

Application of Texas Instruments' TMS320C6400 in 3G Wireless Infrastructure Transceivers

Jelena Nikolic-Popovic
Technical Staff, DSP Applications
Texas Instruments Canada
Email: j-nikolic@ti.com

Abstract. *Third generation wireless transceivers require more programmability than their second generation predecessors. In this paper, we explore how Texas Instruments' TMS320C6400 family of DSPs could provide this programmability, specifically for symbol-rate functionality. Details of the implementation and real-time performance are given for each function. A summary of real-time requirements shows that it is possible to implement close to 400 voice channels or 4 384kbps data channels on a single 800MHz device.*

I. Introduction

While 2nd generation CDMA (IS-95) basestation transceivers are, in general, based on a hardwired (ASIC) solution with limited programmability, the system-level requirements for 3rd generation BTSs are such that a higher level of programmability is necessary. Some of the reasons are as follows:

- The ITU has adopted a “family of standards” approach, rather than adopting a single standard. Various standards in the family are similar, but can probably not be implemented cost efficiently using a single hard wired solution;
- Some algorithms are still not understood well enough to be implemented in hardware (turbo decoding, multi-user detection, smart antennas);
- Due to a wide range of data rates which are to be supported (from 8kbps to 2Mbps), some functions need to be implemented for a large number of parameter sets. For example, it will not be feasible to pre-generate turbo interleaver or rate matching look-up tables for all possible user data rates;
- DSP architectures are reaching performance levels at which DSP devices become comparable in power consumption and cost efficiency to ASIC solutions, especially for infrastructure applications that tend to be lower volume than handset applications.

The discussion in this paper will be limited to a 3G basestation transceiver function, but similar arguments could be made for 2.5G transceivers (EDGE, UWC, etc), as well as 2.5G and 3G transcoders and multimedia processing equipment (MPE).

II. High Level System Partitioning Approach

At a very high level, basic functionality of a 3G BTS can be described as shown in Figure 1. A suggested DSP/hardware partitioning is also shown.

User data arrives at the basestation transceiver via circuit-switched or packet-switched backbone network. The data is split into frames and passed first through symbol-rate processing (FEC encoding, interleaving), then spread and modulated, and sent out to RF and antennas.

Air interface signals arrive through RF from antennas. Finger search, and rake receiver functions are performed (including correlation and despreading), and symbol-rate data is then deinterleaved and FEC decoded.

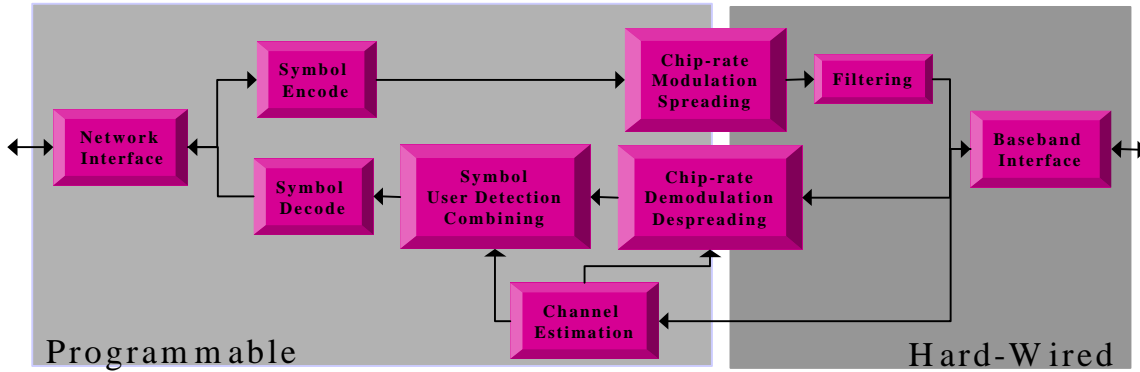


Figure 1: High Level BTS Functionality and Partitioning

It is generally understood that DSP chips are not suitable to entirely implement chip-rate processing. However, it is possible to design, in hardware, somewhat programmable blocks for well-defined portions of the rake receiver. Such blocks would reside outside a DSP and would be controlled by the DSP.

Therefore, the DSP functions could include all of symbol rate processing, along with control for chip-rate processing. The required bus bandwidth between external hardware and DSP is manageable, given today's DSP I/O capabilities. For example, the C6201's external memory bus could support up to 800Mbytes/sec which is sufficient bandwidth for a large number of channels, at symbol-rate.

We now proceed to analyze real-time processing requirements for the DSP (specifically, TMS320C6400), assuming the above presented partitioning.

III. TM320C6400 Features

TMS320C6400 is the Texas Instruments' highest performance DSP core. Like its predecessor, the C6200, the new C6400 is an enhanced VLIW architecture with 8 flexible functional units which can operate in parallel, each running at up to 1.1GHz (initially at 800MHz). Backward compatibility to C6200 is maintained. Additional features relevant to the basestation application are:

- Special-purpose instructions (Deal/Shuffle, Bit-Reverse, Variable Shift)
- Packed data processing (SIMD with dual-16 bit or quad-8 bit operations)
- Denser code, more registers and wider buses from memory to CPU.

Details of the CPU architecture and instruction set can be found in [1].

At the device level, various on-chip peripherals and I/O interfaces complement the CPU core. These peripherals include multiple parallel and serial busses, two-level cache and an efficient DMA controller.

IV. Implementation Details for Symbol-Rate functions

The high-level partitioning suggested in Section II assumes that symbol-rate function, as well as some chip-rate control functions are implemented in a programmable DSP.

In this section, we will show examples of how the architectural features of C6400 apply to symbol-rate functions performed in a 3G BTS. The sequence of symbol-rate functions for the transmitter side is shown in Figure 2. Receiver functions are simply inverted transmitter functions (i.e. decoder in place of encoder and deinterleaver in place of interleaver).

Most functions are implemented in C language and compiled using compiler revision 4.00. The implementation is optimized for the C6400 device.

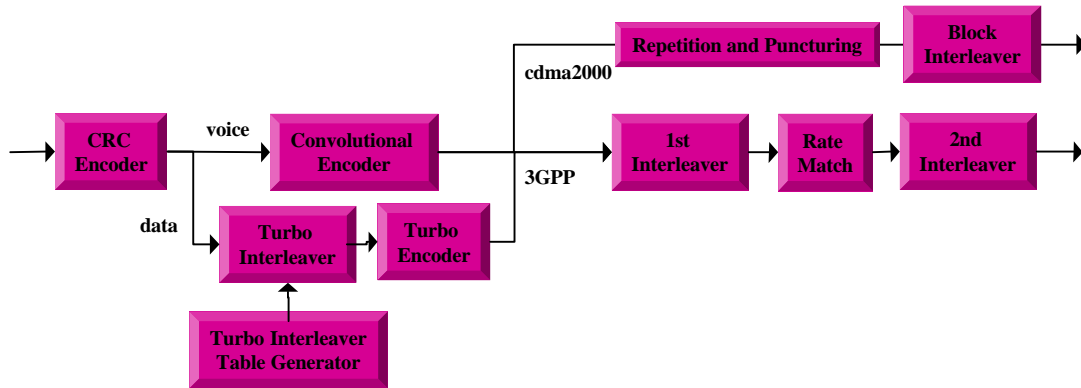


Figure 2: Symbol-Rate BTS functions for transmitter

A. CRC Encoder/Decoder

Description. The function of the CRC encoder is to compute an M-bit CRC, based on a frame of N bits and given generator polynomial of order M, and append these M bits at the end of the original N-bit frame. The function of the CRC decoder is to compute a M-bit CRC on received frame of (N+M) bits and compare to zero – the check is passed if the result is zero.

Implementation. This function is most efficiently implemented using a table look-up approach. The required table size for up to 16-th order generator polynomial is 256x2 bytes.

Performance. One iteration of the outer loop, which processes 32 bits of input data, takes 38 cycles, which translates to 0.04MHz for a voice channel, and scales linearly with data rate. The performance of the algorithm is limited by its sequential nature rather than CPU resources (functional units, registers, memory buses).

B. Convolutional Encoder

Description. A frame of N bits (in a bit-packed format) is encoded with a rate $r=1/2, 1/3$ or $1/4$, constraint length $K=9$ convolutional encoder. Individual outputs are multiplexed to produce a single $n*N$ -bit frame. Encoder polynomials vary from standard to standard.

Implementation. The implementation is based on a block-XOR method, where shifted versions of a 32-bit segment of the input stream are XOR-ed together to yield 16 bits of valid output at a time. This implementation is described in more details in [2].

Performance. For rate $1/2$ encoder, encoding and output multiplexing is implemented in a single for-loop. One iteration of this loop, which processes 32 bits of input data, takes 12 cycles, which translates into 0.01MHz for a voice channel and scales linearly with data rate.

For rate $1/3$ encoder, because of complexity of 3-way multiplexing, encoding and multiplexing is implemented in separate for-loops. A single iteration of the encoding loop takes 16 cycles, and a single iteration of the multiplexing loop takes 21 cycles, yielding 37 cycles for 32 bits of input, or 0.02MHz for a voice channel.

The implementation takes advantage of C6400's SHFL (shuffle) instruction, which takes two 16 bit registers and bit-wise multiplexes their contents into a single 32-bit register.

C. Viterbi Decoder

Description. The function of the Viterbi decoder is to estimate the most likely sequence of information bits. The estimation is done based on received noisy versions of encoded bits (in a 8-bit soft decision format), and the known structure of the trellis.

Implementation. For each trellis stage, $2^{(r-1)}$ branch metrics are computed, where r is the code rate. The most cycle intensive portion of the algorithm is computation of $2^{(K-1)/2}=128$ Viterbi butterflies (add/subtract, compare, select, store transition). At the end of the frame, traceback is performed starting from known last state (state 0), or, in the case of sliding window approach, from the state that, at the last trellis stage, has the highest accumulated metrics.

Performance. The Viterbi butterfly operation takes 10 cycles/8 butterflies. In addition, about 5 cycles per stage are required for branch metric computation (prior to state accumulation) and, 5 cycles per stage are for traceback (after the frame is completed). Overall, this results in 1.5MHz for one voice channel.

D. Turbo Encoder

Description. The input to the encoder is bit-packed information sequence. This sequence is bit-interleaved (off-place), based on a pre-generated look-up table for turbo interleaver. The original and interleaved sequences are then encoded using two rate 1/1, K=4 recursive convolutional encoders. The original sequence and two encoded sequences are then merged into a single bit-packed sequence using a 3-way multiplexing operation. If rate 1/2 is required, every other bit is punctured out and 2-way multiplexing is performed.

Implementation. Due to the recursive nature of the encoder, the operation of the kernel is bit-wise rather than 32-bit-block wise, as was the case with the convolutional encoder. Both encoders are performed inside the same for-loop. Multiplexing is performed separately for rate 1/3.

Performance. For rate 1/3 turbo code, a single iteration of the encoding loop, which processes one bit per iteration, takes 9 cycles. A single iteration of the multiplexing loop (32 bits per iteration) takes 21 cycles. Turbo interleaving takes 3.5 cycles/bit. Overall, turbo encoder takes 13.2 cycles/bit, or 4.92MHz for 384kbps channel.

The implementation takes advantage of following specialized C6400 instructions: ROTL (rotate left), SHFL (shuffle), and packed data processing instructions (PACK2, SHR2, SUB2).

E. Turbo Decoder

Description. The functionality of turbo decoder is described in more detail in [3]. Basic building blocks are MAP decoders and interleavers/deinterleavers. MAP decoding consists of accumulating state metrics in the forward direction (α), backward direction (β), and computing extrinsic information based on alpha, beta and local branch metrics (γ). We consider max* implementation, where following approximation is used:

$$\log(e^{L1} + e^{L2}) \approx \max(L1, L2) + \text{corr}(|L1 - L2|)$$

where $\text{corr}()$ is a lookup table with a small number of entries (less than 12).

Implementation. The kernel consists of three for-loops: (1) γ -computation, which takes 2 cycles/stage; (2) α -computation, which takes 10 cycles/stage, and (3) β /extrinsic computation, 20 cycles/stage. The main optimization method is to unroll the inner loop that runs through encoder's 8 states. In addition, to optimize for memory usage, sliding window approach can be taken. This is at expense of cycle-count performance since now there is a larger number of overall stages to be processed, due to overlap.

Performance. For the overall decoder (interleaving/deinterleaving and 2 MAP executions, for 6 iterations), at 384kbps, about 168.75MHz are required, based on a 100-bit sliding window.

F. Interleaving

Description. Channel interleaving in cdma2000 and 1st and 2nd interleaving in 3GPP are done using block-interleaving. Conceptually, bits are written into a two dimensional matrix row-wise and read from it column-wise. The number of columns can be 1,4,8,32 or 64.

Implementation. A single function implements interleaving for all possible column sizes C, and requires different lookup tables (of size C) for each column size.

Performance. A single iteration of inner loop, which operates on bits, takes 2 cycles. This translates to 0.06-0.11MHz for a voice channel.

G. Deinterleaving

Description. This function simply inverts the operation of the interleaver. However, the deinterleaver is operating on soft-decisions (assumed to be 8-bit signed quantities), whereas interleaver operates on bits.

Implementation. Each possible column value is implemented using a different C function, for maximum efficiency. Each implementation uses a small lookup table, whose size is equal to the number of columns. The kernel consists of an outer loop that counts rows, and inner loop which counts columns.

Performance. The inner loop performance is 1.25 cycles/byte, or 0.01-0.04 MHz for a voice channel. We see that the processing is simpler at byte-level than at bit-level. Since this function operates on 8-bit values, the implementation takes advantage of packed data processing instructions on C6400, such as PACK4, PACK2.

H. Rate Matching

Description. The function of rate matching (in the transmitter, operates on bits) and rate dematching (in the receiver, operates on 8-bit soft decisions) is to perform puncturing or repetition in order to achieve desired number of output bits, given some length of the input. Note that repetition and puncturing in cdma2000 is a simplified version of the more generic rate matching algorithm in 3GPP, and can be performed at the end of encoding with almost no performance penalty.

Implementation. Separate kernels are implemented for repetition and puncturing. In addition, each kernel is implemented for bit-packed values as well as bytes. The latter one is simpler and therefore more efficient. The puncturing kernel simply consists of stepping through the input array, incrementing the array index by one each time, and determining which input value to copy into the output array. The repetition kernel takes a value of the input array, copies it into the output array, determines, based on the 3GPP algorithm, whether it is necessary to repeat the value at that particular index, and, if so, performs one additional copy of the same input value into the output array.

Performance. For soft decisions (8-bit values), puncturing and repetition kernels take 3 cycles/input index, or 0.12Mhz for voice channel. For hard decisions (packed bits), puncturing kernel takes 4 cycles/index and repetition kernel takes 5 cycles per input index, or 0.16-0.20MHz. The implementation for hard decisions takes advantage of BITR (bit reverse) and SSHVR (variable shift) instructions.

V. Real-Time Requirement Summary

The C6400 implementations of symbol-rate functions described in the previous section yield, in summary, MHz/channel requirements shown in Table 1. Results for both cdma2000 and 3GPP are shown, taking as examples a voice channel at ~8kbps, and a data channel at ~384kbps:

FUNCTION		3GPP		cdma2000	
		voice	data	voice	data
Transmitter	CRC Encoder	0.04	0.47	0.02	0.37
	Convolutional Encoder	0.02	N/A	0.01	N/A
	Turbo Interleaver	N/A	1.17	N/A	1.00
	Turbo Encoder	N/A	3.75	N/A	2.47
	1st Interleaver	0.06	2.31	0.08	1.28
	Rate Match	0.17	4.65	N/A	N/A
	2nd Interleaver	0.11	1.97	N/A	N/A
	TOTAL	0.40	14.32	0.11	5.12
Receiver	2nd Deinterleaver	0.04	1.11	N/A	N/A
	Rate Dematch	0.12	2.90	N/A	N/A
	1st Deinterleaver	0.01	0.29	0.02	0.55
	Viterbi Decoder	1.50	N/A	1.40	N/A
	Turbo Decoder	N/A	168.75	N/A	135.00
	CRC Decoder	0.04	0.48	0.03	0.38
	TOTAL	1.71	173.53	1.44	135.93

Note: Turbo decoder MHz is based on 6 iterations, sliding window of 100-bits

Table 1: Real Time Performance Summary (MHz)

At 800MHz, a C6400 device could support up to 400 voice channels, or up to 4 384kbps channels.

VI. System-Level Considerations

The DSP MHz performance described in previous section is one of the most performance parameters, but a series of system-level constraints will ultimately determine if this highest performance level can be sustained. Some of the constraints are considered in this section.

Memory Requirements. The total code size for all symbol rate functions is below 16Kbytes. Data requirement is minimized by packing bits into 32-bit words at the transmitter. Since most functions are implemented off-place, separate buffers are needed for input arrays and outputs array. The implementation does not make significant use of static tables (apart from the turbo interleaver) and only dynamic channel buffers are required. If memory is a more critical parameter than MIPS performance, all interleaving functions can be performed in-place, with tables generated on the fly.

Memory Subsystem considerations. The memory architecture on the C6400 family is a two-level cache with 16Kbyte level 1 (L1) cache for each program and data. Level two (L2) memory can be used as mapped memory or cache. Typically, it will be used as mapped memory to transfer data into on-chip memory via DMA. A typical configuration could consist of data in L2 memory and program in L1 memory. Simulations have shown that cycle count penalty due to cache misses is not more than 10% on the average, and is lower at the transmitter than at the receiver.

Additional functions. In addition to symbol rate functions, network interface and chip-rate control functions can be performed on the DSP. Network interface may require HDLC-like or ATM-like functionality. Chip-rate control functions may include for example, channel estimation. The functions which are interfacing to the hardware may not consume the most MIPS, but may have tight timing requirements, therefore making the interrupt latency of the DSP an important consideration factor. And finally, some overhead should be allowed for a real-time operating system or a basic scheduler, such as DSP/BIOS. This aspect may be important since resources for a large number of channels need to be centrally controlled and managed.

VII. References

- [1] “*TMS320C6000 CPU and Instruction Set User’s Guide*”, Literature Number SPRU189E, Texas Instruments 2000.
- [2] “*Viterbi Decoding Techniques in the TMS320C54x family*”, Application Note, Literature Number SPRA071, Texas Instruments, 1999.
- [3] J. Nikolic-Popovic, “*Real-Time Implementation of cdma2000 Turbo Codes on TMS320C6200*”, Proceedings of ICPAT’99, Orlando, Florida.