

# MEMORY SUB-BANKING SCHEME FOR HIGH THROUGHPUT TURBO DECODER

Mayank Tiwari, Yuming Zhu and Chaitali Chakrabarti

Department of Electrical Engineering

Arizona State University, Tempe, AZ 85287

Email: {mayank, yuming, chaitali}@asu.edu

## ABSTRACT

Turbo codes have revolutionized the world of coding theory with their superior performance. However, the implementation of these codes is both computationally and memory-intensive. Recently, the sliding window (SW) approach has been proposed as an effective means of reducing the decoding delay as well as the memory requirements of Turbo implementations. In this paper, we present a sub-banked implementation of the SW-based approach that achieves high throughput, low decoding latency and reduced memory energy consumption. Our contributions include derivation of the optimal memory sub-banked structure for different SW configurations, study of the relationship between memory size, energy consumption and decoding latency for different SW configurations and study of the effect of number of sub-banks on the throughput and decoding latency of a given SW configuration. The theoretical study has been validated by SimpleScalar for a rate 1/3 MAP decoder.

## 1. INTRODUCTION

Turbo codes [1] have become very popular and have been adopted in mobile standards such as 3GPP for IMT-2000 and WCDMA for their superior error correcting capability. The superior performance is due to the combination of parallel concatenated coding and iterative decoding, which results in very high complexity. Innovations at the algorithmic and architectural level are needed in order to design low-power high-throughput Turbo decoders for mobile systems.

The Turbo decoding algorithm is both computation and memory intensive. In order to reduce the power consumption, several architectures have been proposed which implement approximate versions of the Turbo decoding algorithm [2, 3, 4]. In order to reduce the decoding latency and increase the throughput, the sliding window (SW) approach has been used in [5, 6, 7, 8, 9]. The SW approach has also been used to optimize the memory for MAP Turbo decoders in [8] and SISO-APP decoders in [9].

All of the existing work on Turbo decoder architectures assume a monolithic memory structure. However, memory sub-banking is an effective means of achieving high throughput, as was demonstrated in [10] for the Viterbi decoder. In this paper we present a novel memory sub-banking scheme for high throughput SW-based MAP Turbo decoder. Our contributions are as follows.

- Derivation of the optimal memory sub-banking structure (number and size of each sub-bank) that supports very high throughput and low decoding latency for different SW configurations.
- Study of energy consumption, decoding latency and memory size for different SW configurations.
- Study of the relationship between number of sub-banks, throughput and decoding latency for a given SW configuration.

This research was supported by the Consortium of Embedded and InterNetworking Technologies (CEINT) at ASU.

- Comparison of the energy consumption for different SW configurations when implemented on cache and sub-banked memory structures.

Such an analysis will aid the designer in choosing the optimal memory configuration given the constraints on coding performance, memory size, number of sub-banks, throughput, decoding latency and energy consumption.

The rest of the paper is organized as follows. Section 2 gives a brief description of MAP algorithm and application of sliding window approach. Section 3 describes the proposed optimal sub-banking structure. Section 4 presents through simulations the trade-offs between throughput, memory size, decoding latency and energy consumption. Section 5 concludes the paper.

## 2. THE MAP ALGORITHM WITH SW APPROACH

Fig. 1 shows the diagram of a Turbo encoder and decoder. The Turbo encoder consists of two recursive systematic convolutional (RSC) encoders and a random interleaver. The Turbo decoder consists of two SISO decoders (corresponding to the two component RSC encoders), an interleaver and a de-interleaver placed between the two decoders. The first SISO decoder generates soft outputs, which are interleaved and used to produce an improved estimate of the a priori probabilities of the information sequence for the second decoder. Similarly, the output of the second SISO decoder is fed to the first SISO decoder through de-interleaver.

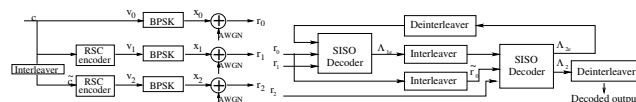


Fig. 1. Basic Turbo encoder and iterative decoder

In the Turbo decoder, the MAP algorithm minimizes symbol (or bit) error probability. For each transmitted symbol, it generates a hard estimate and a soft output in the form of a *posteriori probability* (APP) based on the received sequence [11]. The log-likelihood  $\Lambda(c_t)$  represents the soft output of the MAP decoder. It can be used as an input to another decoder in a concatenated scheme or in the next iteration in an iterative decoder. In the final operation, the decoder makes a hard decision by comparing  $\Lambda(c_t)$  to a threshold equal to zero.  $\Lambda(c_t)$  is calculated in terms of  $\alpha$ ,  $\beta$ , and  $\gamma$ , where  $\alpha$  is the forward path metrics,  $\beta$  is the backward path metrics, and  $\gamma$  is the branch metrics.

$$\Lambda(c_t) = \log \frac{\sum_{(l', l) \in B_t^1} \alpha_{t-1}(l') \gamma_t^1(l', l) \beta_t(l)}{\sum_{(l', l) \in B_t^0} \alpha_{t-1}(l') \gamma_t^0(l', l) \beta_t(l)} \quad (1)$$

where  $l'$  is the state at time  $t-1$ ,  $l$  is the state at time  $t$ ,  $B_t^i$ ,  $i = 0, 1$  is the set of transitions  $S_{t-1} = l' \rightarrow S_t = l$  that are caused by input  $c_t = i$ . The forward recursion parameter  $\alpha$  and the backward recursion

parameter  $\beta$  can be defined as

$$\alpha_t(l) = \sum_{l'=0}^{M_s-1} \alpha_{t-1}(l') \cdot \sum_{i \in (0,1)} \gamma_t^i(l', l), \quad t = 1, 2, \dots, \tau \quad (2)$$

$$\beta_t(l) = \sum_{l'=0}^{M_s-1} \beta_{t+1}(l') \cdot \sum_{i \in (0,1)} \gamma_{t+1}^i(l', l), \quad t = \tau - 1, \dots, 0 \quad (3)$$

where  $M_s$  is the number of states. The boundary conditions for  $\alpha$  are  $\alpha_0(0) = 1$  and  $\alpha_0(l) = 0$  for  $l \neq 0$  and for  $\beta$  are  $\beta_\tau(0) = 1$  and  $\beta_\tau(l) = 0$  for  $l \neq 0$ .  $\gamma$  is the branch metric parameter.

In the standard MAP algorithm, the decision delay is equal to the received frame size. For large frames, the memory required for the decoder implementation is also excessive since the forward and backward recursions are performed on the whole frame. For a frame of size  $N$ , the decoder needs to store  $N$  survivors and  $N$  survivor path metrics during both the forward and backward recursion calculations.

The sliding window approach partly eliminates the large storage problem of the Turbo decoder by initializing the  $\alpha$  or  $\beta$  metrics at an intermediate point instead of at the end of the frame. Assume that the required decision depth for a single component convolutional code is  $L$ . The decoder then does not need to start from the initial node; it can start at any node and after depth  $L$ , the decision is likely to be made as reliably as starting from the initial node.

The sliding window approach can be applied to (1) either  $\alpha$  or  $\beta$  metric calculation referred to as the single flow structure and (2) both  $\alpha$  and  $\beta$  metric calculation referred to as the double flow structure [8]. Let the input data frame of length  $N$  be divided into blocks  $\{D_i\}$ . Consider the case when  $\beta$  metric calculation is done by sliding window approach and  $\alpha$  calculations are done in regular way as shown in Fig. 2. Define  $\eta$  as the ratio of actual metrics calculated per dummy metric;  $\eta = \frac{p}{q}$ , where  $p$  and  $q$  are integers and  $\frac{p}{q}$  is an irreducible rational number. We divide the input data into blocks such that the actual and dummy metric calculations are performed on one or more blocks. If  $L$  is the decision depth required for reliable metric calculation using the sliding window approach, then the number of actual calculations for  $L$  dummy calculations is  $\eta L$  (by definition). To make both  $L$  and  $\eta L = p \frac{L}{q}$  a multiple of block size, each block  $\{D_i\}$  is of size  $\frac{L}{q}$ .

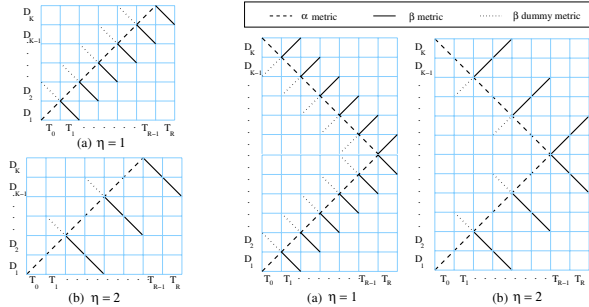


Fig. 2. Single flow structure

Fig. 3. Double flow structure

The path metric calculations and extrinsic output calculations can be made parallel by carefully selecting which operations are done on which blocks. In Fig. 2,  $D_i$  represents one block of input data of size  $\frac{L}{q}$ , and  $T_j$  represent a time slot which is defined as the time required for processing a block of data.  $\beta_{i,d}$  is the dummy  $\beta$  calculation done on  $D_{i+1}$  and is required for calculating  $\beta_i$ . Fig. 2(a) represents the case for  $\eta = 1$ , where the number of dummy  $\beta$  calculations is equal to the actual number of  $\beta$  calculations and Fig. 2(b) shows the structure for  $\eta = 2$ , where the number of actual  $\beta$  calculations is double the number of dummy  $\beta$  calculations.

To determine the metric calculations and extrinsic output that are done simultaneously, select a time slot  $T_j$ . For  $\eta = 1$  (see Fig. 2(a)), in time slot  $T_1$ , the calculation of  $\beta_{2,d}$  is done on block  $D_3$ ,  $\alpha_2$  calculations on block  $D_2$ , and  $\beta_1$  and  $Ex_1$  calculations on block  $D_1$ . Thus

3 processing units are required to work on different  $D_i$ 's. Clearly, the blocks that will be operated on in parallel will depend on the value of  $\eta$ .

For the double flow structure, assuming the number of processing units is twice as large, we can determine the path metric calculations and extrinsic output calculations that are done simultaneously. In the double flow structure shown in Fig. 3, sliding window is applied on  $\beta$  metrics calculation on lower half of the received data (blocks  $D_0, D_1, \dots, D_{\frac{K}{2}-1}$ ) and on  $\alpha$  metrics calculation on upper half of the received data (block  $D_{\frac{K}{2}}, \dots, D_{K-1}$ ). For  $\eta = 1$  (see Fig. 3(a)), in time slot  $T_1$ , the calculation of  $\beta_{2,d}$  is done on block  $D_3$ ,  $\alpha_2$  on block  $D_2$ ,  $\beta_1$  and  $Ex_1$  on block  $D_1$ ,  $\alpha_{K-1,d}$  on block  $D_{K-2}$ ,  $\beta_{K-1}$  on block  $D_{K-1}$ ,  $\beta_K$  and  $Ex_K$  on block  $D_K$ .

### 3. SUB-BANKED STRUCTURE FOR TURBO DECODERS

For a given  $\eta$ , the choice of memory configuration (number and size of banks) affects the throughput. Our aim is to design the *optimal* memory configuration for an architecture that achieves maximum throughput. Maximum throughput corresponds to the case where the output generation rate is equal to the maximum possible input data rate, and there is no need to store the input data. The optimal memory configuration is one that requires minimum total memory and minimum number of single port memory sub-banks. We choose single port sub-banks since it is cheaper both in terms of access time and energy compared to multi-port structures. All throughput calculations have been normalized with respect to the throughput of the Turbo decoder using optimal sub-banked structure for  $\eta = 1$ .

Time	Data Memory					α Memory			Dummy	Output
	0	1	2	3	4	0	1	2		
0	-D <sub>1</sub>	D <sub>2</sub>	+D <sub>3</sub>	·	·	+α <sub>1</sub>	·	·	·	·
1	D <sub>1</sub>	-D <sub>2</sub>	-D <sub>3</sub>	+D <sub>4</sub>	·	α <sub>1</sub>	+α <sub>2</sub>	·	β <sub>2,d</sub>	·
2	D <sub>1</sub>	-D <sub>2</sub>	-D <sub>3</sub>	D <sub>4</sub>	+D <sub>5</sub>	α <sub>1</sub>	-α <sub>2</sub>	+α <sub>3</sub>	·	β <sub>2</sub> , Ex <sub>2</sub>
3	-D <sub>1</sub>	+D <sub>2</sub>	D <sub>3</sub>	-D <sub>4</sub>	-D <sub>5</sub>	-α <sub>1</sub>	+α <sub>2</sub>	α <sub>3</sub>	β <sub>3,d</sub>	β <sub>3</sub> , Ex <sub>3</sub>
4	+D <sub>7</sub>	D <sub>6</sub>	D <sub>5</sub>	-D <sub>4</sub>	-D <sub>5</sub>	+α <sub>5</sub>	-α <sub>4</sub>	α <sub>3</sub>	·	β <sub>4</sub> , Ex <sub>4</sub>
5	-D <sub>7</sub>	-D <sub>6</sub>	-D <sub>5</sub>	+D <sub>4</sub>	D <sub>5</sub>	α <sub>5</sub>	+α <sub>6</sub>	-α <sub>5</sub>	β <sub>5,d</sub>	β <sub>5</sub> , Ex <sub>5</sub>
6	-D <sub>7</sub>	-D <sub>6</sub>	+D <sub>5</sub>	D <sub>4</sub>	D <sub>5</sub>	α <sub>5</sub>	-α <sub>6</sub>	+α <sub>7</sub>	·	β <sub>6</sub> , Ex <sub>6</sub>
7	D <sub>7</sub>	+D <sub>10</sub>	-D <sub>9</sub>	-D <sub>8</sub>	-D <sub>9</sub>	-α <sub>5</sub>	+α <sub>6</sub>	α <sub>7</sub>	β <sub>7,d</sub>	β <sub>7</sub> , Ex <sub>7</sub>

Fig. 4. Optimal memory layout for  $\eta = 2$

#### 3.1. Optimal sub-banking structure for $\eta = \frac{p}{q}$

We assume that each data block of size  $\frac{L}{q}$  is stored in one sub-bank. For a frame length of  $N$ , we have  $K = \frac{Nq}{L}$  such data blocks. In order to do  $L$  dummy  $\beta$  metric calculations,  $q$  sub-banks are accessed. There are  $p$  actual  $\beta$  calculations per  $q$  dummy  $\beta$  calculations. Thus for  $\alpha$  and  $\beta$  metric calculations of  $p$  blocks,  $p$  sub-banks are accessed.

The  $\beta$  metric calculations starts (in reverse order) after  $\alpha$  metric calculations finishes. Thus we store the  $\alpha$  metrics in  $p$  sub-banks for the  $\beta$  calculations. One more sub-bank is needed to store the newly calculated  $\alpha$  metrics (since it is calculated continuously). Thus a total of  $U_\alpha = (p + 1)$  sub-banks are required for  $\alpha$  metric calculations and storage. The number of sub-banks to store data is  $U_{data} = p + 2q + 1$ . This has been derived by mathematically analyzing the lifetime of the blocks. The derivation has been omitted due to lack of space.

**Total memory:** Since the data memory sub-bank has to store two incoming state metrics, and the  $\alpha$  memory sub-bank only needs to store one value, for word length  $W_i$ , the  $\alpha$  memory is of size  $U_\alpha \times W_i = \frac{L}{q}(p + 1)W_i$  and the data memory is of size  $U_{data} \times 2W_i = \frac{L}{q}(p + 2q + 1)2W_i$ . The total memory size is given by

$$Total\ memory = \frac{(3p + 4q + 3)}{q} \cdot L \cdot W_i \quad (4)$$

**Graphical representation:** The number of sub-banks and the sub-bank size is shown in Fig. 5 and Fig. 6 respectively. Note that the sub-bank size for all  $\eta$  that have the same  $q$  is the same. Thus the sub-bank size for  $\eta = \frac{1}{2}, \frac{3}{2}, \frac{5}{2}, \dots$  is the same and is half of the sub-bank

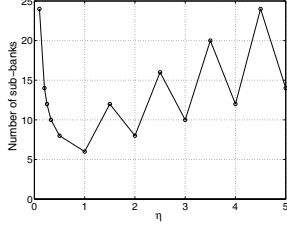


Fig. 5. Optimal number of sub-banks

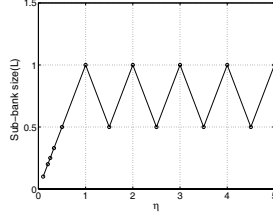


Fig. 6. Sub-bank size for optimal configuration

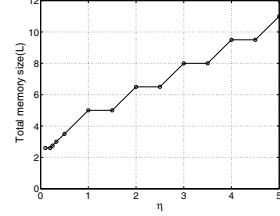


Fig. 7. Total memory size for optimal configuration

size for  $\eta = 1, 2, 3, \dots$ . The total memory size required for optimal configuration is shown in Fig. 7. The total memory increases as  $\eta$  increases since it is a function of the number of sub-banks and the sub-bank size. However, the memory size for  $\eta = 1$  and  $\eta = \frac{3}{2}$  or  $\eta = 2$  and  $\eta = \frac{5}{2}$  is the same. The memory size for  $\eta = 1$  and  $\eta = \frac{3}{2}$  is same because the sub-bank size for  $\eta = 1$  is double the sub-bank size required for  $\eta = \frac{3}{2}$ , and the total number of sub-banks required for  $\eta = 1$  is half of the number of sub-banks required for  $\eta = \frac{3}{2}$ .

### 3.2. Data flow for the optimal sub-banked structure for $\eta = 2$

Fig. 4 shows the dataflow of the optimal sub-bank structure for single flow with  $\eta = 2$ . The optimal configuration consists of 8 memory sub-banks: 5 sub-banks are needed for storing input data (shown under data memory column) and 3 sub-banks are needed for  $\alpha$  metrics (shown under  $\alpha$  memory column). The block  $D_i$  represents input data block being stored in the memory at time slot  $T_j$ .  $\alpha_i$  represents forward recursion metrics calculated on data block  $D_i$ .  $\beta_{i,d}$  represents the dummy  $\beta$  calculations required for backward recursion metrics  $\beta_i$  and are done on block  $D_{i+1}$ . As soon as  $\beta_{i,d}$  calculation is finished,  $\beta_i$  metrics is calculated on sub-bank  $D_i$ .  $Ex$  for sub-bank  $D_i$  is defined as  $Ex_i$  and is calculated using  $\alpha_i$  and  $\beta_i$ . We do not require any sub-bank for storing  $\beta_{i,d}$  because we are only concerned with the last dummy metric value and not the intermediate ones. There is also no need to store  $\beta_i$  as it is immediately used up for the calculation of  $Ex_i$ . As the extrinsic information for a sub-bank  $i$  is finished,  $\alpha$  metric for that sub-bank,  $\alpha_i$ , is used up completely. As soon as any data or  $\alpha$  values are used, they become obsolete and are replaced by the new values. A “+” sign show a write pointer while a “-” sign shows a read pointer in Fig. 4. The total number of  $\alpha$  and  $\beta$  calculations are same as the input data size.  $\beta$  dummy calculations are done in every other time slot.

The dataflow is pipelined: an output is generated for every input after an initial latency. In Fig. 4, consider the scenario where  $D_1$  and  $D_2$  have already been loaded into sub-bank 0 and 1. At time  $T_0$ ,  $\alpha_1$  is computed for  $D_1$ . At the same time,  $D_3$  is read into sub-bank 2 for future processing. At time  $T_1$ ,  $\alpha_2$  is computed for  $D_2$ ,  $\beta_{2,d}$  is calculated on  $D_3$ , and input data  $D_4$  is read into sub-bank 3. At the end of  $T_1$ , we have sufficient reliable metrics to start calculating  $\beta_2$  and generating output  $Ex_2$ . So, the decoding latency for optimal configuration of  $\eta = 2$  is 2 time slots ( $T_0$  and  $T_1$ ). Starting from  $T_2$ , we generate an output for every input data. After  $Ex_2$  is generated,  $\alpha_2$  and  $D_2$  are no longer necessary and the new input data block  $D_6$  replaces  $D_2$  in sub-bank 1 in time slot  $T_3$ . Since an output is generated in every cycle after  $T_2$ , there is always a sub-bank whose data can be replaced. Thus no additional sub-banks are needed and 5 sub-banks are sufficient to store the input data.

## 4. SIMULATION RESULTS AND TRADEOFF ANALYSIS

### 4.1. Simulation setup

A rate  $\frac{1}{3}$  MAP Turbo decoder with an interleaver size  $N = 1200$  was considered in our simulation. Decision depth,  $L$ , was taken to be 60. Each stage contains 4 states and 2 branch metric paths converging to each state. The sub-banked structures were simulated using SimpleScalar [12] to obtain the number of accesses to each bank. The throughput and decoding latency values that were obtained theoretically were also verified using SimpleScalar. To compare the energy reduction

using the sub-banked structure, the monolithic cache memory structure was simulated using SimpleScalar based Watch tool [13].

### 4.2. Throughput and decoding latency for optimal configuration

Fig. 8 shows the throughput and decoding latency as a function of  $\eta$ . Note that the throughput is always 1 irrespective of the value of  $\eta$ . This is by definition; the sub-banked memory structure was designed such a way to achieve throughput of 1 in all cases.

The decoding latency, defined as the time lag between the start of  $\alpha$  metric calculation and extrinsic information calculation, increases with  $\eta$ . This is because as  $\eta$  increases, dummy  $\beta$  calculations start late and  $\alpha$  metrics are stored for longer time duration. The decoding latency is also affected if the number of sub-banks is smaller than that of the optimal configuration: it either remains same or increases with decrease in the number of sub-banks.

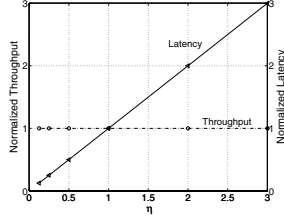
### 4.3. Tradeoff analysis

For a given a value of  $\eta$ , we have derived the optimal number of sub-banks corresponding to which the memory requirement is the lowest and the throughput (normalized) is 1 in section 3.1. In this section, we evaluate the relationship between total memory size, number of sub-banks and throughput. For example, for  $\eta = 2$ , the optimal number of sub-banks is 8. But the number of sub-banks can be reduced to 6 without decreasing the throughput. However, reduction in the number of sub-banks comes with an increase in the size of each sub-bank. As a result, the total memory size increases to approximately 1.5 times the optimal configuration (see Fig. 9). The total number of sub-banks can be further reduced to 5 by keeping the same sub-bank size (decreasing the total memory requirement). But this comes at the expense of the throughput dropping by 1.5 times. If the number of sub-banks is further reduced (can be made as low as 1), either the total memory size increases or the throughput decreases. Fig. 9 and Fig. 11 shows the total memory size and decoding latency respectively as a function of the number of sub-banks for  $\eta = 2$ . Fig. 10 describes the throughput for  $\eta = 2$ . Note that the throughput decreases to less than  $\frac{1}{2}$  if the number of sub-banks is 3 or lower.

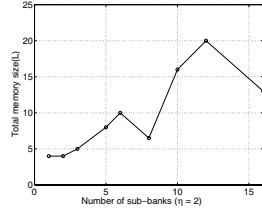
If the number of available sub-banks is larger than the optimal number, a higher throughput can be achieved by applying the double flow structure [8]. Take the example for  $\eta = 2$ ; a  $16 = (2 \times 8)$  sub-banked structure is the optimal double flow structure that gives double throughput (compared to the optimal single flow structure). In fact, for  $\eta = 2$ , we can get double throughput as long as the number of sub-banks is greater than or equal to 12. The total memory increases if the number of sub-banks are reduced from 16 to 12 (see Fig. 9).

### 4.4. Energy consumption

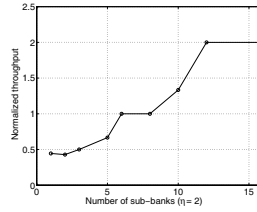
Next, we evaluate the energy consumption of a cache based memory structure and sub-banked memory structure for different values of  $\eta$ . Using SimpleScalar based Watch tool, we get the energy consumption profile of the MAP decoder for the following cache memory structure: L-1 data cache size 128:32:4 (128 is the number of sets, 32 bytes is the line size and 4 is set associativity). L-2 data cache 1024:64:1, L-1 instruction cache 512:32:1 and L-2 instruction cache similar to the L-2 data cache. Figure 12 describes the memory energy, datapath energy and total energy for different values of  $\eta$ . As  $\eta$  increases, the total energy, memory energy and datapath energy decreases proportionally.



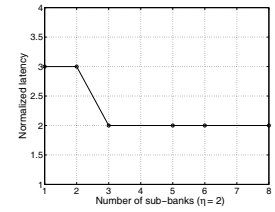
**Fig. 8.** Throughput and decoding latency for optimal configuration



**Fig. 9.** Total memory vs. number of sub-banks ( $\eta = 2$ )



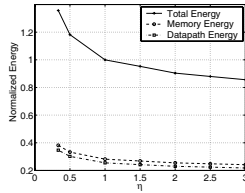
**Fig. 10.** Throughput vs. number of sub-banks ( $\eta = 2$ )



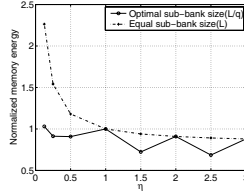
**Fig. 11.** Decoding latency vs. number of sub-banks ( $\eta = 2$ )

This is because as  $\eta$  increases, the number of dummy calculations decreases, and the number of total calculations decreases.

Fig. 13 shows the energy consumption of the optimal sub-banked memory structure as a function of  $\eta$ . The memory energy model used assumes that as the sub-bank size doubles, the access energy increases by 30% [14]. The trend of the energy for sub-banked structure will be same as Fig. 12 if we use equal sub-bank of size  $L$  as shown by dotted line in Fig. 13. For  $\eta = \frac{L}{q}$ , the size of each data sub-bank is  $\frac{L \cdot W}{q}$  and therefore, the memory access energy decreases as  $q$  increases. The memory energy for  $\eta = \frac{1}{2}, \frac{3}{2}, \frac{5}{2}$  is therefore less than the memory energy for  $\eta = 1, 2, 3$ . As  $\eta$  increases, the number of dummy metric calculation decreases. Thus for  $\eta \geq 1$ , the memory energy for  $\eta = 2$  is lower than  $\eta = 1$ , the memory energy of  $\eta = \frac{3}{2}$  is lower than  $\eta = \frac{5}{2}$  etc. For  $\eta < 1$ , the number of dummy metric calculation increases and memory access energy decreases. So the total memory energy does not vary to a large extent.



**Fig. 12.** Normalized energy for cache memory structure (Wattch)



**Fig. 13.** Normalized energy for sub-banked memory structure

The number of accesses in both the cache and sub-banked structure is same for a specific value of  $\eta$ . Since the sub-bank memory size is smaller than the cache memory size, the total memory energy for sub-banked structure will be significantly less than the cache structure.

By using sub-banked structure, we have demonstrated that as  $\eta$  increases, the memory energy decreases while maintaining the normalized throughput = 1. This is quite different from [8], where as  $\eta$  increases, normalized throughput decreases and energy increases. We believe that the difference is because of the memory configuration: implementation of single port sub-banked memory structures here versus monolithic memory in [8].

## 5. CONCLUSION

The aim of this work was to derive the optimal memory sub-bank structure for the Turbo decoder that achieves maximum throughput. The optimal structure for any  $\eta$  was derived theoretically and verified using SimpleScalar. The relation between memory size, throughput, decoding latency and energy was also studied.

The value of  $\eta$  is typically determined from the coding performance requirement of the Turbo code since as  $\eta$  increases, the performance degrades. The value of  $\eta$  can also be determined from the hardware specification. For instance, if the number of sub-banks is given, or the total memory size is given, we can choose  $\eta$  from Fig. 5 - Fig. 7. The choice is however complicated by the fact that as  $\eta$  increases, the decoding latency increases and the memory energy decreases. Our analysis helps

determine the optimal value of  $\eta$  by balancing the hardware, latency and energy requirements and then determining the optimal sub-bank structure for that value of  $\eta$ .

## 6. REFERENCES

- [1] C. Berrou, A. Glavieux and P. Thitimajshima, "Near Shannon limit error-correcting coding and decoding," *Proceedings of International Conference on Communications*, pp. 1064–1070, 1993.
- [2] J. Kaza and C. Chakrabarti, "Energy-efficient Turbo decoder," *IEEE International Conference on Acoustics, Speech, and Signal Processing*, vol. 3, pp. 3093–3096, 2002.
- [3] D. Garrett, B. Xu and C. nicol, "Energy Efficient Turbo Decoding for 3-G mobile," *IEEE Proceedings of International Symposium on Low Power Electronics Design*, pp. 328–333, 2001.
- [4] S.Hong and W.E.Stark, "Design and implementation of a low complexity VLSI Turbo-code decoder architecture for low energy mobile wireless communications," *Journal of VLSI Signal Processing Systems 24*, pp. 43–57, 2000.
- [5] Z.Wang, Z.Chi, and K.K.Parhi, "Area-efficient high-speed decoding schemes for Turbo decoders," *IEEE Transaction on Very Large Scale Integration (VLSI) Systems*, vol. 10, no. 6, pp. 902–912, Dec. 2002.
- [6] M. J. Thul, F. Gilbert, T. Vogt, G. Kreiselmair, and N. Wehn, "A scalable system architecture for high-throughput Turbo-decoders," *IEEE Workshop on Signal Processing Systems*, pp. 152–158, 2002.
- [7] S. Lee, N.R. Shanbhag, and A.C. Singer, "A low-power VLSI architecture for Turbo decoding," *IEEE Proceedings of International Symposium on Low Power Electronics Design*, pp. 366–371, 2003.
- [8] C. Schurgers, F. Catthoor, and M. Engels, "Memory optimization of MAP Turbo decoder algorithms," *IEEE Transaction on Very Large Scale Integration (VLSI) Systems*, vol. 9, no. 2, pp. 305–312, Apr. 2001.
- [9] M. M. Mansour and N. R. Shanbhag, "VLSI architecture for SISO-APP decoders," *IEEE Transaction on Very Large Scale Integration (VLSI) Systems*, vol. 11, no. 4, pp. 627–650, Aug. 2003.
- [10] R. Cypher and C. B. Shung, "Generalized trace-back technique for survivor memory management in the Viterbi algorithm," *Journal of VLSI signal Processing*, pp. 85–94, 1993.
- [11] B. Vucetic and J. Yuan, *Turbo codes, principles and applications*, Kluwer Academic Publishers, 2000.
- [12] T.Austin, E.Larson, D.Ernst, "SimpleScalar: An infrastructure for computer system modeling," *IEEE Computer*, vol. 35, no. 2, pp. 59–67, Feb. 2002.
- [13] D. Brooks, V. Tiwari and M. Martonosi, "Wattch: a frame work for architectural power analysis and optimizations," *International Symposium on Computer Architecture*, pp. 83–94, 2000.
- [14] V. Deleluz, M. Kandemir, N. Vijaykrishnan, M. J. Irwin, A. Sivasubramaniam, and I. Kolcu, "Compiler-directed array interleaving for reducing energy in multi-bank memories," *15th International Conference on VLSI Design Proceedings*, pp. 288–293, 2002.