue for a new cell arrival. So the upper layer can be aware of a cell arrival. If the queue is full, the cell is automatically discarded by this micro engine. The TX side also runs similarly.

- We decided that, bulk cell copy is a burden for the core. So we again designed EDMA micro engines, which uses a similar queue for destination cell buffer pointers and source cell buffer pointer. The upper layer supplies the destination pointers and source pointer, then invokes the micro cell copy engine, then execute other jobs. When the copy finishes the EDMA interrupt the core.
- We fitted all the code and data into 1Mbyte internal memory and used all the optimization techniques described in this paper. We also utilized an optimization scheme which the DSP architecture provides; Texas calls it "Software Pipelining".

5.3 Tactical ISDN Terminal DTE V.24 & X.21 I/F :

We designed a Tactical ISDN Terminal with a DTE I/F for V.24 and X.21. The I/F supports rates of 1.2 to 64Kbit/s for synchronous and 1.2 to 38.4Kbit/s for asynchronous. The control line activities are sensed by a CPLD which signals the host CPU : Intel 80186. The data lines RX/TX are connected to the McBSP(Multi Channel Buffered Serial Port) of TMS320c5416-120 and the McBSP is serviced by the DMA for both directions. The data received from the DTE side is mapped into X.30 frames to be carried through 2B+D ISDN I/F to the network.

The challenges according to system design were :

- According to V.24 standard for asynchronous communication, the DTEs communicating through the network may have a rate difference up to 25% of the baud rate. This must be tolerated by adjusting the stop bit size by the same amount.

Our solution was :

- We sampled the line at a rate of 16 times faster than the actual transfer(baud) rate. So the bit value ZERO and ONE are sampled as "0000000000000000" and "1111111111111111" respectively. If a need for stop bit adjustment arises, the number of the ONEs is decreased according to the difference. Ex: if 12,5% rate difference is needed 14 ONEs is pumped to McBSP for stop bit.

5.4 Tactical PBX Voice Mail Feature

We designed a card for our Tactical-PBX for record and playback voice mail. This card should also be compatible with cost, reliability and environmental condition requirements dictated by the specifications. The speech signal is encoded by G.723.1(6.4Kbit/s) and recorded to flash EPROM. During playback, G.723.1 parameters are read from flash EPROM, decoded, converted to A-Law and pumped to PCM. Texas TMS320vc5416-120 DSP and Intel 16Mbit flashes are used.

The challenges according to system design were :

- DSP must handle 4 channel encode and 8 channel decode simultaneously.
- 4 channel encoded data must be written to flash which is partitioned as 1Kbyte sectors. 8 channel decode data must be read from these flash sectors. And all these should be handled simultaneously.
- During the record & playback operations ongoing, record deletion must be handled simultaneously.
- Deleted flash sectors can not be written again, unless the block including the sector is erased. So a defragmentation task is running in the background moving sectors to other blocks, organizing records and erasing flash blocks.

Our solutions were :

- We used two DMA channels of TMS320vc5416, one for transmit(8 decode) and one for receive(4 encode) as 240 sample pumper and 240 sample collector respectively, from/to McBSP. By this way, the core is interrupted once only every 30msec. Otherwise, every 125µs the core should handle 12 interrupts, more critically, as the length of a timeslot in 2.048Mbit TDM is 3.9µs, our deadline would also be 3.9µs.
- Programming flash is around 100µs per word and read is around 200ns. The core should not wait flash for 100µs. We designed a flash driver layer executing a FSM which orders, preempting and reports the program/read/erase requests from upper layers. When the request is completed the requester is informed.
- The flash driver is designed in such a way that, a fast read operation preempts a long programming or erase operation using flash EPROM commands like SUSPEND.
- We divided flash blocks of 64K each to 1K sectors, which can store approx. 1s coded speech parameters. But when a record is deleted, the sectors are signed as deleted*(for state modification in flash sectors, we utilized the rule "flash programming is in fact zeroing the ones".Ex : full_state = 0xFFFF, deleted_ state = 0xFFFE, defrag state = 0xFFFC. Zeroing the rightmost bits in order)* and these sectors can not be programmed again. So we designed a defragmentation background task, which continuously checks the deleted sectors in a block and if the percentage of the deleted sectors of the block exceeds the threshold, move used sectors to the spare block, erase the moved block and make it the new spare block.

REFERENCES

[1]Texas TMS320C6416 Data Sheet July 2004
[2]Texas TMS320C64xx EDMA Architecture March 2004
[3]Texas TMS320C64xx EDMA Performance Data March 2004
[4]Texas TMS320C6000 CPU and Instruction Set Ref. Guide Oct. 2000
[5]Texas TMS320C6000 Programmer's Guide August 2002
[6]Texas TMS320C6000 Optimizing Compiler User's Guide October 2002
[7]Texas TMS320C6000 EDMA IO Scheduling and Performance Mar 2003
[8]Texas TMS320C6000 DSP EDMA Controller Ref. Guide July 2003.
[9]Texas TMS320C54x DSP Ref. Set Vol 5 : Enhanced Peripherals 1999
[10]Motorola MPC8260 PowerQUICC II Family Reference Manual 2003