Chapter 4
Parallel Architectures for Turbo Product Codes Decoding

Camille Leroux, Christophe Jego, and Patrick Adde

4.1 Introduction

High throughput telecommunication systems such as long-haul optical transmission or passive optical networks require powerful error correcting codes in order to increase their optical budget. In such speed-constrained applications, the classical (255,239) Reed–Solomon code is gradually being replaced by more powerful forward error correction (FEC) schemes. In [1], turbo product codes (TPC) [2] are seen as the third generation FEC for optical transmission systems. TPC tend to be good candidates for emerging optical systems. The inherent parallel structure of the product code matrix confers to TPC a good ability for parallel decoding. Nevertheless, enhancing parallelism rate rapidly induces the use of a prohibitive amount of memory. Many architectural solutions were proposed to efficiently exploit parallelism in TPC decoding. Moreover, TPC decoding provides several level of parallelism and it is not always clear which level is the most efficient. In this chapter, several parallelism level of TPC decoding are identified. Each parallelism level is characterized in terms of the potential hardware efficiency that it may bring to the architecture. From this design space exploration, we focus on one efficient architecture that exploits different levels of parallelism.

After a brief introduction of the TPC coding and decoding concept in Sect. 4.2, a straightforward hardware implementation of a TPC decoder is presented in Sect. 4.3 in order to highlight the inherent problem of parallelization in TPC decoding.

C. Leroux • C. Jego (✉)
IMS Laboratory, Bordeaux-INP, France
e-mail: camille.leroux@ims-bordeaux.fr; christophe.jego@ims-bordeaux.fr

P. Adde
LabSticc, TELECOM Bretagne, France
e-mail: patrick.adde@telecom-bretagne.eu

C. Chavet, P. Coussy (eds.), *Advanced Hardware Design for Error Correcting Codes*, DOI 10.1007/978-3-319-10569-7_4

Then, Sect. 4.4 defines and characterizes all the parallelism levels in TPC decoding. A review of existing architectural solutions is given before the detailed description of a TPC decoder architecture without any interleaving resource. This TPC decoder includes a fully parallel SISO decoder architecture which is also described in detail. Finally, Sect. 4.7 gives some synthesis results and demonstrates the efficiency of the proposed TPC decoder by comparison with current TPC decoders.

4.2 TPC Coding and Decoding Principles

The concept of product codes is a simple and efficient method to construct powerful codes with a large minimum Hamming distance d using cyclic linear block codes [3]. Despite the existence of several other decoding algorithms [4], the Chase–Pyndiah algorithm [2] is known to give the best trade-off between performance and decoding complexity [5]. Product codes were adopted in 2001 as an optional correcting code system for both the up link and down link of the IEEE 802.16 standard (WiMAX) [6].

4.2.1 Product Codes

Let us consider two systematic cyclic linear block codes C_1 having parameters (n_1,k_1,d_1) and C_2 having parameters (n_2,k_2,d_2) where n_i, k_i, and d_i $(i=1,2)$ stand for code length, code dimension, and minimum Hamming distance, respectively. As shown in Fig. 4.1, the product code $P=C_1\times C_2$ is obtained by (a) placing $(k_1\times k_2)$ information bits in a matrix of k_1 rows and k_2 columns, (b) coding the k_1 rows using code C_2, and (c) coding the n_2 columns using code C_1.

Considering that C_1 and C_2 are linear block codes, n_1 rows are codewords of C_2 exactly as all n_2 columns are codewords of C_1 by construction. Furthermore, the

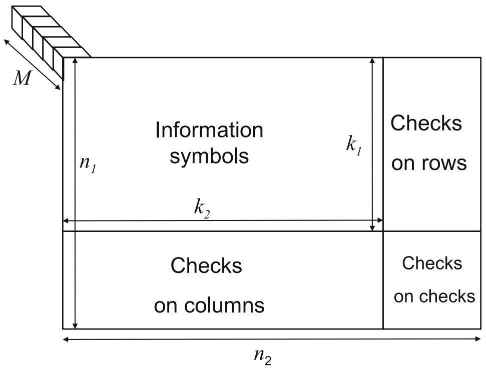


Fig. 4.1 Product code matrix structure

parameters of the resulting product code $C_P(n_p, k_p, d_p)$ are given by $n_p = n_1 \times n_2$, $k_p = k_1 \times k_2$, and $d_p = d_1 \times d_2$. The code rate R_p is given by $R_p = R_1 \times R_2$. Thus, it is possible to construct powerful product codes using two linear block codes. In the following sections, without loss of generality, we consider a squared product code, meaning that $n_1 = n_2 = n$. The most commonly used component codes are Bose Chaudhuri Hocquenghem (BCH) codes. These codes are an infinite class of linear cyclic block codes that have capabilities for multiple error detection and correction. Reed–Solomon (RS) codes can also be used as component codes. RS codes are non-binary codes in which symbols are represented on $M_{RS} = log(n+1)$ bits while $M_{BCH} = 1$. As discussed later, RS-TPC present several advantage in terms of parallelism and decoding performance [7, 8]. Without loss of generality, in the remaining of the chapter, unless specified otherwise, we assume that $M = 1$.

4.2.2 Iterative Decoding of Product Codes

Product codes usually have high dimension which precludes Maximum-Likelihood (ML) soft-decision decoding. Yet, the particular structure of this code family lends itself to an efficient iterative "turbo" decoding algorithm offering close-to-optimum performance at high enough signal-to-noise-ratios (SNRs). The Turbo-decoding of product codes consists in successively alternate decoding rows and columns using soft-input soft-output (SISO) decoders. Repeating this soft-decision decoding during several iterations enables the reduction of the bit error rate (BER). Each decoder has to compute soft information R'_{it+1} from the channel received information R and the information R'_{it} computed during the previous half-iteration. Despite the existence of several other decoding algorithms [4], the Chase–Pyndiah algorithm is known to give the best trade-off between performance and decoding complexity [5]. The Chase–Pyndiah SISO algorithm for a $t = 1$ BCH code [2, 9] is summarized below. t represents the maximum number of correctable errors for the component code.

1. Search for the L least reliable bits in the previous half-iteration output vector R'_{it} such that λ_i represents the ith minimum, $1 < i < L$.
2. Compute the syndrome $S(t_0)$ of R'_{it},
3. Compute the parity of R'_{it},
4. Generate p test patterns τ_i obtained by inverting some of the L least reliable bits ($p \leq 2^L$).
5. **For each** test pattern $(1 \leq i \leq p-1)$
   - Compute the syndrome $S(\tau_i)$,
   - Correct the potential error by inverting the bit position $S(\tau_i)$,
   - Recompute the parity considering the detection of an error and the parity of R'_{it},
   - Compute the square Euclidean distance (metric) M_i between R'_{it} and the considered test pattern τ_i.

6. Select the Decided Word (DW) among test patterns having the minimal metric (M_{DW}) and choose Cw competitors codewords $\mathbf{c}_i$ $(1 < i < Cw)$ having the second, third, ...,. ith minimum metric.
7. **For each** symbol of the DW,

   - Compute the new reliability F_{it}:

$$F_{it} = \begin{cases} \beta_{it} = (|R'_{it}| + \sum_{i=1}^{L} \lambda_i) - \min(M_i) \text{when no competitor exists} \\ \\ F_{it} = \min_2(M_i) - \min(M_i) \text{otherwise,} \end{cases}$$

   - Compute extrinsic information $W_{it} = F_{it} - R'_{it}$,
   - Add extrinsic information (multiplied by α_{it}) to the channel received word, $R'_{it+1} = R + \alpha_{it} W_{it}$.

As explained in [10], decoding parameters L, p, Cw and the number of quantization bits q of the soft information have a considerable effect on decoding performance and complexity. The α_{it} coefficient allows decoding decisions to be damped during the first iterations. β_{it} is an estimation of F_{it} when no competitor exists. As detailed in [11], it is based on the least reliable bits value.

Figure 4.2 shows the BER performance of various $t = 1$ BCH and RS codes. In general, for a fixed t value, the code rate increases with N. This explains why the BER curves are shifting to the right when N increases. However, for large codelengths, the slope is steeper.

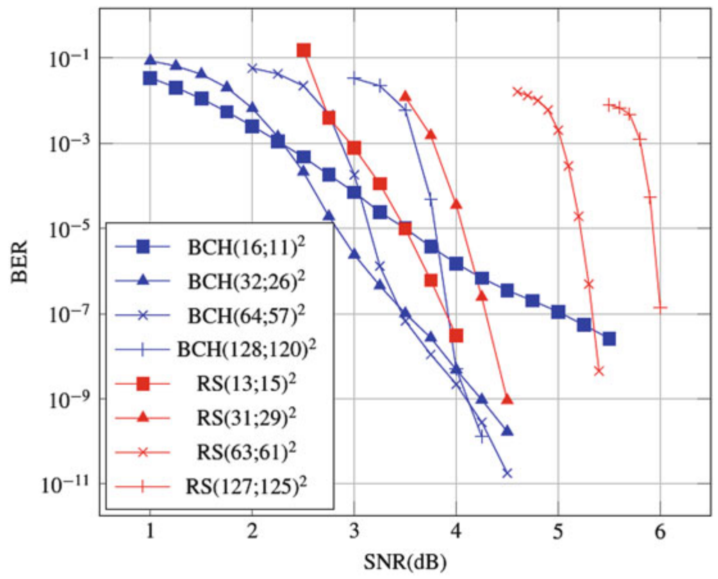


Fig. 4.2 BER performance of various BCH and RS product codes on an AWGN channel

4.3 Straightforward Hardware Implementation of a TPC Decoder

4.3.1 Global TPC Decoder Architecture

In a straightforward implementation of a TPC decoder, the channel information matrix R (consisting in n^2 q-bits LLRs) is stored in a memory. As shown in Fig. 4.3, since the SISO decoder reads R_A during the whole decoding process, this memory has to be duplicated so that the next channel information matrix R_B can be written while the decoder processes the current matrix R_A. A single sequential SISO decoder reads information from the R memory and performs the decoding process by updating the R' messages iteratively. Assuming I decoding iterations, the SISO decoder should update $2 \times I \times n^2$ LLRs.[1] In the most favorable case, let us assume that the SISO decoder is able to update one LLR per clock cycle, the resulting throughput is $T = f/(2In^2)$, where f is the clock frequency. For a $(32,26)^2$ BCH code with six decoding iterations and a clock frequency of 500 MHz, the resulting throughput is 40 Kb/s. This kind of architecture is clearly too slow for high throughput applications. In this chapter, various methods to enhance the parallelism are reviewed.

4.3.2 Sequential SISO Decoder Architecture

The TPC decoder architecture described in Fig. 4.3 includes a SISO decoder that sequentially process incoming LLRs. Figure 4.4 shows the structure of such a sequential SISO decoder. It is subdivided into four units.

The reception unit

- computes the syndromes of the incoming vectors,
- selects the p least reliable bits.

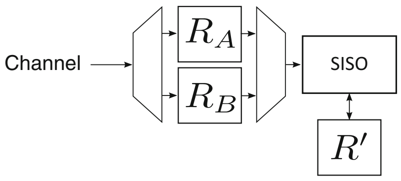


Fig. 4.3 Sequential implementation of a TPC decoder

[1]A full iteration corresponds to a row-wise decoding followed by a column-wise decoding, which explains why the R' matrix has to be updated $2I$ times.

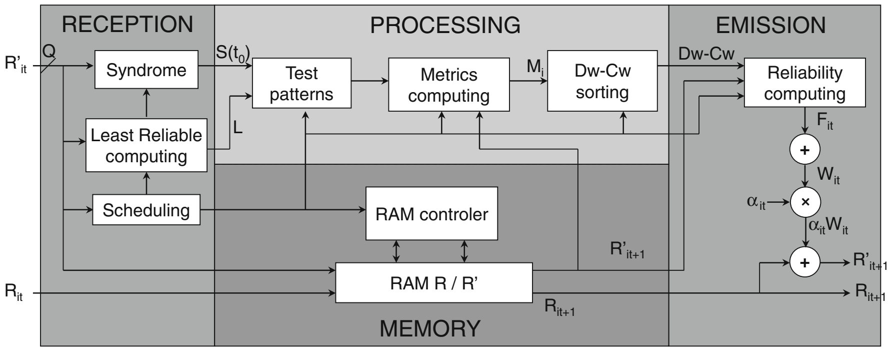


Fig. 4.4 Architecture of a SISO decoder

The processing unit

- determines the test vectors by inverting some of the least reliable bits,
- computes the metric of each test vector,
- selects the most likely test vector (the one with the minimum soft-distance)
- selects the Cw concurrent test vectors (2nd minimum, 3rd minimum, etc.).

The emission unit

- computes the new reliability F_{it} of each outcoming bit of the considered vector,
- computes and ponderates the extrinsic information $\alpha_{it} W_{it}$,
- generates the soft-output LLRs for the next half-iteration R'_{it+1}.

The memory unit stores the channel information and the soft information for the current half-iteration.

In terms of latency, the syndrome and the least reliable bits can only be computed once the whole $n\times$-LLR vector has been shifted in. Only then, the test vectors processing and the soft-output computation can be performed. This means that it takes at least n clock cycles to read R and $[R'_{it}]$, 1 clock cycle to perform the test vector computation,[2] and n clock cycles to write back the n LLRs in the R' memory. This means that n LLRs require at least $2n+1$ clock cycles to be updated which corresponds to a throughput of $T = f/(4In^2)$ (we assume that $n \gg 1$). Taking the SISO latency into account, the previously estimated 40 Kb/s TPC decoder has in fact a throughput of only 20 Kb/s.

The hardware complexity of a sequential SISO decoder is rather low, thanks to its serial-processing nature. The SISO decoder designed in [10, 12] has an equivalent complexity of a few thousands gates. The computational complexity of the SISO decoder depends on the choice of algorithmic parameters. As mentioned in Sect. 4.2.2, the Chase–Pyndiah algorithm includes parameters L, p, Cw, q which impact on both the decoding performance and the computational complexity of the

[2]This assume that one is able to design a parallel processing unit that computes and select metrics in a single clock cycle.

TPC decoder. Depending on the application one should identify a parameter set that enables sufficient decoding performance while minimizing the hardware footprint of the resulting SISO decoder. In [10,12], a case of study is detailed for a $(32,26)$ BCH code SISO decoder. Depending on the parameter set that is selected, the complexity of a SISO decoder varies by a factor 2. This shows that the algorithmic parameter set is an important factor to take into account when designing a TPC decoder.

4.4 From Parallelism Levels to Parallel Architectures

An architecture can be characterized by different metrics such as throughput, latency, hardware complexity, power consumption, routing density, etc. In this study, we aim at high speed architectures with low hardware complexities. Consequently, the performance is measured with throughput (T) while the cost function is the hardware area (A). In such a context, the efficiency of an architecture is defined as the throughput/complexity ratio : $E = T/A$. An efficient architecture would process a high data rate with a low hardware area.

The parallelism of an architecture can be defined as "the ability of the system to process several data in parallel." We formally define the parallelism P of a decoder as the number of bits that can be processed/decoded in a single clock cycle. The parallelism directly impacts the performance of an architecture. In order to quantify the benefit/disadvantage brought by the application of a parallelism P_i to an architecture, we define three metrics, the *speed gain* G_S, the *area ratio* R_C, and the *efficiency gain* G_E:

$$\begin{cases} G_S(P_i = p) = \frac{T_{P_i=p}}{T_{P_i=1}} \\ \\ R_A(P_i = p) = \frac{A_{P_i=p}}{A_{P_i=1}} \\ \\ G_E(P_i = p) = \frac{E_{P_i=p}}{E_{P_i=1}} = \frac{G_S(P_i=p)}{R_A(P_i=p)} \end{cases}$$

A parallelism level P_i is considered to be *effective* if $G_S(P_i) > 1$, while it is *efficient* when $G_E(P_i) > 1 \iff G_S(P_i) > R_A(P_i)$. One should notice that several parallelism levels may be combined but it may also be impossible to associate them. The exploitations of several parallelism levels at the same time depend on the architecture that implements these levels. In the remaining of this section, all parallelism levels in TPC decoding are detailed and characterized from the highest level (frame parallelism) down to the lowest level (intra-symbol parallelism). For each level, we provide a condition that makes the use of the considered level efficient. We also provide some reference of existing TPC decoders that use these parallelism levels.

4.4.1 Frame Parallelism

The highest level of parallelism can be observed at the frame level and this is known as *frame parallelism*. It is a form of spatial parallelism and is suitable to any decoding scheme. In TPC decoding, a frame is defined as a product code matrix. The frame parallelism consists in duplicating the processing resources, e.g., the turbo-decoder. By using this parallelism level in TPC decoding, P_{frame} matrices can be decoded at the same time. Considering P_{frame} turbo-decoders that have the same throughput T_0, the *speed gain* and *area ratio* are equivalent: $G_S = R_A = P_{frame}$. Consequently the efficiency does not increase with P_{frame}: $G_E = 1$. Actually, this level of parallelism is only limited by the affordable silicon area. Although frame parallelism makes TPC decoder architecture more effective, it does not improve its efficiency. Moreover some buffering/multiplexing resources are needed to broadcast incoming LLRs to the different decoders. The only advantage of frame level parallelism is the design time since it can be implemented by a straightforward duplication of resources on the silicon.

4.4.2 Iteration Parallelism

In a sequential TPC decoder implementation, each iteration is performed by the same SISO decoder that reads and writes data in the Interleaving Memories (IM). It is however possible to exploit the *iteration parallelism* by duplicating the elementary decoder and the associated memories in a pipelined structure. The memories have to be duplicated so that all SISO decoders can work in parallel. The maximum depth of such a structure equals to the maximum number of iteration it_{max}. Iteration parallelism is a type of temporal parallelism. Here again, the throughput benefit equals to the complexity ratio : $G_S = R_A = P_{it}$. It means that the iteration parallelism does not improve efficiency. Figure 4.5 shows a pipelined TPC decoder. It includes I stages; each of which processing one frame. This explains why the channel memory R has to be duplicated. It is also possible to implement less than

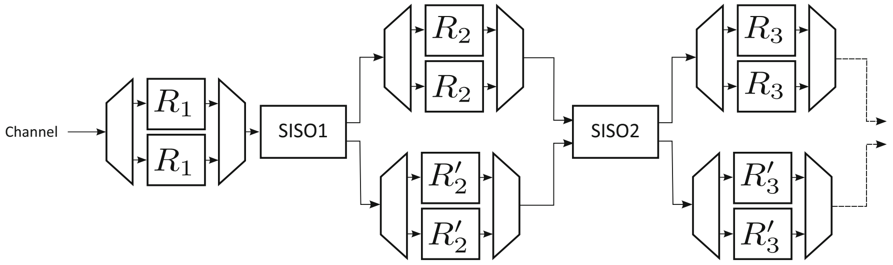


Fig. 4.5 Pipelined TPC architecture

I stages and to loop back on the hardware resources. The iteration parallelism was applied in [7] where five iterations are duplicated over five different FPGA devices. It enables to reach a throughput of 5 Gb/s.

4.4.3 Sub-block Parallelism

In a product code matrix, each row (column) is encoded independently from the others (See Sect. 4.2.1). This interesting property may also be used during the decoding process, where each row (column) is decoded independently. In an implementation prospective, it means that more than one decoder can be assigned to row (column) decoding. Considering a product code matrix of size n^2, a maximum number of n decoders can be duplicated for row (column) decoding. We designate this parallelism level as *sub-block parallelism* P_{sb}. Assuming that the duplication of SISO decoders does not induce interleaving resources duplication, G_E can be expressed as:

$$G_E = \frac{P_{sb}(A_{SISO} + A_{\pi})}{P_{sb}A_{SISO} + A_{\pi}}$$

$$G_E > 1 \Longleftrightarrow P_{sb} > 1$$

A_{SISO} and A_{π} are the areas of the SISO decoder and the interleaving resource, respectively.

In [10,13–15] solutions based on Barrel-shifter and Omega network are proposed to avoid data access conflicts when $P_{sb} = n$. This makes the *complexity ratio* lower than the *speed gain*, which means that the *efficiency gain* of the architecture increases.

4.4.3.1 Barrel Shifter

In a straightforward application of sub-block parallelism, one simply duplicates the SISO decoders. The decoder is then composed of P_{sb} SISO decoders, a memory storing n^2 q-bits LLRs from the channel R and one memory storing n^2 q-bits LLRs for the matrix $[R'_{it}]$. However, this architecture is limited by memory access conflicts. Depending on the considered iteration, the P_{sb} SISO decoders need to access a total of P_{sb} data either row-wise or column-wise. In [13], this problem is overcome for $P_{sb} = n$: a barrel shifter is introduced between SISO decoders and the interleaving register file in order to allow row/column-wise data accesses of P_{sb} data in parallel as shown in Fig. 4.6. This comes at the extra cost of a barrel-shifter with area of $O(n \log n)$. This solution enables to use the sub-block parallelism at its highest rate only: $P_{sb} = n$. The extra-complexity consists in a simple barrel shifter with a

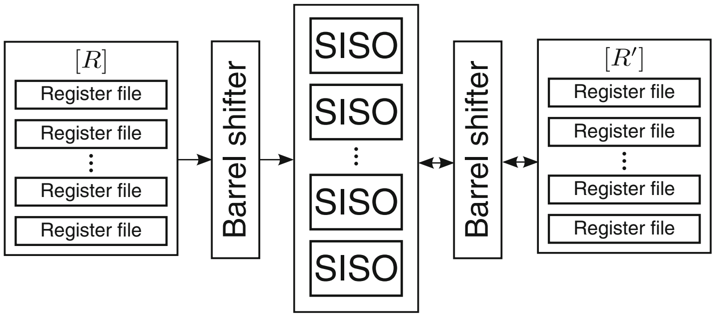


Fig. 4.6 Barrel-shifter-based parallel TPC decoder

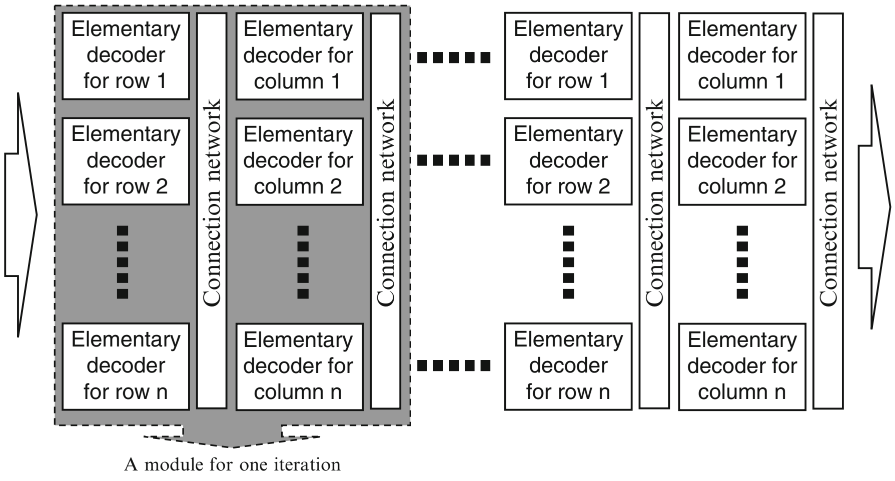

Elementary decoder for row 1
Elementary decoder for row 2
Elementary decoder for row n
Connection network
Elementary decoder for column 1
Elementary decoder for column 2
Elementary decoder for column n
Connection network
Elementary decoder for row 1
Elementary decoder for row 2
Elementary decoder for row n
Connection network
Elementary decoder for column 1
Elementary decoder for column 2
Elementary decoder for column n
Connection network
A module for one iteration

Fig. 4.7 Omega network-based parallel TPC decoder without interleaving memory

complexity of $O(n\log(n))$. However, it still includes a large amount of interleaving memory for storing R'. This is especially problematic if one wants to use iteration parallelism where the interleaving resources have to be duplicated.

4.4.3.2 Omega Network

In [10, 14, 15], it is suggested to replace the interleaving memory by a simple interconnection network (Omega network). This is made possible by the cyclic nature of the component codes (BCH or RS codes): applying a circular shift on a codeword ends up in another codeword. In terms of decoding, this means that the decoding process can start with any bits in the codeword. The decoding process is then applied on a shifting diagonal. This avoids data access conflicts as long as data are correctly routed from one iteration to another as shown in Fig. 4.7.

The interleaving-memory-less architecture was prototyped on an FPGA device [10]. This TPC decoder also has a maximal sub-block parallelism ($P_{sb} = n$), while the hardware complexity of the interleaving resources is drastically reduced since interleaving memory is no more needed.

4.4.4 Symbol Parallelism

A finer-grained parallelism is the *symbol parallelism*. It can be defined as the ability of a SISO decoder, to process P_{sym} symbols of the same sub-block (row or column) in parallel. In a sequential SISO decoder, input data are shifted in a serial manner. Every incoming symbol implies some internal metrics to be updated (syndrome, least reliable bits, ...). By increasing P_{sym}, some parts of the decoder datapath have to be duplicated, (e.g., the reliability computation stage). However, the other blocks, such as the test pattern metric computation, or the competitor vector determination block, remain identical when P_{sym} increases. Consequently, the *area ratio* is lower than the *speed gain* : $G_E > 1$. For an architecture that avoid interleaving resource duplication, the following inequality is verified:

$$G_E > 1 \Longleftrightarrow A_{DEC}(P_{sym} = p) < p \times A_{DEC}(P_{sym} = 1)$$

$A_{DEC}(P_{sym} = p)$ is the hardware complexity of a SISO decoder with a symbol parallelism equal to p. Increasing P_{sym} also means that the interleaving memory should be able to read/write more than one data during the same clock cycle. In [12, 16] solutions were provided in order to exploit this parallelism while avoiding interleaving memory duplication. Logic synthesis results confirm that the efficiency increases with P_{sym}.

4.4.4.1 Memory Merging

In [16] an architecture that uses symbol parallelism in conjunction with sub-block parallelism is proposed. The idea is to store several LLRs at the same address and to design elementary decoders able to process $P_{sym} = m$ symbols during the same clock period (denoted as m-decoders). A half-iteration structure includes n/m decoders each decoding m symbols in one clock period and an interleaving memory of size $4 \times q \times M \times n^2$. This scheme actually exploits symbol parallelism on one dimension of the matrix and sub-block parallelism on the other dimension in such a way that $P_{sb} = P_{sym} = m$. The resulting throughput is $O(m^2)$ while the overhead factor of the decoder complexity is $\sim \frac{m^2}{2}$. In this work, the maximum reached parallelism rate is $m^2 = 64$, with $m = 8$ SISO decoders.

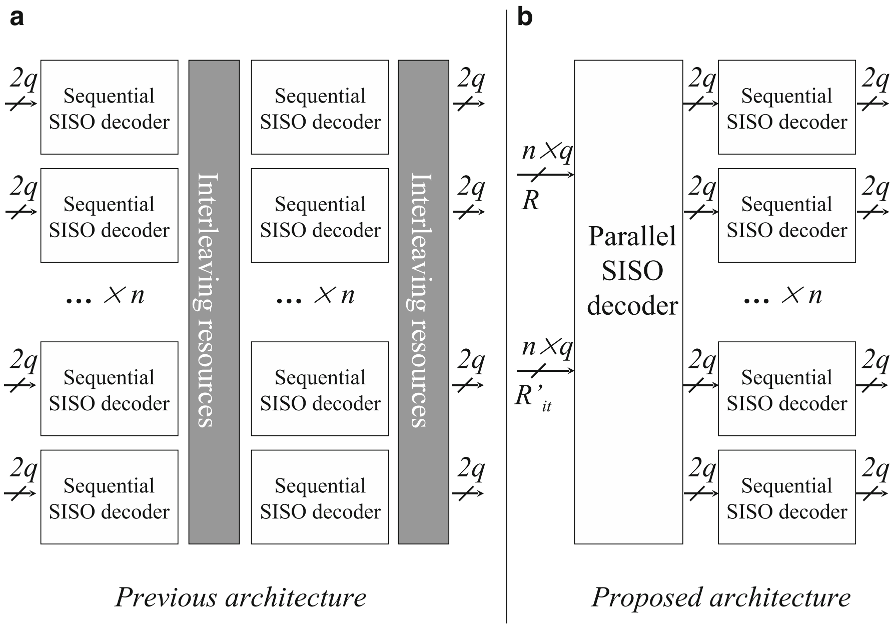


Fig. 4.8 Omega network-based parallel TPC decoder (**a**) and fully parallel SISO-based TPC decoder architecture (**b**)

4.4.4.2 Fully Parallel SISO Decoder

In [12], an architecture with $P_{sym} = n$ is proposed. A fully parallel SISO decoder enables to decode a whole column in a single clock cycle after a few cycles of latency. The n generated LLRs are then directly fed into n sequential SISO decoders which perform row decoding. In such an architecture the interleaving resources are simply removed since generated data are immediately consumed. Logic synthesis results show the higher efficiency of this architectural solution in comparison with the previously described ones. This can be easily explained by the fact that the complexity of one fully parallel SISO decoder is lower than n SISO decoders. This TPC decoder architecture will be described in detail in Sect. 4.5 (Fig. 4.8).

4.4.5 Intra-symbol Parallelism

In TPC decoding, BCH codes are often used for their good decoding performance/-complexity trade-off. In [7, 17], it was shown that using RS codes as component codes can provide similar decoding performance with a reasonable computational complexity overhead.

From an architectural point of view, the non-binary structure of RS codes enables to exploit an extra parallelism level, the *intra-symbol parallelism* P_{is}. In an RS code of size n, a symbol consists in $M = \log(n+1)$ bits (see Fig. 4.1). An RS-SISO decoder can either shift-in symbols bit by bit or symbol by symbol. It provides a maximal parallelism rate of $\max(P_{is}) = \log(n+1)$.

Similarly to the symbol parallelism, the resource sharing within the RS-SISO decoder increases the efficiency. However the *efficiency gain* provided by P_{is} is hard to estimate because it is highly related to the internal architecture of the SISO decoder. Nevertheless, it is possible to give a condition that guarantees $G_E(P_{is}) > 1$:

$$A(P_{is} > 1) < P_{is} \times A(P_{is} = 1)$$

In [7], a $(31,29)^2$ RS turbo product code decoder was designed and prototyped. It has an architecture similar to [10] but it includes RS SISO decoders that process one RS symbol per clock cycle. Moreover, the iteration parallelism is used in such a way that the decoding iterations are duplicated on the 5 FPGA devices. The resulting TPC decoder reaches 5 Gb/s.

4.4.6 Comparison of Parallelism Levels

Table 4.1 summarizes benefits of parallelism levels in TPC decoding. For each parallelism P_i, the maximum *speed gain*, the *efficiency gain*, and the P_i value that maximizes the efficiency are given. *Frame parallelism* is only limited by technological issues (e.g., silicon area). This parallelism improves the effectiveness of the architecture; it is straightforward to implement but it does not improve efficiency. *Iteration parallelism* has the same impact but is upper bounded by the maximum number of iteration required by the decoding process.

Application of lower levels of parallelism (P_{sb}, P_{sym}, and P_{is}) improves the architecture efficiency. It is even maximized for highest parallelism value. However, the use of these parallelism levels is not as straightforward as P_{frame} and P_{it}. It requires some specific schedulings and/or implementation strategies.

The TPC decoder architectures mentioned in this section use different levels of parallelism and end up with different hardware efficiency. Table 4.2 provides a comparison of the state-of-the-art TPC decoders in terms of parallelism levels and

Table 4.1 Comparison of parallelism levels in TPC decoding

P_i	$\max(G_S)$	G_E	$\arg(\max(E))$
P_{frame}	∞	$\simeq 1$	$[0; +\infty[$
P_{it}	IT_c	$\simeq 1$	IT_c
P_{sb}	n	≥ 1	n
P_{sym}	n	≥ 1	n
P_{is}	$\log(n+1)$	≥ 1	$\log(n+1)$

Table 4.2 Current TPC decoder architecture comparison

Architecture	P_i	$A_\pi(1/2iter)$	$A_{Dec}(1/2iter)$
[13]	$P_{sb} = n$	$O(2qn^2) + O(2n\log(n))$	$O(n)$
[10, 14]	$P_{sb} = n$	$O(n\log(n))$	$O(n)$
[16]	$P_{sb} = m; P_{sym} = m$	$O(2qn^2)$	$O(m^2/2)$
[12]	$P_{sb} = 1; P_{sym} = n$	$\emptyset$	$< O(n)$

hardware area. For each reference, we provide the exploited parallelism levels. The hardware area is given for a half-iteration for both the interleaving resource and the decoding resources. All these architectures could use the frame and iteration parallelism P_{frame} and P_{it} by duplicating resources.

The TPC decoder in [13] uses n sequential SISO decoders, two memories of size n^2 for R and R' and two barrel shifters. In such an architecture, the critical part is the memory resources that grow with n^2. The TPC decoder in [16] uses a combination of P_{sb} and P_{sym}. This is the first architecture that uses P_{sym}. However the SISO decoders were designed for a maximum parallelism of $m = 8$. Moreover this architecture uses memory resources for interleaving which dominate the resulting hardware area. In [10, 14], the sub-block parallelism is fully exploited. The memories and the barrel shifters are replaced by an omega network to route data from one iteration to the next. The hardware area is then dominated by the n duplicated SISO decoders.

In [10, 13, 14, 16], the rebuilding of the product code matrix is necessary between each half-iteration: memory blocks and/or routing networks are used between half-iterations to read and store R'_{it} and R. Actually, more than 50 % of the complexity is in the memory for IM-based architecture, while it represents less than 10 % for omega network-based structure [14, 15]. On the decoding resources side, increasing the parallelism rate by duplicating computation resources is inefficient since the reuse of available resources is not optimized. In [12], a fully parallel SISO decoder is cascaded with n sequential SISO decoders in such a way that interleaving resources are completely removed. In fact, the internal memory of SISO decoders is sufficient to store the required R and R' matrices. The fully parallel SISO decoder is less complex than n sequential SISO decoders which make this architecture even more efficient. In the next section, due to its higher efficiency, the TPC decoder architecture of [12] is described in detail.

4.5 TPC Decoder Architecture Based on Symbol Parallelism

4.5.1 Proposed IM-Free Architecture Using Fully Parallel SISO Decoder

Considering that one can design a SISO decoder with $P_{sym} = n$, a product code matrix can be decoded without any interleaving resource as shown in Fig. 4.9.

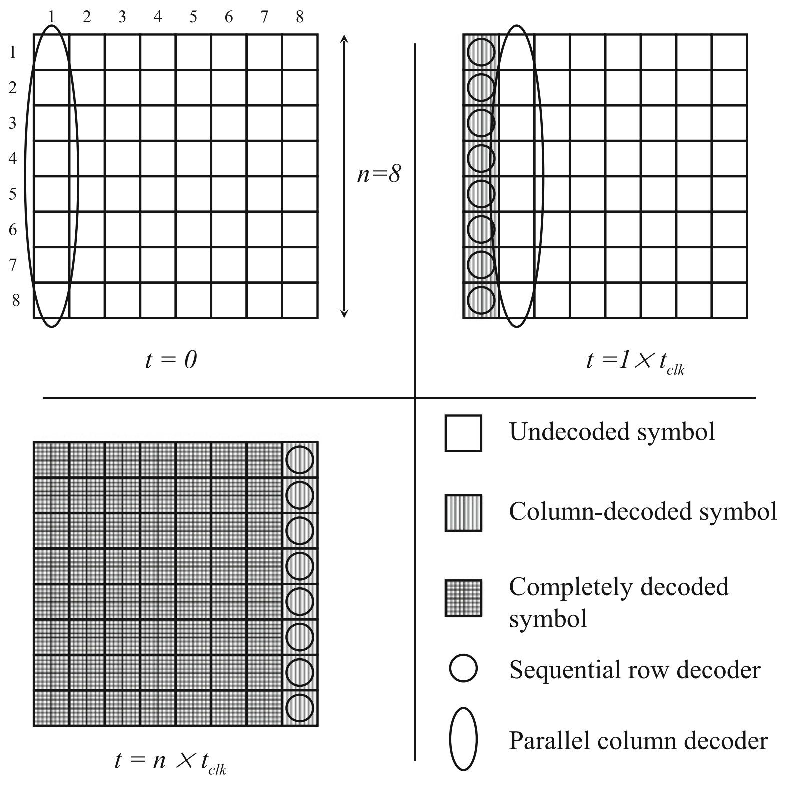


Fig. 4.9 Proposed parallel decoding scheduling of a product code matrix

At $t = 0$, the fully parallel SISO decoder processes the column 1. During the next clock period, n sequential SISO decoders ($P_{sym} = 1$) start decoding the first symbol of each row while the parallel decoder processes the column 2. During the nth clock period, sequential decoders complete matrix decoding while the parallel decoder is already decoding the next matrix. Thus, data generated by the parallel decoder is immediately consumed by the sequential decoders. Consequently, no IM or data routing resources are required between the fully parallel decoder and sequential decoders. The resulting architecture is compared to [10, 14] in Fig. 4.10 for one implemented iteration. This architecture uses row-wise P_{sb} and column-wise P_{sym}. More specifically, we have:

$$\begin{cases} P_{sym}(col) = P_{sb}(row) = n \\ P_{sb}(col) = P_{sym}(row) = 1 \end{cases}$$

One should notice that $P_{sb}(col) = P_{sym}(row)$ can be further exploited.

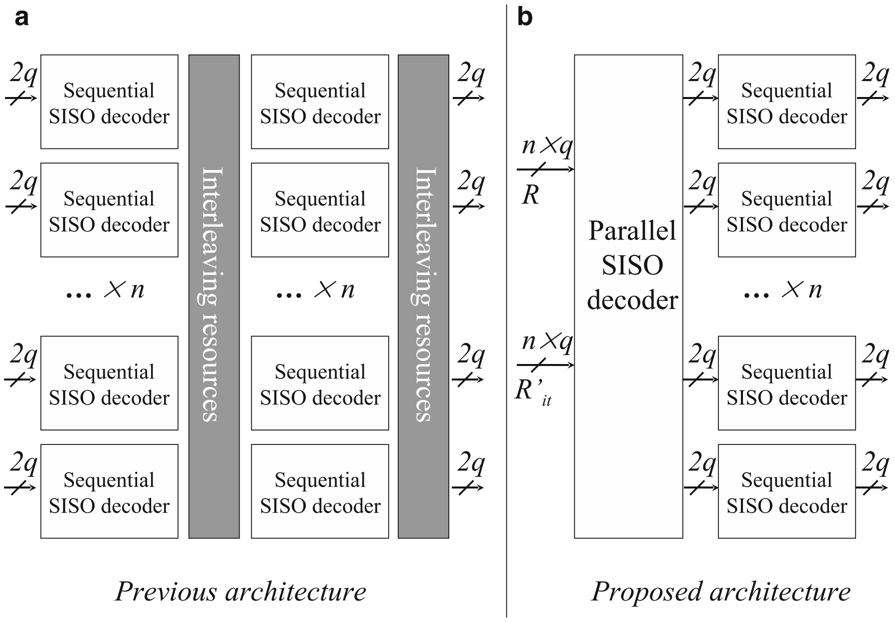


Fig. 4.10 Previous TPC decoder architecture (**a**) and proposed fully parallel SISO-based TPC decoder architecture (**b**)

4.5.2 Toward a Maximal Parallelism Rate

Starting from the IM-free architecture presented in the previous section, parallelism can be further enhanced. Figure 4.11 shows the alternate product code matrix parallel decoding scheme in which $P_{sb}(col) = P_{sym}(row) = m$ and $P_{sym}(col) = P_{sb}(row) = n$. The TPC decoder consists in $m\times$ n-decoders for column decoding and $n\times$ m-decoders for row decoding. An m-decoder can process m symbols in one clock period with $1 \leq m \leq n$. In such an architecture, the maximum reachable parallelism rate $P = n^2$ can be achieved by using n fully parallel SISO decoders for column decoding and n fully parallel SISO decoders for row decoding. Intra-symbol parallelism can also be exploited to increase the total parallelism to $P = P_{sb} \times P_{sym} \times P_{is} = n^2 \log(n)$. However, all these new architectural solutions require to design a SISO decoder able to process n symbols in one clock period.

4.6 Architecture of a Fully Parallel Combinational SISO Decoder

The proposed IM-free TPC decoder architecture requires a fully parallel combinational SISO decoder. To the best of our knowledge, only sequential SISO decoders able to process $m \leq n$ symbols in one clock period have been previously

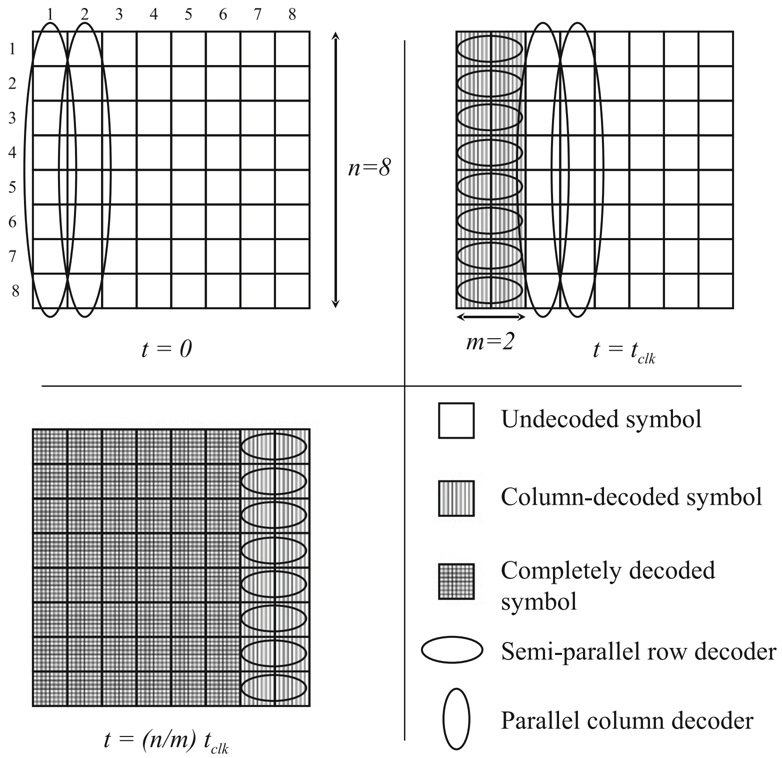


Fig. 4.11 Alternative turbo decoding scheduling for enhanced parallelism rate

designed. The design of a fully parallel combinatorial SISO decoder is a challenging issue. In the following section, such an architecture is described.

4.6.1 Algorithmic Parameter Reduction

As explained earlier in Sect. 4.2, the Chase–Pyndiah algorithm includes parameters (L, τ_p, Cw, q) which impact on both the performance and the complexity of the turbo decoding. BER simulations were performed with different parameters: $L = \{2;3;4;5\}$, $\tau_p = \{4;8;16\}$, $Cw = \{0;1;2;3\}$, $q = \{3;4;5\}$. Performing eight iterations, the parameter set $\mathscr{P}_0 = \{L = 5, \tau_p = 16, Cw = 3, q = 5\}$ gives the best BER performance for a high complexity [5]. However, algorithmic simulations showed that the reduced parameter set $\mathscr{P}_1 = \{L = 3, \tau_p = 8, Cw = 0, q = 5\}$ only induce a performance loss of 0.25 dB at BER$= 10^{-6}$ while it becomes null below BER$= 10^{-9}$. Further reducing these parameters would induce a notable performance loss. For example by simply reducing the number of test patterns:

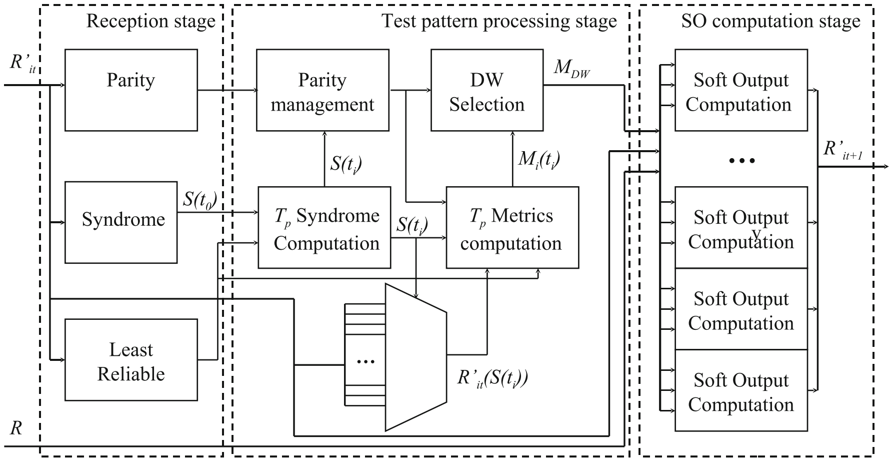


Fig. 4.12 Combinatorial version of the fully parallel SISO decoder

$\mathscr{P}_2 = \{L = 2, \tau_p = 4, Cw = 3, q = 5\}$, the performance loss reaches 0.5 dB. Consequently, using $\mathscr{P}_1$ enables the architecture to be simplified at very low performance lost below BER$=10^{-9}$.

4.6.2 *Fully Parallel SISO Decoder Architecture*

Figure 4.12 depicts the architecture of the fully parallel SISO decoder. In the first attempt a purely combinational architecture was designed. Later, a critical path study mandated the insertion of pipeline stages within the structure. The SISO decoder is split into three stages, namely the reception stage, the test pattern processing stage, and the soft-output computation stage.

4.6.2.1 Reception Stage

The reception stage corresponds to steps (1–3) of the Chase–Pyndiah algorithm detailed in Sect. 4.2. The syndrome of the incoming vector R'_{it} can be derived as $S(R'_{it}) = H \times sign(R'_{it})$ where H is the parity check matrix of the BCH code. A straightforward implementation of such a matrix multiplication is depicted on Fig. 4.13. The H matrix, the corresponding parity check equations, and the syndrome $S(t_0) = [s_2, s_1, s_0]$ implementation of a BCH(7,4) code are detailed.

It can be noticed that some parity check equations have similar terms. For instance, the term $(x_1 \oplus x_0)$ is used in both s_1 and s_2 computation. This means that a reuse of computation resources for an even more efficient implementation is possible. The parity of the incoming vector R'_{it} is computed with a similar structure

a

$$H=\begin{pmatrix} 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{pmatrix}$$

b

$$\begin{cases} s_2 = x_3 \oplus x_2 \oplus x_1 \oplus x_0 \\ s_1 = x_5 \oplus x_4 \oplus x_1 \oplus x_0 \\ s_0 = x_6 \oplus x_4 \oplus x_2 \oplus x_0 \end{cases}$$

c

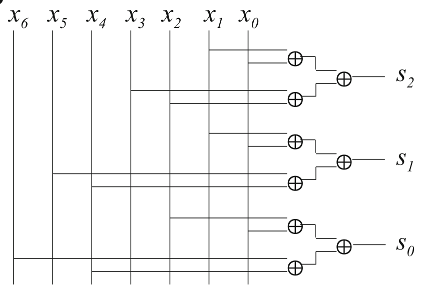
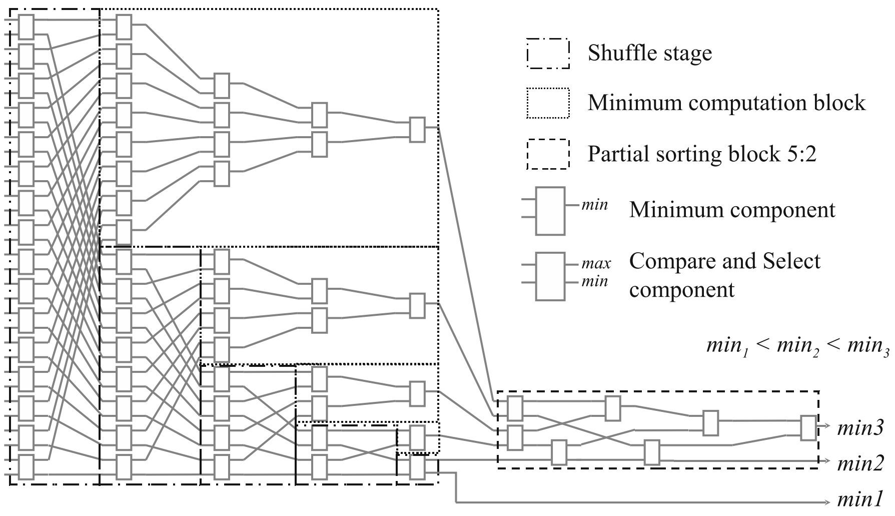


Fig. 4.13 BCH(7,4) code: (**a**) Parity check matrix. (**b**) Parity check equations. (**c**) Syndrome parallel computation implementation



Shuffle stage
Minimum computation block
Partial sorting block 5:2
min
Minimum component
max
min
Compare and Select component
$min_1 < min_2 < min_3$
min3
min2
min1

Fig. 4.14 Sorting network for least reliable bits selection

by “xoring” ($n - 1$) incoming bits. Selecting the least reliable bits among the incoming vector in parallel requires a sorting network. Such structures are composed of interconnected Compare and Select operators (CS). The interconnection scheme depends on the considered sorting algorithm. Many parallel sorting algorithm are conceivable [18]. However, most of them are optimized for a complete sorting, while the Chase–Pyndiah algorithm only requires a partial sorting (i.e., extracting L minima). Consequently we devised a network optimized, in terms of area and critical path, for the partial sorting of L=3 values among n=32, as depicted in Fig. 4.14. The structure is based on shuffle networks coupled with local minima computation blocks. After the first shuffle stage, min_1 is in the lower section while the upper section can either contain min_2 or min_3 or no minimum. The same reasoning is applied recursively. After five shuffle stages, the minimum is determined while five values can still be min_2 and min_3. A local sorting of five values enables the determination of min_2 and min_3 value. This partial sorting network requires 35 CS operators and 29 minimum elements. The critical path consists of nine comparison stages.

4.6.2.2 Test Pattern Processing Stage

The test pattern processing stage corresponds to steps (4–5) of the Chase–Pyndiah algorithm detailed in Sect. 4.2. Instead of being processed sequentially, test patterns are processed in parallel. The syndrome of each test pattern is computed by adding $S(t_0)$ with the position of the inverted reliable bits. The parity management block computes the parity of R'_{it+1} considering the parity of R'_{it} and the detection of an error which is the case when $S(t_i) \neq 0$. Metrics of each test pattern are then computed by adding the contribution of each inverted bit in the current test pattern (least reliable bits, syndrome corrected bits, and the new parity bit). The minimum metric is determined in the DW selection block. The structure is a simple minimum selection tree. The multiplexer selects $R'_{it}(S(t_i))$ in order to compute test pattern metrics.

4.6.2.3 Soft-Output Computation Stage

The last stage is a duplication of n soft-output computation blocks. As shown in Fig. 4.15, this block first computes the new reliability F_{it} of each symbol. Since no competitor word is considered, the β value is automatically assigned. The β value is based on an estimation of the competitor word metric value. It is calculated from the reliability of the corrected bit and the least reliable bits. Then, the extrinsic information is computed and damped by the coefficient α_{it} which is devised to be a power of 2 making the multiplication a simple bit shifting. Finally, the channel information is added to generate the soft output R'_{it+1}. Within this block, all computation are performed in sign and magnitude format. Other arithmetic format were explored but the chosen one requires less computation resources than others.

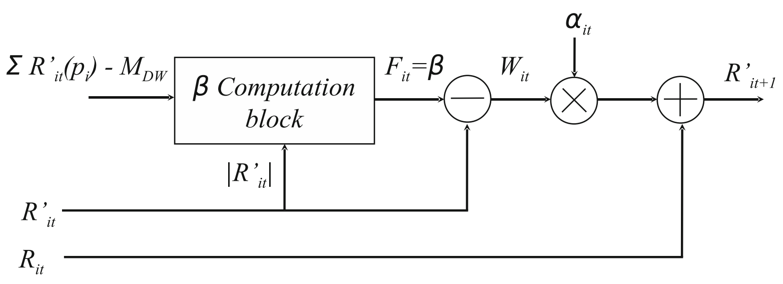


Fig. 4.15 Soft-output computation stage

4.7 Comparison with Existing TPC Decoders

4.7.1 *Logic Synthesis Results of a BCH(32,26) SISO Decoder*

In Sect. 4.4, we demonstrated that exploiting symbol parallelism is efficient if $A_{DEC}(P_{sym} = p) < p \times A_{DEC}(P_{sym} = 1)$. In order to verify this inequality, we compare one parallel ($P_{sym} = n$) BCH SISO decoder *vs* $n\times$ sequential ($P_{sym} = 1$) SISO decoders. Five versions of the BCH(32,26) parallel SISO decoder that have from one to five pipeline stages were designed. The one-pipeline stage version is a fully combinational architecture with register banks only at the input and output stages. Table 4.3 summarizes logic synthesis results of the five different parallel SISO decoders and compares them with $n = 32$ duplicated sequential SISO decoders. s is the number of pipeline stages inserted in the SISO decoder, $f_{\max}$ is the maximum frequency reached during logic syntheses. The throughput T is calculated such that $T = P \times f_{max}$, A represents the area of the design in equivalent gate count, and E is the efficiency: $E = \frac{T}{A}$. Logic syntheses were performed using Synopsys Design Compiler with an ST-microelectronics 90 nm CMOS process. The area is transposed in logic gate count. One equivalent logic gate corresponds to the area of a two-input NAND gate. It enables a more technology-independent measure of the hardware complexity.

As expected, the maximum frequency of the combinational decoder ($s = 1$) is lower than a sequential version. However, by inserting pipeline stages inside the combinatorial structure, an equivalent frequency is reached with $s = 5$. For this last version, the throughput is even higher than n sequential SISO decoders. The hardware cost of the pipeline stages insertion depends on registers location in the decoder architecture. This is the reason why $A(s = 4) < A(s = 3)$. In this particular case, having $s = 4$ pipeline stages enables register stages to be assigned at regular intervals, for a lower hardware cost. In terms of efficiency, a parallel SISO decoder can reach the same throughput as n sequential SISO decoders with a six times lower complexity. The efficiency gain increases with s.

These synthesis results demonstrate the higher efficiency of parallel SISO decoding for the code BCH(32,26). Now, if one considers larger code with the

Table 4.3 Comparison of parallel and sequential BCH(32,26) SISO decoder performance

	Parallel SISO decoder ($P_{sym} = 32$)					32 sequential SISO decoders ($P_{sym} = 1$)
s	1	2	3	4	5	3
$f_{\max}$ (MHz)	125	333	500	500	714	700
T (Gb/s)	4.0	10.7	16.0	16.0	22.9	22.4
A (Kgates)	18	26	31	26	34	200
E (Mb/s/gate)	0.15	0.27	0.34	0.41	0.44	0.07
G_E	2.1	3.9	4.9	5.9	6.3	1

same correction power (i.e., BCH(64,57), BCH(128,120)), the complexity of the reception stage and the soft-output computation stage would grow linearly with the code size n. However the complexity of the test pattern processing stage would only increase linearly with $p < n$. Consequently, the overall complexity of the parallel SISO decoder is lower than a duplication of n sequential SISO decoders. It confirms that a fully parallel SISO decoder enables a better reuse of computation and memory resources and makes the whole TPC decoder more efficient.

One should notice that, for higher correction power ($t > 1$), the algebraic decoding requires more complex algorithms such as Berlekamp–Massey algorithm [19, 20] which make the decoder complexity significantly higher. This is the reason why $t = 1$ BCH codes were selected is this study.

4.7.2 Comparison with Existing TPC Decoder Architectures

Table 4.4 compares hardware performance of existing TPC decoders architectures in an ultra-high-throughput context ($T > 10\,\mathrm{Gb/s}$). For each architectural solution, the decoder main features, the targeted code, the levels of parallelism that were used in order to reach $T = 10\,\mathrm{Gb/s}$, the resulting total parallelism ($P_{total} = \prod_i P_i$), the maximum number of iteration it_{max} are given. We consider that one iteration is actually implemented. The resulting throughput is $T = P_{total} \times f_{max}/it_{max}$. Finally, the gate count (A), the efficiency ($E = T/A$), and the achieved coding gain at BER=10^{-9} are given. Such a low BER is usually targeted in very high speed application (e.g., data transmission over Passive Optical Networks).

For a fair comparison, architectures described in [7, 13, 14, 16] were synthesized with the same technology: ST Microelectronics, CMOS 90 nm with a clock frequency $f_{max} = 500\,\mathrm{MHz}$. For the remaining architectures, we gathered information from the published papers and technical reports.

Two versions of the P_{sym}-based TPC decoder were synthesized. The first one consists in four parallel SISO decoders together with 32 $P_{sym} = 4$-SISO decoders. The reached throughput is then sufficient for 10 Gb/s applications. The second version uses only fully parallel SISO decoder, 32 of such decoders are duplicated for each half-iteration. The maximum throughput is 85 Gb/s for the best efficiency. This architecture uses row-wise P_{sb} and column-wise P_{sym}. The barrel-shifter-based solution [13] can achieve 10 Gb/s with 2.6 Mgates. In order to reach a sufficient parallelism level, it was necessary to use frame parallelism. The efficiency of this approach is six times lower than the P_{sym}-based TPC decoders. This low efficiency is mainly due to the use of interleaving memory.

For the same reason, the TPC decoder with multi-access data [16] has a low efficiency and also requires the use of frame parallelism to achieve 10 Gb/s.

In [14], the elimination of interleaving memories improves the efficiency but the maximum parallelism rate is limited by the code size n. This makes the use of frame parallelism mandatory in an ultra high speed context.

Table 4.4 Comparisons of state-of-the-art TPC decoders

Decoder features	Code	P_i	P_{total}	it_{max}	T (Gb/s)	Area (Mgates)	E (Kb/s/gate)	Coding gain (dB) @BER=10^{-9}
This work	BCH$(32,26)^2$	$P_{sym} = 4, P_{sb} = 32$	128	6	10.7	0.4	26.8	8.0
	BCH$(32,26)^2$	$P_{sym} = 32, P_{sb} = 32$	1024	6	85.3	2.0	42.7	8.0
Barrel shifter + IM [13]	BCH$(32,26)^2$	$P_{sb} = 32$, $P_{frame} = 4$	128	6	10.7	2.6	4.1	8.4
Omega network + no IM [14]	BCH$(32,26)^2$	$P_{sb} = 32$, $P_{frame} = 4$	128	6	10.7	1.6	6.7	8.4
	BCH$(64,57)^2$	$P_{sb} = 64$, $P_{frame} = 2$	128	6	10.7	2.0	5.4	8.6
	BCH$(128,120)^2$	$P_{sb} = 128$	128	6	10.7	2.7	4.0	8.7
Multi-data access IM [16]	BCH$(32,26)^2$	$P_{sym} = 8$, $P_{sb} = 8$, $P_{frame} = 2$	128	6	10.7	3.5	3.1	8.4
Omega network + no IM [7]	RS$(15,13)^2$	$P_{is} = 4$, $P_{sb} = 15$, $P_{frame} = 2$	120	6	10	0.3	33.3	8.4
	RS$(31,29)^2$	$P_{is} = 5$, $P_{sb} = 31$	155	6	12.9	0.8	16.1	8.4
	RS$(63,61)^2$	$P_{is} = 3$, $P_{sb} = 63$	378	6	15.8	1.3	12.1	7.5
Commercial RS decoder (ASICS ws)	RS(255,239)	$P_{is} = 8, P_{frame} = 4$	32	X	10.7	0.12	89	5.0
Omega network + no IM [7]	RS$(31,29)^2$	$P_{is} = 5$, $P_{sb} = 31$	155	1	35	0.4	95	5.2
Commercial TPC decoder (Mitsubishi)	BCH(144,128) xBCH(256,239)	?	?	4	10.0	18.0	0.6	10

The study in [7] shows that RS-TPC are a practical solution for 10 Gb/s transmission over optical networks. As we mentioned in Sect. 4.4, using RS codes enables the use of intra-symbol parallelism. With an omega-network-based architecture, this decoder also presents good efficiency gain for similar decoding performance. One should notice that the P_{sym}-based fully parallel architecture is applicable to RS decoding as well. We expect that the application of intra-symbol parallelism would further increase the overall efficiency of the TPC decoder. Moreover, when comparing a single iteration of RS-TPC decoding with a commercial RS(255,239) code decoder, one can observe that superior efficiency is achieved for slightly better decoding performance.

Mitsubishi proposed a TPC decoder for 10 Gb/s optical transmissions. The component code is a BCH(144,128)xBCH(256,239). These codes are more powerful than $t = 1$ BCH codes that are used in this study. However the implementation is very costly in terms of hardware complexity. Indeed, 18 Mgates are necessary to implement such a decoder, which makes the efficiency very small. This is the cost that has to be paid for a 2 dB extra coding gain provided by this TPC decoder.

Conclusion

TPC decoding is a realistic solution for next generation high throughput optical communications such as long-haul optical transmissions or passive optical networks. The structure of the product codes makes them very suitable for parallelization. However the exploitation of some parallelism levels may not be efficient in terms of throughput/complexity ratio. This is particularly true when interleaving memory has to be duplicated.

In this chapter, we review and characterize all parallelism levels in TPC decoding. This analysis helps to better understand and classify existing TPC decoders. In these TPC decoders, high throughput architecture complexity is made prohibitive by the amount of memory usually required for data interleaving and pipelining.

After this design space exploration, we focus on an architecture that jointly exploits sub-block parallelism and symbol parallelism. This structure enables any interleaving resource to be removed. This TPC decoder requires a fully parallel SISO decoder capable of processing n symbols in one clock period. Such a SISO decoder architecture is described and includes an optimized parallel sorting network.

ASIC-based logic syntheses confirm the better efficiency of the IM-free TPC decoder architecture compared to others. Actually, when compared to other works, the area is reduced while the same throughput is achieved. A BCH(32,26)^2 product code can be decoded at 33.7 Gb/s with an estimated silicon area of 10 μm^2 in 65 nm CMOS technology.

References

1. Akita M, Fujita H, Mizuochi T, Kubo K, Yoshida H, Kuno K, Kurahashi S (2002) Third generation fec employing turbo product code for long-haul dwdm transmission systems. In: Optical fiber communication conference and exhibit, 2002 (OFC 2002), 17–22 March 2002, pp 289–290
2. Pyndiah R, Glavieux A, Picart A, Jacq S (1994) Near optimum decoding of product codes. In: IEEE Global Telecommunications Conference, 1994 (GLOBECOM '94)
3. Elias P (1954) Error-free coding. IEEE Trans Inf Theory 4(4):29–37
4. Forney GJ (1966) Generalized minimum distance decoding. IEEE Trans Inf Theory IT-12:125–131
5. Adde P, Pyndiah R, Raoul O (1996) Performance and complexity of block turbo decoder circuits. In: Proceedings of the third IEEE international conference on electronics, circuits, and systems, 1996 (ICECS '96), vol 1, 13–16 October 1996, pp 172–175
6. IEEE Standard for Local and Metropolitan Area networks (2001) Part 16: Air interface for fixed broadband wireless access systems, December 2001
7. Bidan RL, Leroux C, Jego C, Adde P, Pyndiah R (2008) Reed-solomon turbo product codes for optical communications: from code optimization to decoder design. EURASIP J Wirel Commun Netw 2008:909–912
8. Leroux C, Jego C, Adde P, Jezequel M (2008) On the higher efficiency of parallel reed-solomon turbo-decoding. In: ICECS'08: 15th international conference on electronics, circuits and system, 31st August - 3rd September, 2008
9. Chase D (1972) A class of algorithms for decoding block codes with channel measurement information. IEEE Trans Inf Theory IT:170–182
10. Leroux C, Jego C, Adde P, Jezequel M (2007) Towards Gb/s turbo decoding of product code onto an FPGA device. In: IEEE international symposium on circuits and systems, 2007 (ISCAS 2007), 27–30 May 2007, pp 909–912
11. Adde P, Pyndiah R (2000) Recent simplifications and improvements in block turbo codes. In: 2nd international symposium on turbo codes & related topics, Brest, France, 4–7 September 2000, pp 133–136
12. Leroux C, et al. (2011) Turbo product code decoder without interleaving resource: from parallelism exploration to high efficiency architecture. J Signal Process Syst 64(1):17–29
13. Chi Z, Parhi K (2002) High speed VLSI architecture design for block turbo decoder. In: IEEE international symposium on circuits and systems, 2002 (ISCAS 2002), vol 1, 26–29 May 2002, pp I-901–I-904
14. Jego C, Adde P, Leroux C (2006) Full-parallel architecture for turbo decoding of product codes. In: Electronics letters, vol 42, 31 August 2006, pp 55–56
15. Leroux C, et al. (2009) High-throughput block turbo decoding: from full-parallel architecture to FPGA prototyping. J Signal Process Syst 57(3):349–361
16. Cuevas J, Adde P, Kerouedan S, Pyndiah R (2002) New architecture for high data rate turbo decoding of product codes. In: IEEE global telecommunications conference, 2002 (GLOBECOM '02), vol 2, 17–21 November 2002, pp 1363–1367
17. Piriou E, Jego C, Adde P, Le Bidan R, Jezequel M (2006) Efficient architecture for Reed Solomon block turbo code. In: Proceedings of the IEEE international symposium on circuits and systems, 2006 (ISCAS 2006), 21–24 May 2006, 4 pp
18. Akl SG (1985) Parallel sorting algorithms. Academic, New York
19. Berlekamp ER (1968) Algebraic coding theory. vol 111, New York, McGraw-Hill
20. Massey JL (1969) Shift-register synthesis and bch decoding. IEEE Trans Inf Theory IT:122–127