Joint Source-Channel Decoding ASIP Architecture for Sensor Networks*

Pablo Ituero[1], Gorka Landaburu[2], Javier Del Ser[3], Marisa López-Vallejo[1], Pedro M. Crespo[3], Vicente Atxa[2], and Jon Altuna[2]

[1] ETSIT, Universidad Politécnica de Madrid
{pituero,marisa}@die.upm.es
[2] Mondragon Unibertsitatea
{glandaburu,batxa,jaltuna}@eps.mondragon.edu
[3] CEIT and TECNUN, University of Navarra
{jdelser,pcrespo}@ceit.es

Abstract. In a sensor network, exploiting the correlation among different sources allows a significant reduction of the transmitted energy at the cost of a complex decoder scheme. This paper introduces the first hardware implementation for joint source-channel decoding of correlated sources. Specifically, a dual-clustered VLIW processor with a highly optimized datapath is presented.

Keywords: ASIP, DSC, Factor Graphs, Joint Source-Channel Coding, Sensor Networks, Turbo Codes, VLIW.

1 Introduction

During the last decade the research interest for sensor networks has risen sharply in the scientific community. Such networks consist of densely deployed sensors spread across a certain geographical area. Data is transmitted to either neighboring nodes or a common destination, which collects the information from all the existing sensors. This work focuses on the latter scenario, i.e. a centralized sensor network.

In this context, the high density of nodes in these networks and, consequently, their physical proximity may incur the appearance of correlation among the data collected by the different sensors. This correlation can be exploited, for instance, to reduce the required transmitted energy for a certain level of performance and complexity. Even if there is no communication among sources, the Slepian and Wolf Theorem reveals that distributed compression can be performed as long as the decoder is aware of the correlation among the sources.

However, when the channels from the sensors to the common receiver are noisy, each node has to add controlled redundancy (equivalently, apply forward

* This work was funded by the projects OPTIMA (TEC2006-00739) and MIMESIS (TEC2004-06451-C05-04/TCM) of the Spanish Ministry of Education and Science and TECMAR (IBA/PI2004-3) of the Basque Government Ministry of Industry.

error correcting or *channel coding* techniques) to the transmitted data in order to combat the errors introduced by the channel. Under these circumstances, the separate design of source (compression) and channel coding schemes is an optimal solution whenever the complexity is not an issue (Separation Theorem).

The sensors of these networks are examples of complex embedded systems. Small, low-power, low-cost transceiver, sensing and processing units need to be designed. Actually, communication becomes one of the most power and area consuming elements in the hardware of a sensor network node. Thus, efficient hardware strategies are essential to obtain a good implementation. On the one hand, the power hungry general purpose processors do not satisfy the constraints given by such wireless systems. On the other hand, the high cost and low design productivity of new submicron technologies have turned ASIC designs into extremely expensive solutions. Therefore, special architectures that match the architecture to the application are required. Application Specific Instruction Set Processors (ASIPs) provide an attractive solution to this kind of problems: First, several applications can be run on it, and different generations of the same application are allowed. Second, for the application developer that uses an ASIP instead of an ASIC the time to market is reduced, it is cheaper and there is lower risk. Furthermore, the power overhead related to programmability (standard processors) can be mitigated by ASIPs architectures, especially if they are very dedicated.

In this paper we present a novel application specific processor architecture for centralized sensor networks. Specifically, we present a dual-clustered VLIW processor that implements a joint source-channel SISO decoder within an iterative scheme. To the best of our knowledge, the ASIP described here, based on [1], is the first hardware architecture for distributed coding of correlated sources. A customized datapath allows very high efficiency while providing enough flexibility to execute all the operations involved in the joint source-channel turbo decoding.

The remainder of the paper is outlined as follows. In the next section, an overview of the source data generating process is provided. Section 3 analyzes the proposed iterative joint source-channel turbo-decoder ASIP structure. In Sect. 4 the most important results of the design are presented. Finally Sect. 5 draws some concluding remarks.

2 State of the Art and Signal Processing Foundations

The research activity on the potential of the Slepian-Wolf Theorem for DSC (Distributed Source Coding), through noisy communication networks has gravitated around two different scenarios. On the one hand, for an asymmetric scenario consisting of two nodes (i.e. when only one of the channels is noisy), the authors in [2] proposed a joint source-channel coding scheme for a BSC channel based on turbo codes. Using Irregular Repeat Accumulate codes, the authors in [3] followed the same approach for BSC, AWGN and Rayleigh fading channels. On the other hand, the case of symmetric (i.e. both channels are noisy) joint

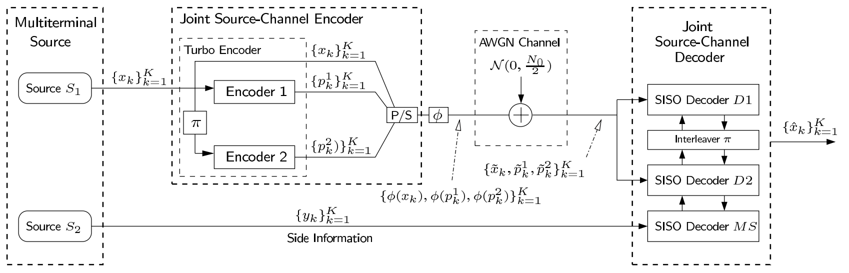

Fig. 1. Turbo joint source-channel decoding scheme with side information from [7]

source-channel coding with AWGN channels has been analyzed using turbo [4] and Low Density Generator Matrix [5] codes. Joint turbo equalization and decoding in asymmetric scenarios with correlated sources was also proposed in [6]. In [7] it was shown that the correlation parameters can be iteratively estimated at the decoder.

Following the system proposed in [7], the communication setup for the proposed VLIW based decoder is shown in Fig. 1. The source block $\{X_k\}_{k=1}^M$ produced by source S_1 is interleaved with a permutation function τ, and then processed by a binary rate $R_j = 1/3$ turbo code [8] with two identical constituent convolutional encoders C_1 and C_2 of constraint length L, giving rise to a set of $S^c = 2^{L-1}$ possible states. The encoded binary sequence is denoted by $\{X_k, P_k^1, P_k^2\}_{k=1}^K$, where we assume that P_k^1 and P_k^2 are the redundant symbols produced for input bit X_k by C_1, C_2, respectively. The input to the AWGN channel (with average energy per symbol E_c and noise variance $\sigma^2 = N_0/2$) is denoted as $\{\phi(X_k), \phi(P_k^1), \phi(P_k^2)\}_{k=1}^K$, where $\phi : \{0,1\} \rightarrow \{-\sqrt{E_c}, +\sqrt{E_c}\}$ denotes the BPSK transformation performed by the modulator. Finally, the received corresponding sequence will be denoted by $\{\tilde{X}_k, \tilde{P}_k^1, \tilde{P}_k^2\}_{k=1}^K$. The main idea behind the scheme in Fig. 1 hinges on the fact that, in order to estimate the original source symbols as $\{\hat{X}_k\}_{k=1}^K$, the joint decoder utilizes, not only the controlled redundancy $\{P_k^1, P_k^2\}_{k=1}^K$ added by the encoding stage, but also the side information $\{Y_k\}_{k=1}^K$ available at the decoder. A brief explanation of the HMM (Hidden Markov Model) based multiterminal source model and the designed decoding algorithm followed in [7] is next provided.

2.1 Adopted Model for the Correlated Multiterminal Source

As stated previously, we consider the general case where the random variable pairs (X_i, Y_i) and (X_j, Y_j) $\forall\, i \neq j$ at the output of the multiterminal source are not independent, i.e. the correlation among the sources represented by such a multiterminal source has memory. This class of multiterminal source may model, for instance, either temperature monitoring or video capture sensors. To emphasize the distributed nature of such modeled sources we will further assume that the random processes $\{X_k\}$ and $\{Y_k\}$ are binary, i.i.d.[1] and equiprobable.

[1] The term *i.i.d.* stands for *independent and identically distributed.*

The following generation model fulfills these requirements: the first component $\{X_k\}_{k=1}^K$ is a sequence of i.i.d. equiprobable binary random variables, while the second component $\{Y_k\}_{k=1}^{\infty}$ is produced by bitwise modulus-2 addition (hereafter denoted $\oplus$) of X_k and E_k, where $\{E_k\}$ is a binary stationary random process generated by a HMM. This model has often been utilized in the related literature [9,10]. The HMM is characterized by the set of parameters $\lambda_{HMM} = \{S_\lambda, \underline{A}, \underline{B}, \underline{\Pi}\}$, where S_λ is the number of states, $\underline{A} = [a_{s',s}]$ is a $S_\lambda \times S_\lambda$ state transition probability matrix, $\underline{B} = [b_{s,j}]$ is a $S_\lambda \times 2$ output distribution probability matrix, and $\underline{\Pi} = [\pi_s]$ is a $S_\lambda \times 1$ initial state probability vector. By changing these parameters of the HMM, different degrees of correlation can be obtained.

As shown in [7], this model can be reduced to an equivalent HMM that directly outputs the pair (X_k, Y_k) without any reference to E_k. Its Trellis diagram has S_λ states and $4S_\lambda$ branches arising from each state, one for each possible output (X_k, Y_k) combination. The associated branch transition probabilities are easily derived from the set of parameters λ_{HMM} of the original HMM and the marginal probability $P(x_k)$, yielding

$$T_k^{MS}(S_{k-1}^{MS} = s', S_k^{MS} = s, X_k = q, Y_k = v) \triangleq \begin{cases} a_{s',s}\, b_{s',0}\, 0.5 \text{ if } q = v, \\ a_{s',s}\, b_{s',1}\, 0.5 \text{ if } q \neq v, \end{cases} \quad (1)$$

where again $s, s' \in \{1, \ldots, S_\lambda\}$ and $q, v \in \{0, 1\}$. The label MS for the branch functions $T_k^{MS}(\cdot)$ and the state variables S_k^{MS} stands for *Multiterminal Source.*

2.2 Joint Source-Channel MAP Receiver Based on SISO Decoders

The tasks of source (compression) and channel coding of $\{X_k\}_{k=1}^K$ are jointly performed by means of a turbo code. The corresponding turbo decoder must be modified to take into account the correlated data $\{Y_k\}_{k=1}^K$ available at the receiver in order to decrease the required transmit energy. Such a joint decoder will estimate the original source sequence $\{X_k\}_{k=1}^K$ as $\{\widehat{X}_k\}_{k=1}^K$ under the MAP (Maximum a Posteriori) rule

$$\widehat{x}_k = \arg\max_{x_k \in \{0,1\}} P\left(x_k | \{\tilde{x}_k, \tilde{p}_k^1, \tilde{p}_k^2, y_k\}_{k=1}^K\right), \quad (2)$$

where $P(\cdot|\cdot)$ denotes conditional probability, and $k = 1, \ldots, K$. This can be achieved by applying the Sum-Product Algorithm (SPA) over the factor graph that jointly describes the two constituent recursive convolutional encoders and the statistical structure of the two correlated sources [11]. This factor graph is plotted in Fig. 2. Observe that such a graph is composed of 3 subgraphs (SISO decoders) corresponding to both convolutional decoders ($D1$ and $D2$) and the multiterminal source (MS), all of them describing Trellis diagrams with different parameters. Notice that variable nodes X_k and Y_k are related via T_k^{MS}, where $T_k^{MS}(\cdot)$ is given in expression (1). Also note the local functions $I_{Y_k}(y_k)$, which are indicator functions taking value 1 when $Y_k = y_k$, i.e. when the variable Y_k equals the known value y_k.

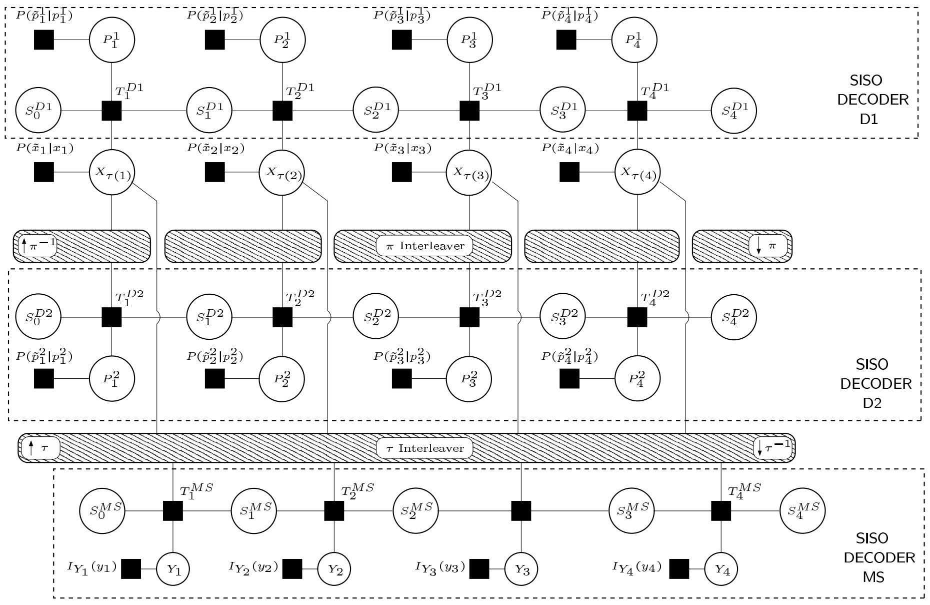

Fig. 2. Factor graph of the turbo joint source-channel decoder proposed in [7]

The SPA, when applied to cycle-free graphs, comprises a finite number of steps and provides the exact values for the computed marginalization [11]. However, the factor graph shown in Fig. 2 has cycles, and hence the algorithm has no natural termination. To overcome this issue, the SPA is separately applied to each subgraph — which, in turn, reduces to the Forward Backward (FBA) or BCJR algorithm — in the order $MS \rightarrow D1 \rightarrow D2$.

In practice, the forward and backward recursions are computed in the logarithmic domain to reduce complexity (Log-MAP and Max-Log-MAP). Let us be more concise and propose the notation:

– $L_{in}^{e,i}(x_k)$: extrinsic LLR (log-likelihood ratio) coming from Di ($i \in \{1,2\}$).
– $L_{in}^{s}(x_k)$: extrinsic LLR coming from SISO decoder MS.
– $L_{out}^{e}(x_k)$: *a posteriori* LLR generated at a certain SISO decoder.
– $t_p(s', s, x_k, y_k) \triangleq log\left(T_k^{MS}(s', s, x_k, y_k)\right)$: Logarithmic version of $T_k^{MS}(\cdot)$.

With the above notation, the Max-Log-MAP FBA recursions for SISO decoder $D1$ result in (similar expressions can be obtained for $D2$):

$$\overline{\gamma}_k(s', s) = \frac{1}{2}x_k\left(L_{in}^{e,2}(x_k) + L_{in}^{s}(x_k) + L_c\tilde{x}_k\right) + \frac{1}{2}L_c\tilde{p}_k^1 p_k^1, \tag{3}$$

$$\overline{\alpha}_k(s) = \underset{s' \in \{1,\dots,S^c\}}{max^*}\{\overline{\alpha}_{k-1}(s') + \overline{\gamma}_{k-1}(s', s)\}, \tag{4}$$

$$\overline{\beta}_{k-1}(s') = \underset{s \in \{1,\dots,S^c\}}{max^*}\{\overline{\beta}_k(s) + \overline{\gamma}_{k-1}(s', s)\}, \tag{5}$$

$$L^e_{out}(x_k) = max^*{}_{(s',s)\in S^+}\left\{\overline{\alpha}_k(s') + \frac{1}{2}L_c\tilde{p}^1_k p^1_k + \overline{\beta}_{k+1}(s)\right\}$$
$$- \max^*_{(s',s)\in S^-}\left\{\overline{\alpha}_k(s') + \frac{1}{2}L_c\tilde{p}^1_k p^1_k + \overline{\beta}_{k+1}(s)\right\}, \qquad (6)$$

where $L_c \triangleq 2\sqrt{E_c}/\sigma^2$, $s', s \in \{1, \ldots, S^c\}$, and S^+ and S^- are the set of Trellis branches (s', s) with output $x_k = 1$ and $x_k = 0$, respectively. For the the SISO decoder MS, the recursions are:

$$\overline{\gamma}_k(s', s) = \frac{1}{2}x_k\left(L^{e,1}_{in}(x_k) + L^{e,2}_{in}(x_k) + L_c\tilde{x}_k\right) + t_p(s', s, x_k, y_k) \qquad (7)$$

$$\overline{\alpha}_k(s) = \max^*_{s'\in\{1,\ldots,S_\lambda\}}\{\overline{\alpha}_{k-1}(s') + \overline{\gamma}_{k-1}(s', s)\} \qquad (8)$$

$$\overline{\beta}_{k-1}(s') = \max^*_{s\in\{1,\ldots,S_\lambda\}}\{\overline{\beta}_k(s) + \overline{\gamma}_{k-1}(s', s)\} \qquad (9)$$

$$L^e_{out}(x_k) = \max^*_{(s',s)\in S^+}\left\{\overline{\alpha}_k(s') + t_p(s', s, x_k, y_k) + \overline{\beta}_{k+1}(s)\right\}$$
$$- \max^*_{(s',s)\in S^-}\left\{\overline{\alpha}_k(s') + t_p(s', s, x_k, y_k) + \overline{\beta}_{k+1}(s)\right\}. \qquad (10)$$

Finally, the overall LLR for the source symbols X_k $(k = 1, \ldots, K)$ will be given by the sum of all LLR values arriving at variable node X_k, i.e. $LLR(x_k) = L^e_{out}(x_k) + L^{e,1}_{in}(x_k) + L^{e,2}_{in}(x_k) + L_c\tilde{x}_k$, over which a *hard decision* is carried out to provide the estimate $\widehat{X}_k$.

3 Hardware Architecture

The process of iterative joint source-channel turbo decoding described before has been implemented by means of a clustered VLIW ASIP. This kind of processor provides the best trade-off between area, performance and power in most signal processing applications.

The whole processor is meticulously optimized for the implementation of the convolutional SISO decoders *D1* and *D2* as well as for the implementation of SISO decoder *MS*. In a general perspective, the architecture comprises a master controller, a datapath and memory elements that store separately the data and the program.

The master controller consists of the instruction memory, the instruction decoder and the address generation unit. Following the approach exposed in [1], our controller is microprogrammed to direct each stage of the pipeline. This provides it with a higher flexibility and optimizes the resources utilization ratio, although complicates the labor of the programmer who is in charge of maintaining the pipeline [12].

All the computations of the system are performed in the Datapath of the processor, therefore the main design effort was carried out on it. This module is responsible for the frequency and latency of the whole decoder. Table 1 shows the computational needs of the datapath. The number of inputs, outputs and necessary hardware resources — adders, multipliers and Add Compare Select (ACS)

Table 1. SISO decoders computational needs

Operation	Inputs	Outputs	Add.	Mult.	ACS
SISOs D1 D2 *Gamma*	5	3	3	1	0
SISOs D1 D2 *Alpha Beta*	10	8	0	0	16
SISOs D1 D2 L^e_{out}	17	1	8	0	32
SISO MS *Gamma*	7	9	10	1	0
SISO MS *Alpha Beta*	10	2	0	0	8
SISO MS L^e_{out}-*LLR*	13	2	9	0	16

structures — are displayed. The Datapath is composed of two FUs (Functional Units), one for the *gamma* operations, which will be referred to as *gamma* FU, and another one for the *alpha, beta, LLR* and L^e_{out} operations which will be referred to as *ABLE* FU.

3.1 *ABLE* FU

The *ABLE* FU must be able to compute six operations that have different amounts of inputs — up to 17 — and outputs and different hardware resources needs. Based on [1] and in order to reduce the connectivity necessities, the number of registers inside the FU and improve the throughput, the serialization of the fetching and asserting of data was considered in the design.

Departing from equations 7 to 10 and 11 to 14 and Table 1 we came up with the design that is depicted in Fig. 3. The figure includes ACS (Add-Compare-Select) structures that implement the Jacobian algorithm approximation in the maximization, i.e. the *max** operator; there are also CSA (Compare Select Accumulate) structures that are used to compute many comparisons throughout several

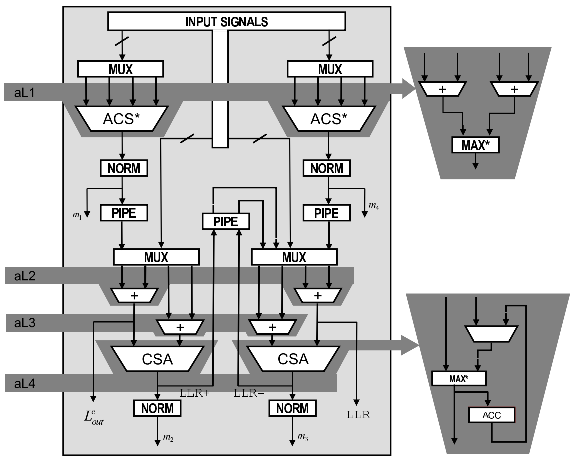

Fig. 3. *ABLE* Unit Implementation

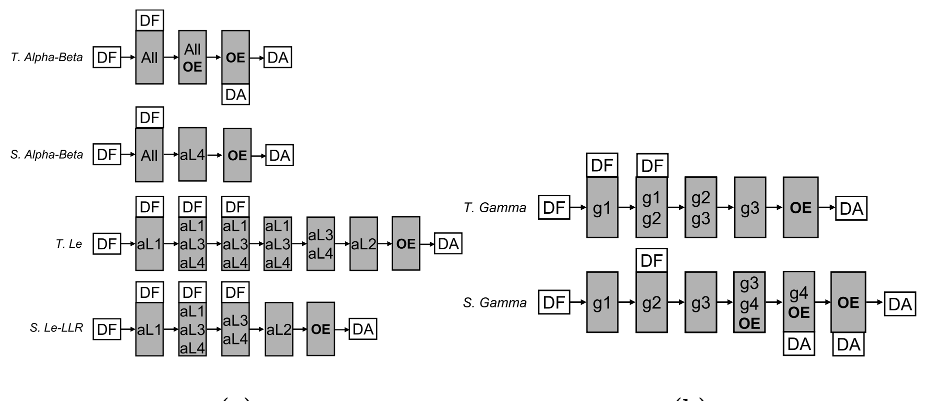

Fig. 4. (a) Pipelined execution of the *ABLE* Unit Operations.(b) Pipelined execution of the *Gamma* Unit Operations.

cycles. Pipeline registers (PIPE) divide the system into two halves comprising add-*max*∗-norm structures which yields a design that is both well-balanced and independent of the max^* module implementation. The figure is segmented into four hardware blocks, aL1 to aL4, that will serve to explain each of the pipeline stages of each operation.

The execution of each operation of the *ABLE* Unit is described in Fig. 4.a. All the operations start with a data fetch (DF) and end with a data assert (DA), in between they use different hardware blocks and also fetch and assert data. Note that, since the outputs of the system are registered, these are always asserted one cycle later than produced, requiring an output enable (OE) signal.

As far as the operations of SISO decoders *D1* and *D2* are concerned, in the case of the *alpha-beta* computations, the unit in the first cycle half the *state metrics* and one *gamma* are fetched and in the second cycle the rest of the data is input, moreover in the second cycle the first set of four *state metrics* is produced and the remaining four *state metrics* are computed in a third cycle, all this entails a throughput of two cycles/symbol. Concerning the L^e_{out} operation, at each of the first four cycles the *state metrics* of a butterfly, which share the same *gamma*, are fetched and subsequently processed, a throughput of four cycles/symbol is achieved.

Regarding the operations of SISO decoder *MS*, the *alpha-beta* computations process all the branches arising from one *state metric* — i.e. *alpha* or *beta* — each cycle so that the number of inputs is reduced. Therefore in the first cycle one *state metric* is fetched along with the corresponding four *gammas*, and in the second cycle the other *state metric* and the rest of the *gammas* are fetched. In the case of the L^e_{out}-*LLR* operations, the procedure is similar to that of the other decoders, but taking in consideration the new trellis section constraints. These two previous operations achieve a throughput of two cycles/symbol.

Table 2. Latencies and throughputs of the datapath operations

Operation	Latency	Throughput
SISOs D1 D2 *Alpha Beta*	3	2
SISOs D1 D2 *LLR* L^e_{out}	7	4
SISO MS *Alpha Beta*	3	2
SISO MS *LLR* L^e_{out}	5	2
SISOs D1 D2 *Gamma*	5	2
SISO MS *Gamma*	6	2

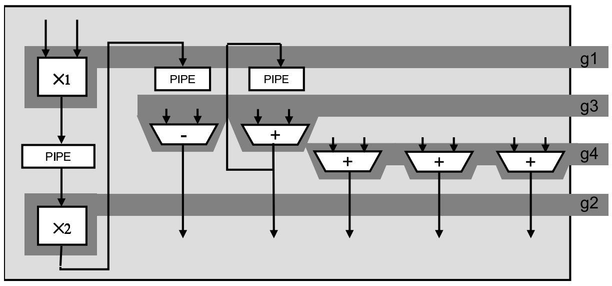

Fig. 5. *Gamma* Unit Implementation

Finally, Table 2 summarizes the latencies and throughputs of all the operations in the *ABLE* FU. Since these computations take their inputs from the *gamma* FU, the *gamma* computations of all the SISO decoders will have to achieve a throughput of two cycles/symbol.

3.2 *Gamma* FU

The *gamma* FU must be able to compute the *gamma* operations of the log-MAP algorithm and the source decoding. At first glance the only remarkable difference between equations 7 and 11 are the terms $\frac{1}{2}L_c\tilde{p}^k_k p^i_k$ — which can take two values, depending on p^i_k — and $t_p(u_k, z_k, s', s)$ — which can take eight different values that are prestored in a look-up table. Taking this into account, along with the figures in Table 1 and the throughput constraint of two cycles/symbol, we devised the structure displayed in Fig. 5. Here, the symbols $X1$ and $X2$ represent the first and second stages of a pipelined multiplier, respectively. Again, the unit is divided into four hardware blocks to ease the comprehension of the operations execution, shown in Fig. 4.b.

The *gamma* operation for SISOs *D1 D2* has to yield three outputs — two *gammas* and $\frac{1}{2}L_c y^p_k x^p_k$ — and achieves this with a latency of five cycles and throughput of two cycles continuously using every block except for “g4” that is left unused. In contrast the *gamma* operation in the source decoding yields nine outputs — eight *gammas* along with $L^{e,1}_{in} + L^{e,2}_{in} + L_c\tilde{x}_k$ — allowing a throughput of two cycles/symbol and a latency of six cycles, in this case all the blocks are used continuously except for “g1” that is used once every two cycles. Table 2 summarizes all these figures.

4 Synthesis Results

Considering that each symbol has to undergo one *alpha*, one *beta* and one L_{out}^{e} computation at each of the three decoders, and having in mind the figures from Table 2 the following results are yielded. The Datapath achieves a throughput of 8 cycles/symbol when implementing decoders *D1* and *D2* and a throughput of 6 cycles/symbol when decoder *SC* is implemented, thus a whole iteration through the three decoders entails a throughput of 22 cycles/symbol.

The design was prototyped in a Xilinx VirtexII 4000 FPGA. The whole system takes up 728 slices which is a very reduced area and makes the system attractive for embedded applications. The ABLE FU takes up 353 slices, almost half of the whole system, whereas the gamma FU takes up 42 slices, a relatively small portion. The maximum frequency of the system was fixed at 73.6 MHz which entails a data throughput of 3.3 MSym/sec for each complete iteration.

5 Concluding Remarks

Exploiting the correlation among different sources in a sensor network has drawn the attention of the research community due to its promising reduction of the transmitted energy. In this work, we have presented a dual-clustered VLIW architecture that supposes the first implementation of a turbo joint source-channel decoder for spatially correlated multiterminal sources with memory. The most critical aspects of the system design have been thoroughly described, especially the complex datapath that embodies two pipelined multioperand functional units. The approach combines the benefits of programmable solutions along with the characteristics of very dedicated designs, resulting highly appropriate for an embedded solution.

References

1. Ituero, P., Lopez-Vallejo, M.: New Schemes in Clustered VLIW Processors Applied to Turbo Decoding. In: 17th IEEE International Conference on Application-Specific Systems, Architecture Processors. (2006)
2. Aaron, A., Girod, B.: Compression with Side Information using Turbo Codes. In: Proceedings of the IEEE Data Compression Conference. (2002) 252–261
3. Liveris, A.D., Xiong, Z., Georghiades, C.N.: Joint Source-Channel Coding of Binary Sources with Side Information at the Decoder using IRA Codes. In: Proceedings of the IEEE International Workshop on Multimedia Signal. (2002) 440–442
4. Garcia-Frias, J.: Joint Source-Channel Decoding of Correlated Sources over Noisy Channels. In: Proceedings of the IEEE Data Compression Conference. (2001) 283–292

5. Zhong, W., Lou, H., Garcia-Frias, J.: LDGM Codes for Joint Source-Channel Coding of Correlated Sources. In: Proceedings of the IEEE International Conference on Image Processing. Volume 1. (2003) 593–596
6. Del Ser, J., Munoz, A., Crespo, P.M.: Joint source-channel decoding of correlated sources over ISI channels. In: Proceedings of the IEEE Vehicular Technology Conference. Volume 1. (2005) 625–629
7. Del Ser, J., Crespo, P.M., Galdos, O.: Asymmetric Joint Source-Channel Coding for Correlated Sources with Blind HMM Estimation at the Receiver. Eurasip Journal on Wireless Communications and Networking, Special Issue on Wireless Sensor Networks **4** (2005) 483–492
8. Berrou, C., Glavieux, A., Thitimajshima, P.: Near Shannon Limit Error-Correcting Coding and Decoding: Turbo Codes. In: Proceedings of the IEEE International Conference on Communications (ICC). (1993) 1064–1070
9. Tian, T., García-Frías, J., Zhong, W.: Compression of Correlated Sources Using LDPC Codes. In: Proceedings of the IEEE Data Compression Conference. (2003) 450
10. Garcia-Frias, J., Zhong, W.: LDPC Codes for Compression of Multi-Terminal Sources with Hidden Markov Correlation. IEEE Communication Letters **7** (2003) 115–117
11. Kschischang, F.R., Frey, B.J., Loeliger, H.A.: Factor Graphs and the Sum-Product Algorithm. IEEE Transactions on Information Theory **47** (2001) 498–519
12. Jacome, M.F., de Veciana, G.: Design challenges for new application-specific processors. Design&Test of computers **17** (2000) 40–50