Texas Instruments

Application Report

SPRA750D - September 2003

Using TMS320C6416 Coprocessors: Viterbi Coprocessor (VCP)

Jelena Nikolic-Popovic *Digital Signal Processing Solutions*

ABSTRACT

Viterbi Coprocessor (VCP) is a programmable peripheral for decoding of convolutional codes, integrated into Texas Instruments' TMS320C6416 DSP device. The VCP is controlled via memory mapped control registers and data buffers. Control registers can be accessed directly by the CPU, whereas data buffers are typically accessed using the EDMA controller. This application note describes the relationship between the theory of Viterbi decoding and VCP implementation, outlines VCP programming procedure, and provides examples. The examples demonstrate how to program VCP for typical 3GPP/IS2000 parameters.

Contents

List of Figures

List of Tables

1 Introduction

Viterbi Coprocessor (VCP) is a programmable peripheral for decoding of convolutional codes, integrated into Texas Instruments’ TMS320C6416 DSP device. The inputs into the coprocessor are 7-bit branch metrics, obtained by combining channel soft decisions. The outputs are bit-packed hard decisions, or 16-bit soft decisions. The VCP also computes the Yamamoto bit.

VCP programmable parameters are:

- Constraint length K (5, 6, 7, 8 or 9)
- Code rate (1/2, 1/3 or 1/4)
- Polynomials
- Frame length and termination (with or without tail bits)
- Initial conditions for state metric computation
- Threshold for Yamamoto bit generation

2 Background on Viterbi Decoding Algorithm

We are interested in the application of Viterbi algorithm application to decoding of convolutional codes.

Convolutional encoder can be thought of as a delay line with (K−1) elements. Parameter K is referred to as *constraint length*. Input to the delay line is a binary information sequence $\{u_n\}$ of length N. The sequence is shifted through a delay line, one bit at a time. For each input bit u_n, there are R=1/r output bits $x_{1n}, x_{2n}, \ldots, x_{Rn}$. Parameter r is referred to as the *code rate*. The output is formed by adding (modulo 2) outputs of delay line elements, according to binary polynomials $G_1=\{g_{11},g_{12},\ldots,g_{1K}\}$; $G_2=\{g_{21},g_{22},\ldots,g_{2K}\}$;..., $G_R=\{g_{R1},g_{R2},\ldots,g_{RK}\}$. The polynomials are usually specified in octal notation. For example, Figure 1 shows a K=9, R=1/2 convolutional encoder with polynomials $G_1G_2=\{561,753\}$.

To facilitate the decoding process, the initial state of delay elements is the all-zero state. In addition, by appending (K−1) zeros (tail bits) at the end of the N-bit input sequence, it is also ensured that the final state is the all-zero state.

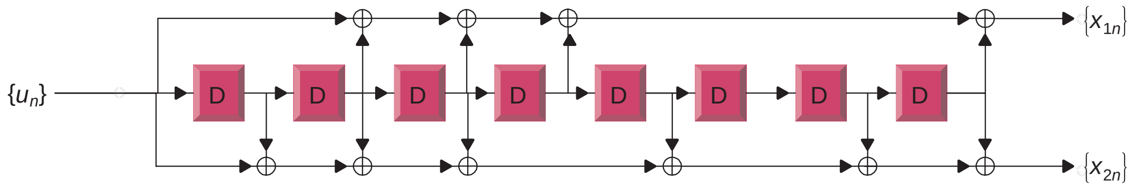


Figure 1. K=9, R=1/2 Convolutional Encoder

Viterbi algorithm is an efficient implementation of a maximum likelihood sequence detector. It produces the most likely transmitted sequence $\{u_{n,est}\}$, given received noisy sequence $\{y_n\}$. Throughout this document, it is assumed that values $\{y_n\}$ represent real, quantized analog values. This is referred to as *soft-decision input*.

In its application to decoding of convolutional codes, the received sequence $\{y_n\}$ is the noisy version of the encoded sequence $\{x_n\}$, and the algorithm estimates the most likely sequence at the input to the convolutional encoder $\{u_n\}$.

The most likely sequence is found by traversing (in forward and backward directions) a trellis whose structure is determined by convolutional code parameters. An example of a trellis for K=5 is shown in Figure 2.

The trellis consists of nodes (states) that are connected by branches. The total number of stages in the trellis, for a terminated frame, is (N+K−1), i.e., it represents the length N of the input data sequence, followed by (K−1) tail bits. At each stage, there are $2^{(K-1)}$ states. The state is the decimal representation of the contents of encoder's memory elements. Two branches are originated in each state (corresponding to binary inputs u_n=0 and u_n=1), and two branches are terminated in each state. Each branch is labeled with 1-bit input label ("0" or "1"), and R-bit output label. For example, on the branch connecting state 1 to state 0, the input label is i=0, indicating the bit that is shifted into the left-most delay element, and the output label is, for the encoder shown in Figure 1, i=11, indicating bits which are produced at the output of the encoder when bit 0 is present at the input, and encoder state is 1.

The entire trellis can be constructed from Viterbi butterflies, a structure consisting of two states at stage *n*, connected by two branches each to two states at stage *n+1*. One such butterfly is highlighted in Figure 2.

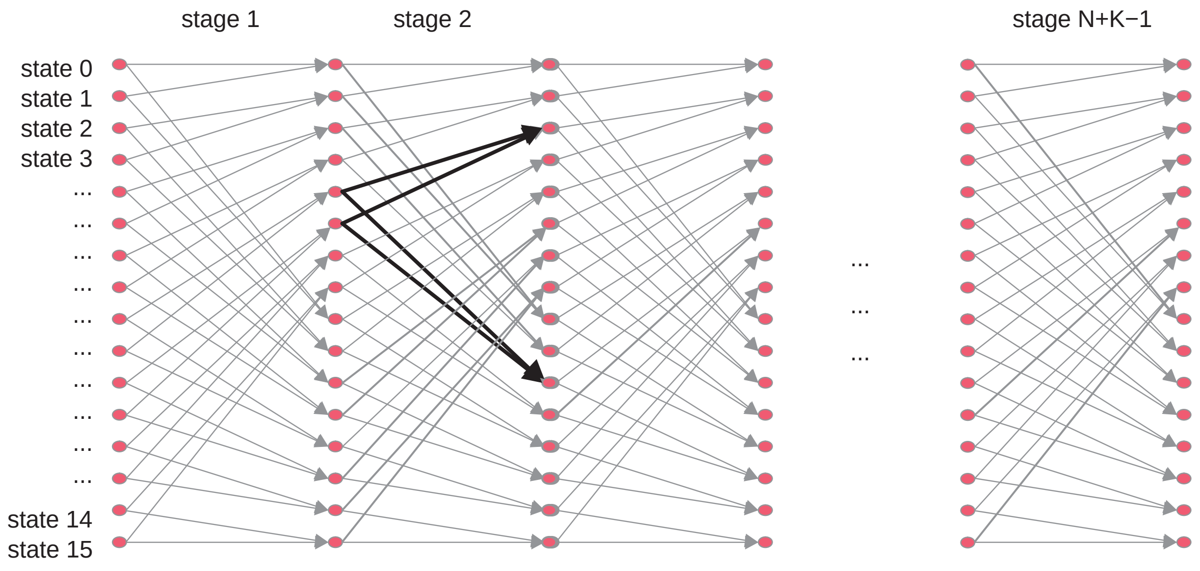


Figure 2. Trellis for a K=5 Convolutional Code

The main steps in the Viterbi algorithm are described in the following sections.

2.1 Branch Metrics Computation

Associated with each branch in the trellis is a branch metric. The branch metric is a measure of how "close" the received noisy values $y_n=\{y_{1n},y_{2n},...,y_{Rn}\}$ are to the output branch label $\boldsymbol{o}=\{o_1,o_2,...,o_R\}$.

For a rate r=1/R code, 2^R different branch output labels $\boldsymbol{o}$ are possible. Therefore, for each stage *n*, we need to compute 2^R branch metrics.

Branch metric $b_{\mathbf{o}}$ is computed as a Euclidean distance between the received noisy sample and branch label. This expression can be simplified as follows:

$$b_{\mathbf{o}n} = y_{1n}(-1)^{o1} + y_{2n}(-1)^{o2} + \ldots + y_{Rn}(-1)^{oR}$$

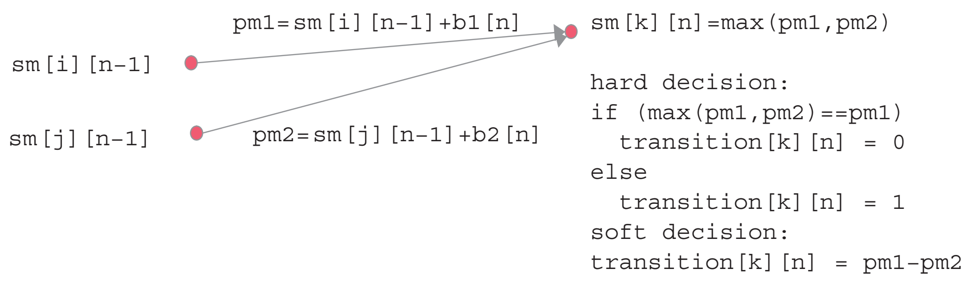
Due to symmetry, $b_{00n}=-b_{11n}$, and $b_{01n}=-b_{10n}$. It therefore suffices to compute 2^{R-1} branch metrics for each trellis stage *n*.

2.2 State Metric Computation

The trellis is traversed in the forward direction in order to accumulate branch metrics along paths through the trellis.

Viterbi algorithm is based on the fact that it is sufficient to accumulate *state* metrics *sm[k], k=0,...,*$2^{K-1}-1$. We have seen that two branches (corresponding to two paths) merge in each state. At each state, the path with the larger accumulated metric is chosen as the *survivor* and the other path is discarded. The path metric associated with the survivor path becomes state metric for the state and stage in which the two paths have merged.

The process of accumulating path metrics and selecting the survivor is graphically represented in Figure 3.



```
hard decision:
if (max(pm1,pm2)==pm1)
  transition[k][n] = 0
else
  transition[k][n] = 1
soft decision:
transition[k][n] = pm1-pm2
```

Figure 3. State Metric Accumulation

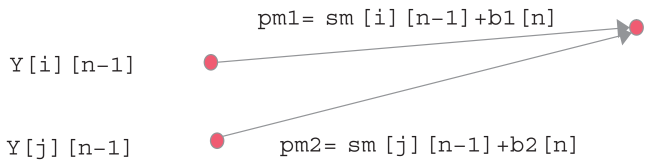
As will be seen in the next section, it is necessary to “remember” the input label of the branch belonging to the survivor path. This information is referred to as *transition bit* and is denoted as `transition[k][n]` in Figure 3. Therefore, one transition bit per state per stage needs to be saved for the next step in the algorithm.

At stage 0, state metrics need to be initialized. One of the choices is to initialize them all to zero. However, in order to take advantage of the fact that the initial state is zero, the state 0 can be “favored” by giving it a higher initial metric than the remaining states. For example, state zero could be initialized to 0 and remaining states to the smallest negative number.

2.3 Yamamoto Bit

In addition to the state metric `sm[k]`, associated with each state `k` is a Yamamoto bit `Y[k]`. The idea behind the Yamamoto bit is to “remember” if, at any stage in the trellis, the distance between the survivor path and the discarded path was smaller than the Yamamoto threshold. If this was the case, it is concluded that the decoding is not reliable and a higher layer in the network may decide to discard the entire frame. For a terminated trellis, the Yamamoto bit associated with state 0 of the last stage, `Y[0][N+K-1]`, is used as a binary frame quality indicator.

The Yamamoto bit is accumulated, as the trellis is traversed, as shown in Figure 4. Path metrics are computed for both paths merging in state `k`, as done for state metric accumulation. The absolute difference between the two path metrics is computed and compared to the Yamamoto threshold. If the difference is smaller than the threshold, Yamamoto bit is set to zero. If the difference is larger than the threshold, then the Yamamoto bit is set to the Yamamoto bit of the state through which the survivor path passed. Therefore, the Yamamoto bit is propagated along the survivor path.



```
if |pm1–pm2| > Threshold
    if pm1 = max(pm1,pm2)
      Y[k][n]=Y[i][n–1]
    else
      Y[k][n]=Y[j][n–1]
else
  Y[k][n] = 0;
```

Figure 4. Yamamoto Bit Accumulation

Note: The Yamamoto Bit may be falsely set to 0 when the number of symbols to be processed is not a multiple of 4 when (FL + (K−1)%4 = 1, 2, 3 for the last set of symbols to be processed is 1, 2, or 3. The extra (3, 2, 1) symbol stages of the Branch Metrics are automatically being inserted by the VCP as 0's. Automatically inserting the zeroed BM stages can cause the Yamamoto Bit to be falsely set to 0, thus falsely setting the Yamamoto Bit to 0. The user should always choose frame length such that FL + (K−1)%4 equals zero when using Yamamoto Bit to avoid this issue.

2.4 Traceback

At the start of the traceback, we first exploit the fact that the encoder terminates in state zero. The traceback therefore starts from state 0 at the last trellis stage, i.e., stage (N+K−2).

We then exploit the transition bits saved during state metric accumulation process. The transition bit associated with state 0 at stage (N+K−2), denoted *transition[0][N+K−2]* , gives information on the origin for the path which terminated in state 0 at stage (N+K−2). If the transition bit is 0, the origin is state 0 at stage (N+K−2), otherwise the origin is state 1 (see Figure 5).

By following the transition bits while traversing the trellis in the backward direction, we are effectively choosing the overall survivor path that corresponds to a particular input sequence. The sequence of input labels of branches along the survivor path is the decoded maximum likely sequence. In , the decoded sequence is u_{est} = {0,1,1,1}. The last four zeros in the path are tail bits and are not part of the information frame.

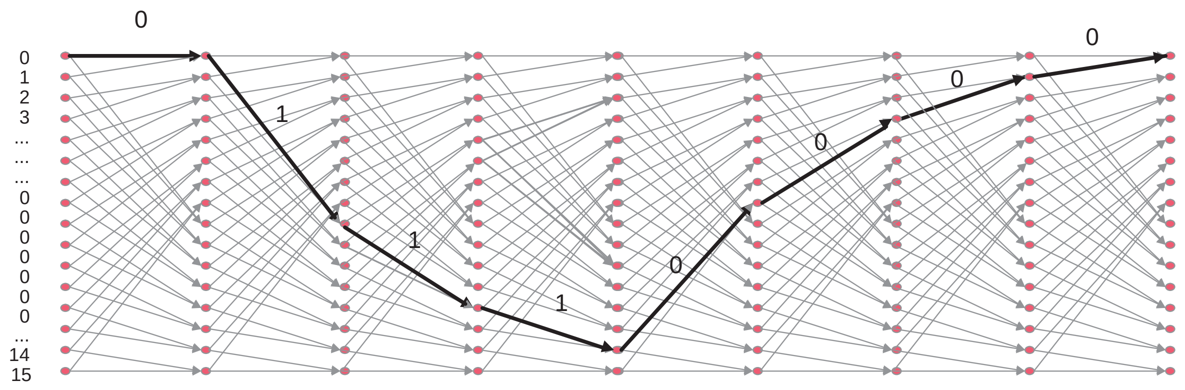


Figure 5. Example of Survivor Path and Associated Decoded Sequence

2.5 Sliding Window Processing

As seen in section 2.2, during state metric accumulation, the transition bits for all states and all stages need to be saved in order to perform traceback. In order to reduce storage requirements, Viterbi decoding can be split into blocks, which are referred to as sliding windows.

The sliding window concept is shown in Figure 6. The state metric accumulation starts at stage 0, and is performed continuously for the entire frame, over (N+K−1) stages. After state metrics (and transition bits) have been accumulated for the first window W1 of (R+C) stages, the traceback starts from the state which has the maximum accumulated metric at the last processed stage. In order to improve reliability of the decisions, the decisions for last C stages will not be used, only the first R. R is called *reliability length*, i.e., it is the portion of the window for which the decoding is *reliable*. C is *called convergence length*, i.e., it is the portion of the window for which the decoding is *converging*.

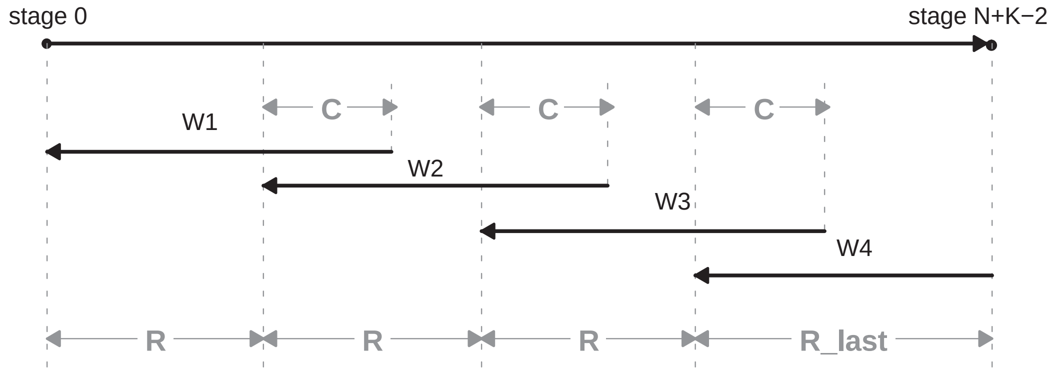


Figure 6. Sliding Window Processing

After the state metrics have been accumulated for additional R stages, the traceback for the second window starts from the state which has the maximum accumulated metric at the last processed stage.

Since the last C stages from the first window were discarded, the reliability portion R of the second window W2 overlaps with convergence portion of window W1.

If the overlap between windows is sufficiently large (Fourney’s rule states that the overlap should be up to 5*(K−1)), then there will be no noticeable degradation in the BER (Bit Error Rate) performance of the algorithm.

Note that the last window does not require convergence portion since the final state at stage (N+K−2) is known to be zero.

3 Relationship Between Viterbi Decoding Theory and VCP Implementation

In this section, we establish the relationship between the theory of Viterbi decoding and the VCP implementation, describing the significance of programmable VCP parameters which affect VCP algorithm. Those parameters are described in [2] and are reviewed in Table 1.

Table 1. Programmable VCP Parameters

Parameter Name	Parameter Description	Register	Size (bits)
F	Frame Length (excluding tail bits)	VCPIC2	16
R	Reliability length	VCPIC2	16
C	Convergence length	VCPIC3	16
TB	Traceback mode (tailed, mixed, convergent)	VCPIC5	2
YAMEN	Yamamoto bit computation enable bit	VCPIC1	1
YAMT	Yamamoto threshold	VCPIC1	12
IMAXS	Maximum state metric	VCPIC4	12
IMINS	Minimum state metric	VCPIC4	12
IMAXI	Maximum State Index	VCPIC5	8
SDHD	Soft decisions or hard decisions	VCPIC5	1
POLY[0:3]	Encoder polynomials	VCPIC0	4 x 8
SYMX	Determines number of symbols transferred per VCPXEVT	VCPIC5	4
SYMR	Determines number of symbols transferred per VCPREVT	VCPIC5	4
OUTF	Output parameter read flag	VCPIC5	1

NOTE: The parameters shown in gray in Table 1 are related to EDMA operation and do not affect the Viterbi algorithm functionality.

3.1 Code Parameters

VCP supports single shift register, rate 1/2, 1/3 or 1/4 convolutional codes with constraint length 5,6,7,8 and 9. Polynomials are programmable as 4x 8-bit values (**POLY**[0:3]), representing binary polynomial coefficients. The code rate and constraint length are not programmed directly, but are computed inside the VCP based on polynomials.

Frame length **F** is programmable. A frame can be terminated with tail bits (**TB** = tailed or mixed) or non-terminated (**TB** =convergent). If the frame is terminated with tail bits, the branch metrics for (K−1) tail bits are also required by the VCP.

3.2 Branch Metrics

For a rate r code, there is a total of $2^{1/r-1}$ different branch metrics which are formed by combining 1/r soft decisions. Branch metrics at the input to the VCP are 7-bit signed values. Additional limitation on the dynamic range of soft decisions comes from state metrics accumulation and is discussed in the section 3.3.1.

3.3 State Metrics

3.3.1 State Metric Accumulation

State metrics are accumulated modulo−2^{12} (the size of accumulated state metric registers is 12 bit). According to a literature result published in [1], modulo−2^{C} truncation of state metrics can be performed without loss of decoding performance if the branch metrics satisfy the following bound:

$$2^{C-1} - 1 \geq (2(K - 1) + 2)B$$

where K is constraint length and B is upper bound for branch metrics.

For example, for C=12 and K=9, the branch metric bound is $B \leq 113.7$ which is slightly smaller than the available 7-bit input range. Since branch metric is a combination of 1/r soft decisions, assuming that soft decisions have the same upper bound, the corresponding bound for soft decisions is 56.8 for rate 1/2, 37.9 for rate 1/3, and 28.4 for rate 1/4 codes.

3.3.2 State Metric Initialization

At the beginning of each frame, state metrics are initialized in the following manner: the state at index **IMAXI** (user input) is set to value **IMAXS** (user input). All other states are set to value **IMINS** (user input). **IMAXS** and **IMINS** are 12-bit signed values. Typically, initial state is known to be zero, **IMAXI**=0.

3.4 Yamamoto Bit

As seen in section 2.3, the computation of the Yamamoto bit requires a threshold, **YAMT**. The threshold is input to the VCP on a per-frame basis. It is a 12-bit value. If Yamamoto bit computation is enabled, i.e., **YAMEN** bit is set, the Yamamoto bit is reported for each frame in VCP’s output register VCPOUT1.

3.5 Traceback

For non-sliding window processing, or for the last window in sliding window processing mode for terminated frames (**TB** = tailed, mixed) , the traceback starts from state 0.

For all intermediate sliding windows, as well as the final window for non-terminated frames (**TB** = convergent), the traceback starts from the state with the largest accumulated state metric.

3.6 Sliding Window Processing

Frame decoding is split into sliding windows, as discussed in section 2.5 , due to the finite size of traceback memory. The traceback memory, internal to the VCP, accommodates $128*256/2^{(K-1)}$ stages for hard decision decoding, or $32*256/2^{(K-1)}$ trellis stages for soft decision decoding.

Frame processing does not need to be split into sliding windows if the frame length (not including tail bits) observes bounds shown in Table 2.

Table 2. Maximum Frame Length for Non-Sliding Window Processing

Constraint Length K	Maximum Frame Length for Non-Sliding Window Processing (TB = tailed)	
	Hard Decisions	**Soft Decisions**
K=9	120	24
K=8	217	49
K=7	378	90
K=6	635	155
K=5	2044	508

If the length of the frame to be decoded does not satisfy bounds from Table 2, sliding window processing is used, and reliability length **R** and convergence length **C** need to be programmed. Some restrictions apply to the selection of **R** and **C** and are listed in Table 3. Note that, for soft decisions, the only choices of **C** are 3(K−1) and 6(K−1), and **R** is fixed given the constraint length K.

The correct operation of VCP is not guaranteed if these conditions on **R** and **C** are not satisfied.

Table 3. Hard Decisions and Soft Decisions with Mixed/Convergent Modes

	Hard Decisions			**Soft Decisions**		
	Traceback mode			Tradeback mode		
	Tailed	**Mixed*/Convergent**		**Tailed**	**Mixed)/Convergent**	
	Fmax	R+C	C possible values	Fmax	R, C=3(K-1) non-punctured code	R, =6(K-1) (punctured code)
K=9	120	124	3,6,9,12,15 * (K-1)	24	R=4, C=24	not allowed
K=8	217	217	3,6,9,12,15,18 *(K-1)	49	R=28, C=21	R=7, C=42
K=7	378	372	3,6,9,12,15,18 *(K-1)	90	R=60, C=18	R=54, C=36
K=6	635	605	3,6,9,12,15,18 *(K-1)	155	R=60, C=15	R=60, C=30
K=5	2044	1020	3,6,9,12,15,18 *(K-1)	508	R=60, C=12	R=60, C=24

*Mixed mode is not allowed for frame sizes that can be handled in tailed mode

Note: Additional configurations that are valid for F, R, and C are R=192, C=96, Rate=1/3, K=7, Convergent mode, Hard decision, and Frame lengths = 278, 310, 342, 358, 480, 482, 486, 624, 626, 768, 770, and 802.

4 VCP Programming Procedure

This section outlines steps required to decode a single frame of data using the VCP. For possible approaches to decoding of multiple frames of the same or different user channels, see section 6.

4.1 Initialize Input Buffers

The user computes branch metrics and stores them in DSP internal or external memory. For a terminated frame with **F** information bits, and code with constraint length K, the total number of symbols is Ntot=**F**+K−1. For non-terminated frame, i.e., no tail bits, the total number of input symbols is Ntot=**F**.

For rate r, constraint length K code, there will be Ntot*($2^{1/r-1}$) 7-bit branch metrics. Branch metrics are organized as described in [].

The DSP memory address of the beginning of the pre-computed branch metrics array will be referred to as &bm[0]. The beginning of the branch metric array should be aligned on a 64-bit boundary.

4.2 Allocate Output Buffers

Hard decisions are transferred from the VCP in 64-bit words, stored in a bit-packed manner. Therefore, for a frame with **F** information bits, the size of the allocated output buffer should be ceil[**F**/64]*8 bytes.

Soft decisions are transferred from the VCP also in 64-bit words, but each soft decision is 16 bits. Therefore, for a frame with **F** information bits, the size of the allocated output buffer should be ceil[**F**/4]*8 bytes.

If the output parameter read flag is set (**OUTF**=1), two additional 64-bit words should be allocated for the output parameter word.

The DSP memory addresses of the beginning of the allocated buffers for VCP decisions and output parameters will be referred to as &sdhd[0] and &output_p[0], respectively. All buffers should be aligned on an 64-bit boundary.

4.3 Prepare VCP Input Configuration Word

For each frame, VCP input configuration registers VCPIC0−VCPIC5 are programmed as shown in []. The register configuration is first prepared in the DSP memory (internal or external). It is transferred to the VCP via EDMA once the VCP is started. The DSP memory address of the beginning of the prepared input configuration is denoted &input_config[0].

4.4 Prepare EDMA Links

When the VCP is ready to process a frame of data, it sends a series of synchronization events (VCPXEVT) to the EDMA, indicating that VCP is ready to receive data. Similarly, once decoding of one frame is completed, the VCP sends another series of synchronization events (VCPREVT) to the EDMA, indicating that data is ready.

For each EDMA transfer, programmable parameters are: (1) transfer options, (2) source address, (3) destination address, (4) frame and element count, (5) frame and element index, (6) reload count and link address. These parameters are described in detail in [3].

The EDMA transfer parameters are summarized in Table 4 and described in detail in [2].

The third row of the table represents the address in the paRAM. Link 0 of each VCPXEVT and VCPREVT have to be programmed at fixed locations in the paRAM, denoted as ADDR_VCPXEVT and ADDR_VCPREVT, respectively. Other links could be programmed anywhere in the paRAM. These additional locations in the paRAM are denoted RELOAD1, RELOAD2, etc.

Element count ELECNT and frame count FRMCNT for VCPXEVT Link 1 and VCPREVT Link 0 are computed from formulas given in [2].

The LINK entry in each parameter set represents the paRAM address of the next linked transfer. LINK=NULL indicates that the next transfer is the NULL transfer used for termination (see [3]).

Table 4. EDMA Links

VCPXEVT Links				**VCPREVT Links**			
Link 0		**Link 1**		**Link 0**		**Link 1 (optional)**	
paRAM address = ADDR_VCPXEVT		paRAM address = RELOAD1		paRAM address = ADDR_VCPREVT		paRAM address = RELOAD2	
OPT: SUM=DUM=INC		OPT: SUM=INC, DUM=FIXED		OPT: SUM=FIXED, DUM=INC (TB=mixed), DUM=DEC(TB=tailed)		OPT: SUM=DUM=INC TCINT=1, TCC = VCPREVT	
SRC= &input_config[0]		SRC= &bm[0]		SRC=VCPDECS		SRC=VCPOUT0	
FRMCNT= 0	ELECNT= 6	FRMCNT	ELECNT	FRMCNT	ELECNT	FRMCNT= 0	ELECNT= 2
DST=VCPIC0		DST=VCPWBM		DST= &sdhd[]		DST= &output_p[0]	
FRMIDX= N/A	ELEIDX= N/A	FRMIDX= N/A	ELEIDX = N/A	FRMIDX = N/A	ELEIDX = N/A	FRMIDX = N/A	ELEIDX = N/A
ELERLD= N/A	LINK =RELOAD1	ELERLD = N/A	LINK = NULL	ELERLD= N/A	LINK = RELOAD2 (OUTF=1) =NULL (OUTF=0)	ELERLD = N/A	LINK= NULL

4.4.1 *Special VCP EDMA Programming Considerations*

The EDMA parameters consist of six words as illustrated in Table 4. All EDMA transfers, in the context of the VCP, must be done using 32-bit word elements, must contain an even number of words, and must and have source and destination addresses double-word aligned.

Note: All EDMA transfers must be double-word aligned and the element count for the VCP EDMA transfer must be a multiple of two. Single-word transfers that are not double-word aligned will cause errors in TCP/VCP memory.

4.5 Start EDMA

The EDMA channels corresponding to VCPREVT and VCPXEVT are enabled in the EDMA Event Enable Register (EER), and these channels are also allowed to generate CPU interrupts by setting appropriate bits in the Channel Interrupt Enable Register (CIER). The EDMA control registers are described in detail in [3].

4.6 Start VCP

CPU writes a “START” command into VCP’s execution word register VCPEXE. This causes the VCP to generate the first VCPXEVT expecting input control. This in turn triggers the EDMA transfer which is programmed into the Event paRAM location corresponding to VCPXEVT.

4.7 Service EDMA Interrupt from VCP Channel at the End of Decoding

The EDMA link associated with the last VCPREVT is configured to generate a CPU interrupt. In the CPU interrupt service routine, the output decision buffer for the completed frame can be processed and decoding of next frame can be initiated.

5 VCP/EDMA Configuration Examples

In this section we show how to program VCP to decode a single frame of data, with typical 3G wireless decoding parameters.

For each example, we will discuss how to determine the VCP/EDMA configuration parameters. Settings of **F**,**R**, **C** , and **SYMX** and **SYMR** will be discussed in detail.

State metric initialization will be set assuming that state 0 is the known start state, so **IMAXI** = 0. This state will be given preference by setting the initial value to **IMAXS**=0x400 (i.e., ½ of the maximum absolute value), and keeping the initial value of the remaining states to **IMINS**= 0x0.

EDMA link configuration for VCPXEVT Link 0 (write to VCP input configuration) and VCPREVT Link 1 (read from VCP output parameters) are constant in all cases and are shown in Table 4. They will not be repeated in the examples, but it is understood that they need to be programmed.

5.1 Hard Decision Outputs

5.1.1 3GPP 12.2kbps – Class A

Using an example from [4], one 20msec AMR speech frame, with output data rate of 12.2kbps, is split into three transport channels. The first transport channel carries 81 class A bits and 12 CRC bits, for a total of 93 bits and is encoded using rate 1/3 convolutional code with constraint length K=9 and polynomials {0x6F, 0xB3, 0xC9, 0x00}.

The branch metric buffer consists of (93 information bits + 8 tail bits)*4 branch metrics/bit = 404 bytes. The output hard decision buffer is ceil[93/64]*8 = 16 bytes. The VCP/EDMA configuration is determined as follows:

1. F/R/C: F=93. Since F < 120, TB mode is tailed, i.e., the processing is not split into sliding windows (see Table 2). R and C are therefore not used. In tailed TB mode, the hard decisions are written in reverse order, i.e., last 64-bit first, so the EDMA destination address should be initialized to the end of the decision buffer, and destination address mode should be “decrement”.

2. SYMX, ELECNT/FRMCNT, for VCPXEVT link 1:

   Since **RATE**=3, $SYMX_{\max} = 8 - 1 = 7$. One possibility is to let $\textbf{SYMX} = SYMX_{\max} = 7$ which results in:

$$
\begin{aligned}
ELECNT &= 2^{RATE-1} \times \min\left(\textbf{SYMX} + 1,\ \frac{\textbf{FL} + K - 1}{4}\right) \\
&= 2^{\textbf{RATE}-1} \times (\textbf{SYMX} + 1) = 2^2 \times 8 = 32
\end{aligned}
$$

   and

$$FRMCNT = ceil\left(\frac{\textbf{FL} + K - 1}{4(\textbf{SYMX} + 1}\right) - 1 = ceil\ \frac{93 + 9 - 1}{4 * 8} - 1 = ceil\ \frac{101}{32} - 1 = 3$$

   The total number of branch metrics transferred in this case is $ELECNT * (FRMCNT + 1) * 4 = 32 * 4 * 4 = 512$, whereas the actual number of branch metrics needed is $(\textbf{FL} + K - 1) * 2^{\textbf{RATE}-1} = 101 * 4 = 404$.

   Although the VCP will work correctly, the EDMA bandwidth can be optimized by setting **SYMX** = 6, which results in *ELECNT* = 28 and *FRMCNT* = 3 . In the latter case, the total number of transmitted branch metrics is 448.

3. SYMR, ELECNT/FRMCNT, for VCPREVT link 0:

   For hard decisions, $\textbf{SYMR} = \min\left(SYMR_{\max},\ ceil\left(\frac{\textbf{FL}}{64}\right) - 1\right) = \min(15,\ 2 - 1) = 1$. Then,

$$ELECNT = 2 * ceil\ \frac{\min((\textbf{SYMR} + 1) * 64, \textbf{FL})}{64} = 2 * ceil\ \frac{\textbf{FL}}{64} = 4$$

   and

$$FRMCNT = ceil\ \frac{\textbf{FL}}{(\textbf{SYMR} + 1) * 64} - 1 = 0$$

4. POLY: POLY[0] = 0x6F (corresponds to 557), POLY[1] = 0xB3 (corresponds to 663), POLY[2]= 0xC9 (corresponds to 771), POLY[3]=0;

We assume that the output parameters will not be read (**OUTF**=0), so that only one link is programmed for the VCPREVT. The resulting VCP/EDMA configuration is shown in Table 5.

Table 5. VCP/EDMA Configuration for 3GPP 12.2kbps (AMR Speech Frame – Class A)

Input Configuration			
F = 93	R = N/A	C = N/A	TB = Tailed
YAMT=0 (YAMEN=0)	IMAXS = 0x400	IMINS = 0	IMAXI = 0
SDHD = HD	SYMX = 6	SYMR = 1	OUTF = 0
POLY[0] = 0x6F	POLY[1] = 0xB3	POLY[2] = 0xC9	POLY[3] = 0x0
VCPXEVT Links		**VCPREVT links**	
Link 1		**Link 0**	
OPT: SUM = INC, DUM = FIXED		OPT: SUM = FIXED, DUM = DEC, TCINT = 1, TCC = VCPREVT	
SRC= `&bm[0]`		SRC =VCPDECS	
FRMCNT= 3	ELECNT= 28	FRMCNT= 0	ELECNT= 4
DST=VCPWBM		DST= `&sdhd[3]`	
FRMIDX= N/A	ELEIDX= N/A	FRMIDX= N/A	ELEIDX= N/A
ELERLD= N/A	LINK= NULL	ELERLD= N/A	LINK= NULL

5.1.2 32kbps

For 32kbps data rate, with 10msec frames, the frame length (without tail bits) is 320 bits. The frame is encoded using rate ½, K=9 convolutional code with polynomials {0x71,0xEB, 0x00, 0x00}.

1. F/R/C: F=320. Since F > 120, TB mode is mixed, i.e., the processing is split into sliding windows). A common selection for C is 3(K−1)=24. With (R+C) < 124 and R%4=0, we can select R=80. In mixed TB mode, the hard decisions are written first 64-bit first, so the EDMA destination address should be initialized to the start of the decision buffer, and destination address mode should be “increment”.

2. SYMX, ELECNT/FRMCNT, for VCPXEVT link 1:

   Since **RATE** = 2, $SYMX_{\max} = 16 - 1 = 15$. One possibility is to let $\mathbf{SYMX} = SYMX_{\max} = 15$ which results in:

$$ELECNT = 2^{RATE-1} \times \min\left(\mathbf{SYMX} + 1, \frac{\mathbf{FL} + K - 1}{4}\right)$$
$$= 2^{\mathbf{RATE}-1} \times (\mathbf{SYMX} + 1) = 2^{1} \times 16 = 32$$

   and

$$FRMCNT = ceil\left(\frac{\mathbf{FL} + K - 1}{4(\mathbf{SYMX} + 1}\right) - 1 = ceil\,\frac{320 + 9 - 1}{4 * 16} - 1 = ceil\,\frac{328}{64} - 1 = 5$$

   The total number of branch metrics transferred in this case is $ELECNT * (FRMCNT + 1) * 4 = 32 * 6 * 4 = 768$, whereas the actual number of branch metrics needed is $(\mathbf{FL} + K - 1) * 2^{\mathbf{RATE}-1} = 328 * 2 = 656$.

Although the VCP will work correctly, the EDMA bandwidth can be optimized by setting **SYMX** = 13, which results in *ELECNT* = 28 and *FRMCNT* = 5. In the latter case, the total number of transmitted branch metrics is 672.

3. SYMR, ELECNT/FRMCNT, for VCPREVT link 0:

   For hard decisions, $\mathbf{SYMR} = \min\left(SYMR_{max},\ ceil\left(\frac{\mathbf{FL}}{64}\right) - 1\right) = \min(15,\ 5 - 1) = 4$. Then,

   $$ELECNT = 2 * ceil\ \frac{\min((\mathbf{SYMR} + 1) * 64, \mathbf{FL})}{64} = 2 * ceil\ \frac{\mathbf{FL}}{64} = 10$$

   and

   $$FRMCNT = ceil\ \frac{\mathbf{FL}}{(\mathbf{SYMR} + 1) * 64} - 1 = 0$$

4. POLY: POLY[0] = 0x71 (corresponds to 561), POLY[1] = 0xEB (corresponds to 753) and POLY[2]=0, POLY[3]=0;

We assume that the output parameters will not be read (**OUTF**=0), so that only one link is programmed for the VCPREVT. The resulting VCP/EDMA configuration is shown in Table 6.

Table 6. VCP/EDMA Configuration for 3GPP 32kbps Frame

Input Configuration			
F = 320	R = 80	C = 24	TB = Mixed
YAMT=0 (YAMEN=0)	IMAXS = 0x400	IMINS = 0	IMAXI = 0
SDHD = HD	SYMX= 13	SYMR= 4	OUTF = 0
POLY[0] = 0x71	POLY[1] = 0xEB	POLY[2] = 0x0	POLY[3] = 0x0
VCPXEVT Links		**VCPREVT links**	
Link 1		**Link 0**	
OPT: SUM = INC, DUM = FIXED		OPT: SUM = FIXED, DUM = INC, TCINT = 1, TCC = VCPREVT	
SRC= `&bm[0]`		SRC =VCPDECS	
FRMCNT= 5	ELECNT= 28	FRMCNT= 0	ELECNT= 10
DST= VCPWBM		DST= `&sdhd[0]`	
FRMIDX= N/A	ELEIDX= N/A	FRMIDX= N/A	ELEIDX= N/A
ELERLD= N/A	LINK= NULL	ELERLD= N/A	LINK= NULL

5.1.3 IS2000 RC3 Voice

In this example we consider the channel structure for the reverse fundamental and supplemental channel, and for radio configuration 3 (see [5]), 9.6kbps data rate. In this case, rate ¼, K=9 convolutional coding is used with polynomials {0xF5,0xB9,0x4B, 0x3B}. There are 184 bits/frame, including frame quality bits (excluding 8 tail bits).

5. F/R/C: F=184. Since F > 120, TB mode is mixed, i.e., the processing is split into sliding windows. A common selection for C is 3(K−1)=24. With (R+C) < 124, and R%4=0, we can

select R=92. In mixed TB mode, the hard decisions are written first 64-bit first, so the EDMA destination address should be initialized to the start of the decision buffer, and destination address mode should be “increment”.

6. SYMX, ELECNT/FRMCNT, for VCPXEVT link 1:

   Since **RATE** = 4, $SYMX_{max} = 4 - 1 = 3$. One possibility is to let $\mathbf{SYMX} = SYMX_{max} = 3$ which results in:

$$ELECNT = 2^{RATE-1} \times \min\left(\mathbf{SYMX} + 1,\ \frac{\mathbf{FL} + K - 1}{4}\right)$$
$$= 2^{\mathbf{RATE}-1} \times (\mathbf{SYMX} + 1) = 2^3 \times 4 = 32$$

   and

$$FRMCNT = ceil\left(\frac{\mathbf{FL} + K - 1}{4(\mathbf{SYMX} + 1}\right) - 1 = ceil\,\frac{184 + 9 - 1}{4 * 4} - 1 = ceil\,\frac{192}{16} - 1 = 11$$

   The total number of branch metrics transferred in this case is $ELECNT * (FRMCNT + 1) * 4 = 32 * 12 * 4 = 1536$, whereas the actual number of branch metrics needed is $(\mathbf{FL} + K - 1) * 2^{\mathbf{RATE}-1} = 192 * 8 = 1536$, so no extra branch metrics are transferred and the above choice provides an optimal use of the EDMA bandwidth.

7. SYMR, ELECNT/FRMCNT, for VCPREVT link 0:

   For hard decisions, $\mathbf{SYMR} = \min\left(SYMR_{max},\ ceil\left(\frac{\mathbf{FL}}{64}\right) - 1\right) = \min(15,\ 3 - 1) = 2$. Then,

$$ELECNT = 2 * ceil\,\frac{\min((\mathbf{SYMR} + 1) * 64, \mathbf{FL})}{64} = 2 * ceil\,\frac{\mathbf{FL}}{64} = 6$$

   and

$$FRMCNT = ceil\,\frac{\mathbf{FL}}{(\mathbf{SYMR} + 1) * 64} - 1 = 0$$

8. POLY: POLY[0] =0xF5 (corresponds to 561), POLY[1] = 0xB9 (corresponds to 753) and POLY[2]= 0x4B, POLY[3]= 0x3B

In this case, we assume that the output parameters will be read (**OUTF**=1), the primary reason is to read the Yamamoto bit. We will set the Yamamoto threshold to **YAMT** = 0x10 (this is a relatively low value compared to the available 11-bit range). The VCPREVT Link 1 is therefore required. The resulting VCP/EDMA configuration is shown in Table 7.

Table 7. VCP/EDMA Configuration for IS2000 RC3 Full Rate

Input Configuration			
F = 184	R = 92	C = 24	TB = Mixed
YAMT = 0x10 (YAMEN=1)	IMAXS = 0x400	IMINS = 0	IMAXI = 0
SDHD = HD	SYMX = 3	SYMR = 2	OUTF = 1
POLY[0] = 0xF5	POLY[1] = 0xB9	POLY[2] = 0x4B	POLY[3] = 0x3B
VCPXEVT Links		**VCPREVT links**	
Link 1		**Link 0**	
OPT: SUM = INC, DUM = FIXED		OPT: SUM = FIXED, DUM = INC, TCINT = 1, TCC = VCPREVT	
SRC= `&bm[0]`		SRC =VCPDECS	
FRMCNT= 11	ELECNT= 32	FRMCNT= 0	ELECNT= 6
DST=VCPWBM		DST= `&sdhd[0]`	
FRMIDX= N/A	ELEIDX= N/A	FRMIDX= N/A	ELEIDX= N/A
ELERLD= N/A	LINK= NULL	ELERLD= N/A	LINK= 1

5.1.4 GSM/EDGE AFS Frames

For one 10msec frame of AFS-coded 12.2kbps voice channel, the frame length is 250 bits, code rate is ½, and constraint length K=5. The actual polynomials are recursive, {G0/G0,G1/G0}, with G0 = 1 + D3 + D4 and G1 = 1 + D + D3 + D4. The VCP does not natively support recursive codes. It should be configured for feed-forward polynomials, {G0,G1}, and the output hard decisions should be followed by re-encoding using the feedback polynomial G0 to obtain the final hard decisions. Therefore, the VCP polynomials are {0x30, 0xB0, 0x00, 0x00}.

1. F/R/C: F=250. Since F < 2044, TB mode is tailed, i.e., the processing is not split into sliding windows (see Table 2). R and C are therefore not used. In tailed TB mode, the hard decisions are written in reverse order, i.e., last 64-bit first, so the EDMA destination address should be initialized to the end of the decision buffer, and destination address mode should be "decrement".
2. SYMX_ACT, ELECNT/FRMCNT, for VCPXEVT link 1:

   Since **RATE** = 2, $SYMX_{\text{max}} = 16 - 1 = 15$. One possibility is to let $\mathbf{SYMX} = SYMX_{\text{max}} = 15$ which results in:

   $$\begin{aligned} ELECNT &= 2^{RATE-1} \times \min\left(\mathbf{SYMX} + 1, \frac{\mathbf{FL} + K - 1}{4}\right) \\ &= 2^{\mathbf{RATE}-1} \times (\mathbf{SYMX} + 1) = 2^1 \times 16 = 32 \end{aligned}$$

   and

   $$FRMCNT = ceil\left(\frac{\mathbf{FL} + K - 1}{4(\mathbf{SYMX} + 1)}\right) - 1 = ceil\,\frac{250 + 5 - 1}{4 * 16} - 1 = ceil\,\frac{254}{64} - 1 = 3$$

   The total number of branch metrics transferred in this case is $ELECNT * (FRMCNT + 1) * 4 = 32 * 4 * 4 = 512$, whereas the actual number of branch metrics needed is $(\mathbf{FL} + K - 1) * 2^{\mathbf{RATE} - 1} = 254 * 2 = 508$. Therefore, the above selection provides the optimal use of EDMA bandwidth.

3. SYMR_ACT, ELECNT/FRMCNT, for VCPREVT link 0:

   For hard decisions, $\mathbf{SYMR} = \min\left(SYMR_{max},\ ceil\left(\frac{\mathbf{FL}}{64}\right) - 1\right) = \min(15,\ 4 - 1) = 3$. Then,

   $$ELECNT = 2 * ceil\ \frac{\min((\mathbf{SYMR} + 1) * 64, \mathbf{FL})}{64} = 2 * ceil\ \frac{\mathbf{FL}}{64} = 8$$

   and

   $$FRMCNT = ceil\ \frac{\mathbf{FL}}{(\mathbf{SYMR} + 1) * 64} - 1 = 0$$

4. POLY: POLY[0] =0x30 (corresponds to G0=1 + D3 + D4), POLY[1] = 0xB0 (corresponds to G1=1 + D + D3 + D4) and POLY[2]=0xC9 POLY[3]=0x0;

We assume that the output parameters will not be read (**OUTF**=0), so that only one link is programmed for the VCPREVT. The resulting VCP/EDMA configuration is shown in Table 8.

Table 8. VCP/EDMA Configuration for GSM − HD

Input Configuration			
F = 250	R = N/A	C = N/A	TB = Tailed
YAMT=0 (YAMEN=0)	IMAXS = 0x400	IMINS = 0	IMAXI = 0
SDHD = HD	SYMX = 15	SYMR = 3	
POLY[0] = 0x30	POLY[1] = 0xB0	POLY[2] = 0x00	POLY[3] = 0x00
VCPXEVT Links		**VCPREVT links**	
Link 1		**Link 0**	
OPT: SUM = INC, DUM = FIXED		OPT: SUM = FIXED, DUM = DEC, TCINT = 1, TCC = VCPREVT	
SRC= `&bm[0]`		SRC =VCPDECS	
FRMCNT= 3	ELECNT= 32	FRMCNT= 0	ELECNT= 8
DST=VCPWBM		DST= `&sdhd[7]`	
FRMIDX= N/A	ELEIDX= N/A	FRMIDX= N/A	ELEIDX= N/A
ELERLD= N/A	LINK= NULL	ELERLD= N/A	LINK= NULL

5.2 Soft Decision Outputs

5.2.1 GSM/EDGE AFS Frames

For this example, we consider same coding parameters as in section 5.1.4. Only **R**,**C** and SYMR_ACT parameters need to be recomputed:

1. F,R,C: F=250. Since F < 508, TB mode is tailed, i.e., the processing is not split into sliding windows (see Table 2). R and C are therefore not used. In tailed TB mode, the soft

decisions are written in reverse order, i.e., last 64-bit value first, so the EDMA destination address should be initialized to the end of the decision buffer, and destination address mode should be "decrement".

2. SYMX, ELECNT/FRMCNT, for VCPXEVT link 1:

   The branch metric transfer is the same for hard decision and soft decision outputs, so the selections for SYMX, ELECNT and FRMCNT are those previously computed in section 5.1.4.

3. SYMR, ELECNT/FRMCNT, for VCPREVT link 0:

   For soft decisions, $\mathbf{SYMR} = \min\left(SYMR_{\max},\ ceil\left(\frac{\mathbf{FL}}{4}\right) - 1\right) = \min(15,\ 63 - 1) = 15$. Then,

   $$ELECNT = 2 * ceil\ \frac{\min((\mathbf{SYMR} + 1) * 4, \mathbf{FL})}{4} = 2 * ceil\ \frac{(\mathbf{SYMR} + 1) * 4}{4} = 32$$

   and

   $$FRMCNT = ceil\ \frac{\mathbf{FL}}{(\mathbf{SYMR} + 1) * 4} - 1 = 3$$

   The total number of soft decisions transferred is $ELECNT * (FRMCNT + 1) * 2 = 32 * 4 * 2 = 256$, whereas the actual number soft decisions needed is **FL** = 250. Therefore, the above selection provides the optimal use of EDMA bandwidth.

VCP and EDMA configuration is shown in Table 9.

Table 9. VCP/EDMA Configuration for GSM − SD

Input Configuration			
F = 250	R = N/A	C = N/A	TB = Tailed
YAMT=0 (YAMEN=0)	IMAXS = 0x400	IMINS = 0	IMAXI = 0
SDHD = SD	SYMX = 15	SYMR = 15	
POLY[0] = 0x0C	POLY[1] = 0x0D	POLY[2] = 0x0	POLY[3] = 0x0
VCPXEVT Links		**VCPREVT links**	
Link 1		**Link 0**	
OPT: SUM = INC, DUM = FIXED		OPT: SUM = FIXED, DUM = DEC, TCINT = 1, TCC = VCPREVT	
SRC= `&bm[0]`		SRC = VCPDECS	
FRMCNT= 3	ELECNT= 32	FRMCNT= 3	ELECNT= 32
DST=VCPWBM		DST= `&sdhd[4*32–1]`	
FRMIDX= N/A	ELEIDX= N/A	FRMIDX= N/A	ELEIDX= N/A
ELERLD= N/A	LINK= NULL	ELERLD= N/A	LINK= NULL

6 Multichannel Operation Considerations

The coprocessor will typically be used in an operating environment where a series of frames is to be decoded in a most efficient manner. The efficiency could be with respect to one or more of the following parameters:

- **EDMA paRAM space:** Each frame of data requires a certain number of links in the paRAM, and due to the limited size of the paRAM, it may not be feasible to pre-program into paRAM all EDMA links for all frames to be decoded.
- **CPU interrupt rate:** The CPU intervention may be required to initialize input buffers for new frames to be decoded, process decoded frames, and program new EDMA links.
- **Percentage of coprocessor capabilities required:** If the coprocessor is used at maximum processing power, then frame decoding should be scheduled in such a manner as to keep the coprocessor constantly active, i.e., not let it wait for new input data to be transferred in, or decoded data to be transferred out.

In this section, we discuss several approaches to scheduling decoding for a series of frames. Each method optimizes one of the above mentioned parameters.

6.1 Method 1: paRAM-Efficient

This method is simple to program and requires the least number of links in the EDMA paRAM.

The paRAM usage and EDMA linking is shown Figure 7 .

Assuming that the coprocessor is initially idle (i.e., in “RESET” state), and appropriate EDMA channels and CPU interrupts are enabled, the suggested procedure is as follows:

1. CPU programs Link 0 for VCPXEVT and Link 0 for VCPREVT into the paRAM location corresponding to VCPXEVT and VCPREVT, respectively. The remaining links are programmed anywhere in the paRAM space. The transfers are terminated as follows:
   − The last link for VCPXEVT is linked to a NULL transfer, and does not generate CPU interrupt.
   − The last link for VCPREVT is also linked to a NULL transfer, and it generates CPU interrupt with TCC which correspond to VCPREVT.
2. The CPU sends “START” command to the VCP and continues any non-interfering processing.
3. When CPU receives EDMA interrupt, with TCC=VCPREVT, the CPU performs necessary input/output buffer management and repeats steps (1)−(2) for the next frame.

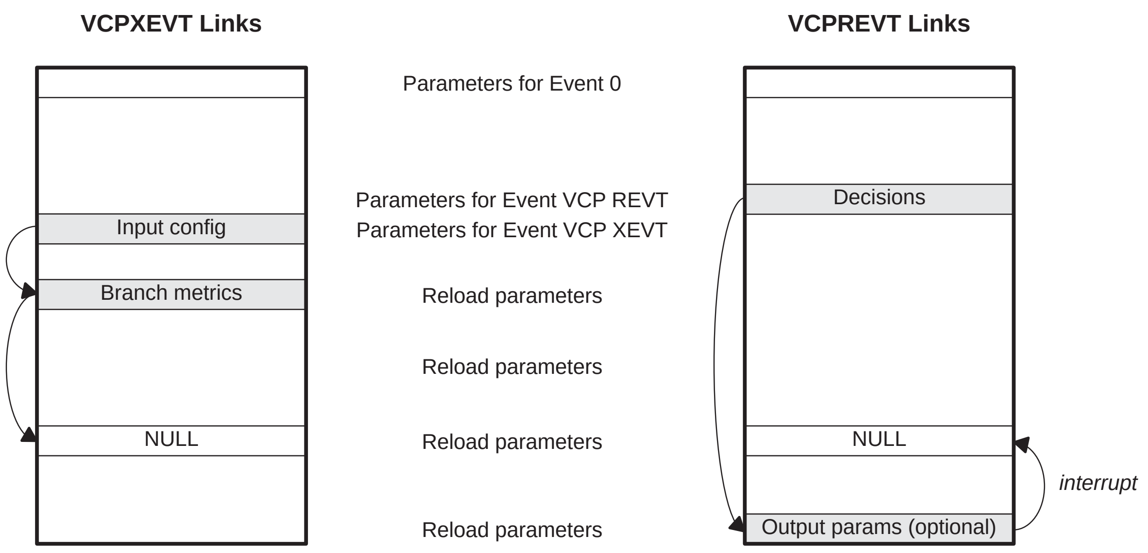


Figure 7. Method 1: paRAM Entries

6.2 Method 2: Continuous Decoding

The main problem with the approach outlined in section 6.1 is that the coprocessor is kept waiting for input data, i.e., decoding of a new frame does not start as soon as the coprocessor is ready for it.

This problem could be remedied by keeping EDMA links for two frames in the paRAM. The CPU is interrupted once frame #n has been decoded, in order to program links for frame #(n+2). Meanwhile, the coprocessor is processing frame #(n+1). This assumes that the TCP processing delay for frame #(n+1) is sufficiently large such that the CPU has enough time to respond to interrupt and write links for frame #(n+2) into the paRAM.

This concept is illustrated in Figure 8. The procedure is as follows:

1. The CPU programs all links for the first two frames. Note that the first VCPREVT and first VCPXEVT link for frame #1 are written into Event parameters.
2. The CPU sends “START” command to the VCP and continues any non-interfering processing
3. Once the transfer associated with link called “Output Params (optional) # 1,3,5,..” is completed, the CPU interrupt is generated. The CPU overwrites links associated with channel #1 with those associated with channel #3. Note that that the first VCPREVT and first VCPXEVT link for frame #3 are written into Reload parameters, since the Event parameter space is used by the link currently in progress.
4. Once the transfer associated with link called “Output Params (optional) # 2,4,6..” is completed, the CPU interrupt is generated. The CPU overwrites links associated with channel #2 with those associated with channel #4.
5. Steps (4) and (5) are repeated as long as there are frames to be processed. If the CPU gets interrupted and there are no additional frames to be processed, the last previously programmed link for both REVT and XEVT should be relinked to the NULL parameter set to terminate the transfer.

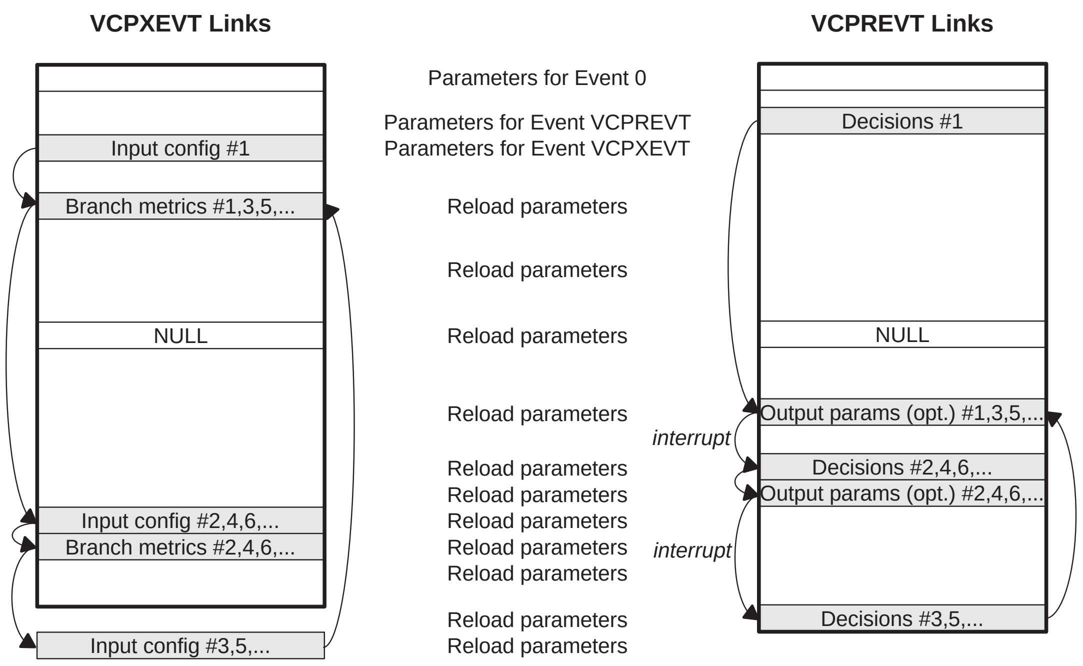


Figure 8. Method 2: paRAM Entries

The method could be expanded to more than two preprogrammed frames, provided there is space for additional links.

6.3 Method 3: Lowest CPU Interrupt Rate

In this method, we build on the idea of keeping the coprocessor continuously running. In addition to that, the process of EDMA link programming into paRAM is automated: instead of CPU pre−programming a small number of frames into paRAM, the CPU could program a large number of links and temporarily store them in L2 memory. The EDMA is then responsible for transferring preprogrammed links into paRAM.

For this scenario, two additional transfer parameters in the paRAM are required: one which copies links for even frames from L2 memory to paRAM (even copy), and another one which copies odd frames from L2 memory to paRAM (odd copy).

The “odd copy” #(2n-1) transfer is chained to the output parameter transfer for an odd frame #(2n-1). The transfer will be initiated immediately upon completion of output parameter #(2n-1) transfer. The output parameter #(2n−1) transfer will also link to decision #(2n) transfer, but the link will not be initiated until the synchronization event VCPREVT occurs, while the odd copy transfer, which is chained, will proceed without waiting for synchronization. The odd copy transfer will copy, into the paRAM reload zone, prepared transfer parameters for frame #(2n+1) and even copy #(2n) transfer. The even copy transfer should be copied into the paRAM location corresponding to Event K2. The above mentioned transfer parameters are copied as a part of the same EDMA transfer, as separate frames, and therefore it is required that the difference between the starting memory addresses of two consecutive frames be the same. For example, parameters for frame #(2n+1) and even copy #(2n) could be at consecutive locations in the paRAM. The odd copy transfer #(2n-1) needs to complete before the completion of output parameters #(2n) transfer. The first copied transfer which will be needed is even copy transfer

#(2n). To guarantee this, the copy transfer should be given a higher priority than the remaining transfers.

Similarly, the even copy #(2n) transfer is chained to the output parameter transfer for an even frame #(2n). The even copy transfer will copy prepared transfer parameters for frame #(2n+2) and odd copy #(2n+1). The odd copy transfer should be copied into the paRAM location corresponding to Event K1. The even copy transfer #(2n) needs to complete before the completion of output parameters #(2n). The first copied transfer which will be needed is odd copy transfer #(2n+1).

The scenario is illustrated in Figure 9.

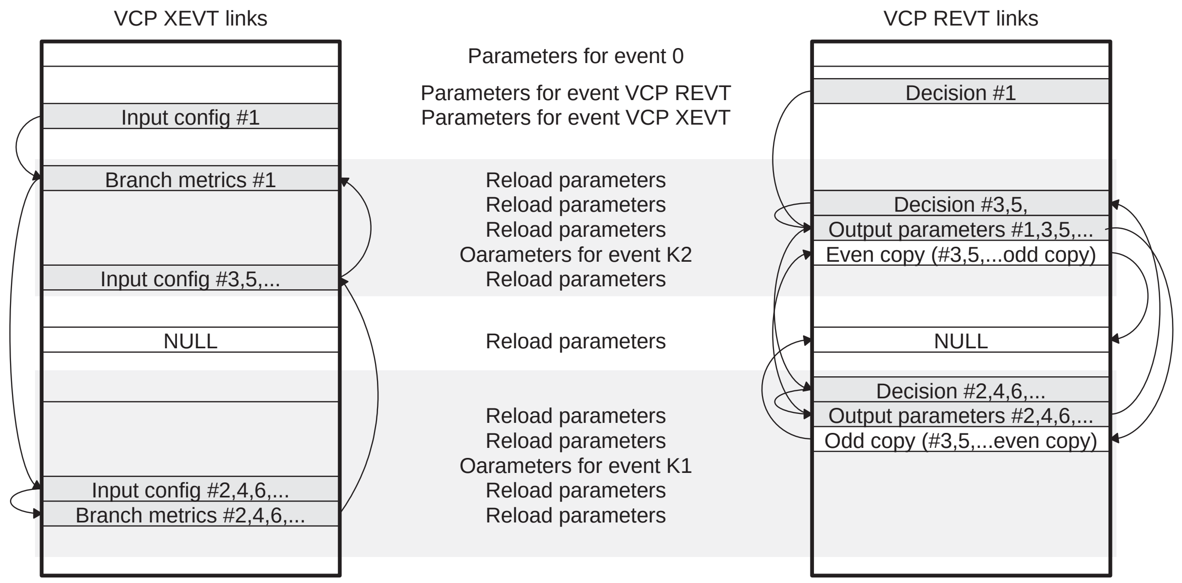


Figure 9. Method 3: paRAM Entries

7 References

1. *A. P. Hekstra*, “An alternative to Metric Rescaling in Viterbi Decoders,” *IEEE Transactions on Communications, vol 37, no 11, November 1989* , pp. 1220−1222.
2. *Viterbi Decoder Coprocessor User’s Guide* − Literature number SPRU533
3. *TMS320C6000 Peripherals User’s Guide* − Literature number SPRU190D
4. 3G TS 25 212 V3.1.0 (1999−12), *Multiplexing and channel coding (FDD)*
5. “Physical Layer Standard for cdma2000 Spread Spectrum Systems,” *TIA/EIA/IS-2000-2, prepared by TR45.5*.

Products		**Applications**	
Amplifiers	amplifier.ti.com	Audio	www.ti.com/audio
Data Converters	dataconverter.ti.com	Automotive	www.ti.com/automotive
DSP	dsp.ti.com	Broadband	www.ti.com/broadband
Interface	interface.ti.com	Digital Control	www.ti.com/digitalcontrol
Logic	logic.ti.com	Military	www.ti.com/military
Power Mgmt	power.ti.com	Optical Networking	www.ti.com/opticalnetwork
Microcontrollers	microcontroller.ti.com	Security	www.ti.com/security
		Telephony	www.ti.com/telephony
		Video & Imaging	www.ti.com/video
		Wireless	www.ti.com/wireless