High speed architectures for finding the firsttwo maximum/minimum values

High speed architectures for finding the first two maximum/minimum values

Luca G. Amarù, Maurizio Martina, *Member IEEE*, Guido Masera, *Senior Member IEEE*

Abstract

High speed architectures for finding the first two maximum/minimum values are of paramount importance in several applications, including iterative (e.g. turbo and LDPC) decoders. In this brief, stemming from a previous work, based on radix-2 solutions, we propose higher and mixed radix implementations that improve the architecture latency. Post place and route results on a 180 nm CMOS standard cell technology show that the proposed architectures achieve lower latency than radix-2 solutions with a moderate area increase.

Index Terms

Turbo decoder, LDPC decoder, minimum values generator, tree structure approach.

I. Introduction

Recently, several simplified algorithms for channel decoding have been proposed for Low-Density-Parity-Check (LDPC) [1], and turbo codes [2]. The λ-min algorithm [3], min-sum and its improved versions [4], are widely used in LDPC decoders, [5], [6]. Similarly, in [7] a novel multi-input $\max^*$ approximation for turbo and turbo Trellis-Coded-Modulation (TCM) coding is proposed. All these works share the need for finding the first two maximum/minimum (max/min) values in a set of M elements. As an example, in the min-sum algorithm [4] the magnitude of the i-th output of a check node having degree d_c is given by the first min of the d_c inputs' magnitudes, unless this equals the i-th input's magnitude, in which case the second min is employed. In [7], the Jacobian logarithm of n inputs is computed as the first max (*max1*) plus a correction term that depends on *max1-max2* where *max2* is the second max. A similar problem can be found also in K-best Multiple-Input-Multiple-Output (MIMO) detectors [8]–[10], non-binary LDPC decoders [11] and Turbo Product Codes [12], [13] where the computation of the first W max/min values is required. For the sake of brevity the extension to the search of the first W max/min values is not investigated in this brief.

All the architectures proposed in [14]–[16] for finding the first two max/min values are based on radix-2 tree structures or a proper blend of radix-2 and radix-3 blocks. Even if these architectures are remarkably small in terms of area, they require a relevant number of stages (especially for large M) which negatively affects the delay. However, some applications as iterative decoders (e.g. turbo codes [2], [7]), require high throughput and include feedback loops in the processing: in these cases pipelining does not provide any throughput improvement and different solutions are necessary.

In this brief, stemming from [16] we analyze the implementation of a generic radix-K solution and we further extend the design space considering Mixed Radix Architectures (MRAs). In [17] a similar work is presented. However, it focuses only on using binary and trinary trees. On the contrary, this work offers a systematic treatment including MRAs and, to the best of our knowledge, it is the first work addressing MRAs for finding the first two max/min values.

II. Finding the first two max/min values

A. Problem formulation

Given a set $\mathcal{X}^M = \{x_0, \ldots, x_{M-1}\}$ made of M elements we want to find the first two max/min values, namely $y_0^M = \max(\mathcal{X}^M)$ and $y_1^M = \max(\mathcal{X}^M \setminus \{y_0^M\})$ (similarly substituting $\max$ with $\min$). For the sake of simplicity in the following we will discuss only the $\max$ case as the $\min$ equivalent solution can be straightforwardly derived. According to the tree-based solution proposed in [16], the procedure to find the first two max values out of M assigned ones can be decomposed in $\log_2(M)$ levels where level l contains $M/2^l$ Comparing Stages (CSs) with $1 \leq l \leq \log_2(M)$. CSs at the first level ($l = 1$) receive two input values and sort them, so each CS is made of one comparator. Each CS at higher levels ($l > 1$) receives two couples of sorted values from the previous level and outputs one sorted couple. If we approximate the delay of a CS with the delay of a comparator, we can infer that the delay of such a structure is $O(\log_2(M))$. This assumption will be used along this section. However, in section IV we will give area and delay results obtained via real implementation of the proposed architectures. On the other hand, we can implement a radix-M solution that is able to find the first two max values with a delay $O(1)$. This reduced delay is paid in terms of complexity as we need to compare every input value with each other indeed. Thus, the total number of comparators (avoiding duplicating comparisons) is $M + (M-1) + \cdots + 1 = M \cdot (M-1)/2$.

B. Fixed Radix Architecture (FRA)

A delay $O(\log_K(M))$ is obtained by using a tree structure made of radix-K CSs with $K < M$ (Fig. 1). At $l = 1$ there are M/K concurrent CSs, each of which finds the first two max values out of its K inputs with a delay $O(1)$. Thus, each of these CSs contains $K \cdot (K-1)/2$ comparators working concurrently. The total number of comparators at $l = 1$ is

$$C_{l=1} = [M \cdot (K-1)]/2. \tag{1}$$

Then, we define $\mathcal{X}^{K_1}[i]$ the set of K elements processed by the i-th CS ($0 \leq i \leq M/K - 1$) at $l = 1$ ($\bigcup_{i=0}^{M/K-1} \mathcal{X}^{K_1}[i] = \mathcal{X}^M$) and $y_0^{K_1}[i] = \max(\mathcal{X}^{K_1}[i])$, $y_1^{K_1}[i] = \max(\mathcal{X}^{K_1}[i] \setminus \{y_0^{K_1}[i]\})$. For $1 < l \leq \log_K(M)$, the i-th CS receives K couples $y_0^{K_{l-1}}[i \cdot K + j], y_1^{K_{l-1}}[i \cdot K + j]$ with $0 \leq i \leq M/K^l - 1$ and $0 \leq j \leq K - 1$. Inspired by [16] we can infer that $y_0^{K_l}[i]$, the first max output by the i-th CS at level l, is obtained comparing the K *max1* values from the previous level, $y_0^{K_{l-1}}[i \cdot K + j]$. Thus, $K \cdot (K-1)/2$ comparators are required to compute $y_0^{K_l}[i]$. Similarly, to compute $y_1^{K_l}[i]$ we need to compare $y_0^{K_{l-1}}[i \cdot K + p]$ and $y_1^{K_{l-1}}[i \cdot K + q]$ with $0 \leq p, q \leq K - 1$

and $p \neq q$. As a consequence, to complete the computation of $y_1^{K_l}[i]$ further $K \cdot (K-1)$ comparators are required, as detailed in section III-B. The total number of comparators for $1 < l \leq \log_K(M)$ is

$$C_{l>1} = \sum_{l=2}^{\log_K(M)} \frac{M}{K^l} \cdot \frac{3K \cdot (K-1)}{2} = \frac{3(M-K)}{2}. \tag{2}$$

From (1) and (2) the total number of comparators is

$$C = C_{l=1} + C_{l>1} = (M \cdot K + 2M - 3K)/2. \tag{3}$$

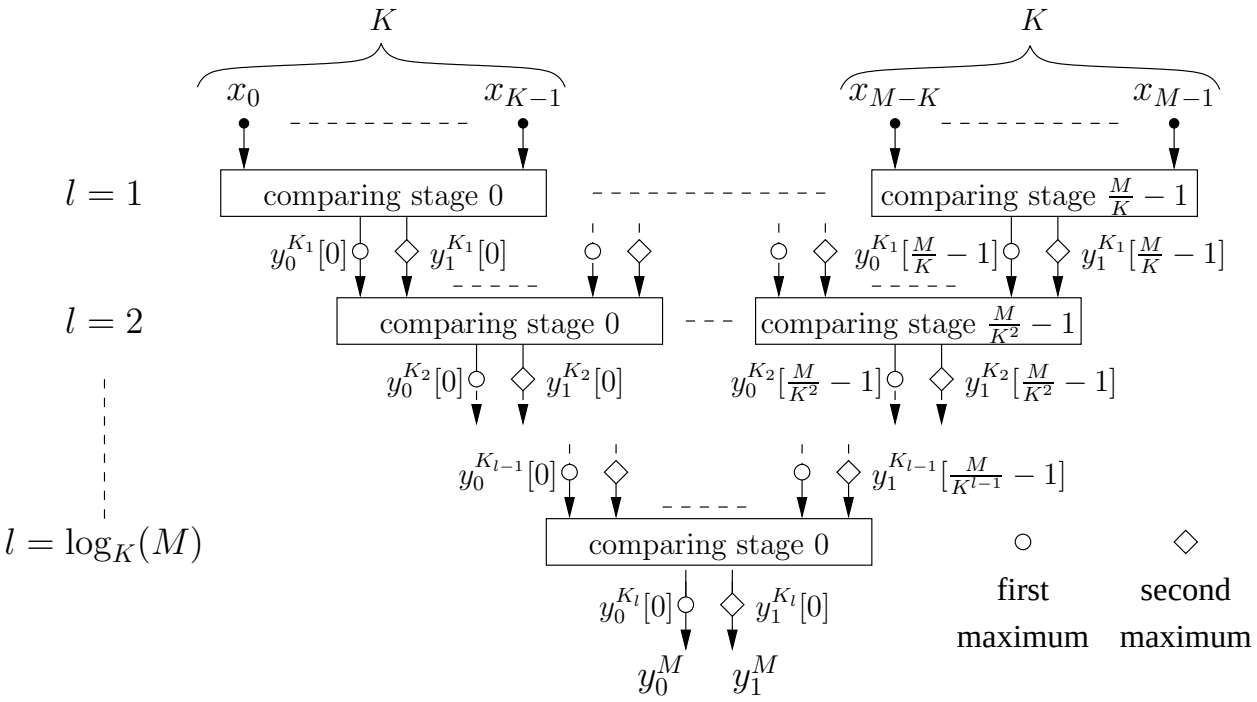

Figure 1. Radix-K tree structure

C. Mixed Radix Architecture

The solution detailed in the previous paragraphs can be applied as is only when $\log_K(M) \in \mathbb{N}$. However, several cases of practical interest where M is not a power of K can give better area/latency trade-offs. To that purpose we propose to use MRA, namely different levels in the tree use different radix. In the following we will refer to this solution as InteR-level-MRA (IR-MRA). Further flexibility could be achieved by using a different radix for each CS at level l, namely, $K_l[i]$ is the radix of the i-th CS at level l. This solution will be referred to as IntrA-level-MRA (IA-MRA).

1) IR-MRA: If N is the number of levels we have

$$C = \frac{M \cdot (K_1 - 1)}{2} + \frac{3M}{2} \sum_{n=2}^{N} \frac{K_n - 1}{\Phi_n(1)} \tag{4}$$

with $\Phi_n(\rho) = \prod_{i=\rho}^{n-1} K_i$ and K_l the radix at level l. Imposing

$$M = \prod_{l=1}^{N} K_l \qquad M, K_l \in \mathbb{N}_+ \tag{5}$$

we can find several arrays $\mathbf{K}_N = \{K_1, \ldots, K_N\}$ that satisfy (5), where the elements of $\mathbf{K}_N$ are taken from the set of the dividers of M. To that purpose we introduce $\mathcal{D}_N$ as the set of all the arrays $\mathbf{K}_N$ that satisfy (5) and δ_N, the cardinality of $\mathcal{D}_N$. If we impose that M is a power of two and we take the logarithm of (5) we obtain $\gamma = \log_2(M) = \sum_{l=1}^{N} \log_2(K_l)$. As a consequence, the problem of finding δ_N simplifies to finding the set of N positive integers ($\log_2(K_l)$) whose sum is γ. Thus, we obtain $\delta_N = \binom{\gamma-1}{N-1}$. As an example, for $M = 32$ and $N = 3$ there are six possible IR-MRA ($\delta_3 = 6$). To find the solution that requires the minimum C we consider (4) and impose $\partial C / \partial K_l = 0$ for each l:

$$K_1 = \sqrt{3 \sum_{n=2}^{N} \frac{K_n - 1}{\Phi_n(2)}} \qquad K_l|_{1<l<N} = \sqrt{\sum_{n=l+1}^{N} \frac{K_n - 1}{\Phi_n(l+1)}} \tag{6}$$

with (5) as a constraint, so that K_N is chosen to satisfy (5).

We can rewrite indeed (6) as

$$K_1 = \sqrt{3(K_2 - 1)} \qquad \text{if} \quad N = 2 \tag{7}$$

$$\left.\begin{aligned} K_1 &= \sqrt{3(2K_2 - 1)} \\ K_l &= \sqrt{2K_{l+1} - 1} \quad 1 < l < N-1 \\ K_{N-1} &= \sqrt{K_N - 1} \end{aligned}\right\} \quad \text{if} \quad N > 2. \tag{8}$$

Finally, by substituting (7) and (8) in (5) we obtain

$$M = K_2 \cdot \sqrt{3(K_2 - 1)} \tag{9}$$

$$M = \sqrt{3(2K_2 - 1)} \cdot \prod_{l=2}^{N-2} \sqrt{2K_{l+1} - 1} \cdot \sqrt{K_N - 1} \cdot K_N. \tag{10}$$

Unfortunately, both (9) and (10) can be written as polynomial Diophantine equations with integer coefficients that do not always admit solutions in $\mathbb{N}$ [18]. Usually in IR-MRAs δ_N is of the order of few tens; thus, $\mathcal{D}_N$ can be explored exhaustively as shown in section IV.

2) IA-MRA: The formalization proposed in section II-C1 to compute C can be extended to the case of IA-MRA as

$$C = \sum_{l=1}^{N} C_l \qquad C_l = \sum_{i=0}^{O_l} \alpha_l \cdot K_l[i] \cdot (K_l[i] - 1) \tag{11}$$

where O_l is the number of CSs at level l and $\alpha_{l=1} = 1/2$, $\alpha_{l>1} = 3/2$. This implies that (5) should be rewritten as

$$M = \prod_{l=1}^{N} \overline{K}_l \qquad M \in \mathbb{N}_+ \qquad \overline{K}_l = \frac{1}{O_l} \sum_{i=0}^{O_l} K_l[i] \tag{12}$$

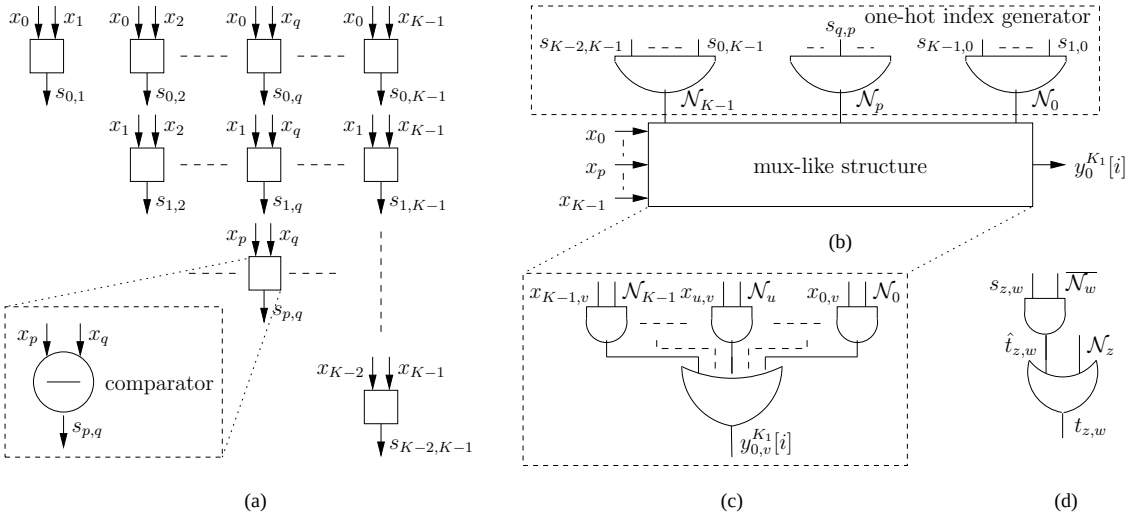

Figure 2. $l = 1$ CS architecture (radix-K): array of comparators (a), one-hot index generator for $\mathcal{N}$ (b), 1-bit mux-like structure (c), generation of $t_{z,w}$ (d)

where $\overline{K}_l$ is the average radix at level l. The minimization of (11) is Diophantine, as for the IR-MRA case, as a consequence, it is not always possible to explicitly find optimal solutions. Even if δ_N tends to be large, we explored exhaustively $\mathcal{D}_N$, as for the IR-MRA case. Experimental results show that in different cases IR-MRAs and IA-MRAs achieve the same C for a given N. For the sake of brevity, we will concentrate on IR-MRAs. However, in section IV we will show IA-MRAs results when they have lower complexity than the corresponding IR-MRA.

III. Architectural description

Each CS is made of three main parts: an array of comparators, some One-Hot Index Generators (OHIGs), and some Mux-Like Structures (MLSs).

A. Level $l = 1$ comparing stages

Note that in this section we consider input values belonging to one CS, namely x_j with $0 \leq j \leq K-1$. Thus, indices do not correspond to the ones given in section II when $\mathcal{X}^M$ and $\mathcal{X}^{K_1}[i]$ were defined.

Let us define x_p and x_q as two inputs out of the possible K ones, $s_{p,q}$ as the sign of $x_p - x_q$ and $s_{q,p} = \overline{s_{p,q}}$ where $(\bar{\cdot})$ is the one-complement operator (see Fig. 2 (a))[1]. If input x_n is *max1* then $s_{q,n} = 1$ for every q such that $0 \leq q \leq K-1$ and $q \neq n$. Now we build an array $\mathcal{N}$ containing K elements, where the p-th element is

$$\mathcal{N}_p = \bigwedge_{q=0, q\neq p}^{K-1} s_{q,p} \tag{13}$$

and $\bigwedge$ stays for the logic-and operation. As it can be observed, if n is the index of *max1* of the i-th CS, namely $n = \arg(\max(\mathcal{X}^{K_1}[i]))$, $\mathcal{N}$ is the One-Hot (OH) binary representation of n. Finally, $\mathcal{N}$ is used as the selection

[1]If $x_p = x_q$, $s_{p,q}$ is not relevant as choosing either x_p or x_q is the same.

signal of an MLS (see Fig. 2 (b)). According to (13) we concurrently compute all the bits of the OHIG resorting to K *and-gates* each of which receives $K-1$-input $s_{q,p}$ signals (see the dashed box in Fig 2 (b)). The MLS can be described as follows. Given $x_u \in \mathcal{X}^{K_1}[i]$, $0 \leq u \leq K-1$ and being $x_{u,v}$ the v-th bit of x_u we obtain $y_{0,v}^{K_1}[i]$, the v-th bit of $y_0^{K_1}[i]$, as

$$y_{0,v}^{K_1}[i] = \bigvee_{u=0}^{K-1} x_{u,v} \wedge \mathcal{N}_u \tag{14}$$

where $\bigvee$ is the logic-or operation. According to (14) a 1-bit MLS is made of a K-input *or-gate* and K 2-input *and-gates* (see Fig. 2 (c)). As a consequence, we concurrently obtain each bit of $y_0^{K_1}[i]$ by replicating the 1-bit MLS.

A similar approach is used to obtain $y_1^{K_1}[i]$, namely combining $s_{z,w}$ signals with a *blinding* circuit that acts as a mask for the position of *max1*[2]. Let us define $\hat{t}_{z,w} = s_{z,w} \wedge \overline{\mathcal{N}_w}$ where $\hat{t}_{z,w} = 1$ when $x_w \geq x_z$ and $x_w \neq y_0^{K_1}[i]$. If input x_n is *max1* $s_{n,w} = 0$ and $\hat{t}_{n,w} = 0$. As a consequence, to identify *max2* we can not simply compute $\hat{\mathcal{M}}_w = \bigwedge_{z=0,z\neq w}^{K-1} \hat{t}_{z,w}$ as for *max1*. We can avoid this problem by introducing $t_{z,w} = \hat{t}_{z,w} \vee \mathcal{N}_z$: if input x_n is *max1*, $\mathcal{N}_n = 1$ and $t_{n,w} = 1$ (see Fig. 2 (d)). We can now build a second array $\mathcal{M}$ that contains K elements, where the w-th element is

$$\mathcal{M}_w = \bigwedge_{z=0,z\neq w}^{K-1} t_{z,w}. \tag{15}$$

As for *max1*, m is the index of *max2* in the i-th CS, $m = \arg(y_1^{K_1}[i] = \max(\mathcal{X}^{K_1}[i] \setminus \{y_0^{K_1}[i]\})$, $\mathcal{M}$ is the OH binary representation of m and the v-th bit of $y_1^{K_1}[i]$ is obtained as

$$y_{1,v}^{K_1}[i] = \bigvee_{u=0}^{K-1} x_{u,v} \wedge \mathcal{M}_u. \tag{16}$$

According to (15) and (16), $y_1^{K_1}[i]$ is obtained with an OHIG and an MLS, (Fig. 2 (b) and (c)) by substituting $s_{q,p}$ and $\mathcal{N}$ with $t_{z,w}$ and $\mathcal{M}$ respectively.

B. level $l > 1$ comparing stages

A radix-K CS at $l > 1$ computes the first two max values among its K input couples $y_0^{K_{l-1}}[i \cdot K + j], y_1^{K_{l-1}}[i \cdot K + j]$, $0 \leq i \leq M/K^l - 1$, $0 \leq j \leq K-1$. The *max1* output by the i-th CS ($y_0^{K_l}[i]$) is obtained by processing the K input values $y_0^{K_{l-1}}[i \cdot K + j]$ with the same architecture used for $l = 1$.

Let us define $\mathcal{Y}_0^{K_{l-1}}[i]$ and $\mathcal{Y}_1^{K_{l-1}}[i]$ as the two sets containing $y_0^{K_{l-1}}[i \cdot K + j]$ and $y_1^{K_{l-1}}[i \cdot K + j]$ values respectively (each set contains K values). As far as *max2* is concerned, two cases are possible: i) $y_1^{K_l}[i] \in \mathcal{Y}_0^{K_{l-1}}[i]$, namely $y_1^{K_l}[i]$ is one of the *max1* received as inputs. ii) $y_1^{K_l}[i] \in \mathcal{Y}_1^{K_{l-1}}[i]$, $y_1^{K_l}[i]$ is one of the *max2* values. Case i) can be solved with the same architecture used for $l = 1$. On the other hand, case ii) requires to compare $y_0^{K_{l-1}}[i \cdot K + p]$ and $y_1^{K_{l-1}}[i \cdot K + q]$ with $0 \leq p, q \leq K-1$ and $p \neq q$. Let us define $y_0^{K_{l-1}}[a]$ and $y_1^{K_{l-1}}[b]$ with $a \neq b$ as one element of $\mathcal{Y}_0^{K_{l-1}}[i]$ and $\mathcal{Y}_1^{K_{l-1}}[i]$ respectively. Then, the sign of $y_0^{K_{l-1}}[a] - y_1^{K_{l-1}}[b]$ is $s_{a,b}^{0,1}$. If f is

[2]The formulation proposed in the following should ease understanding the underlying idea even if from a formal point of view it has some redundancy.

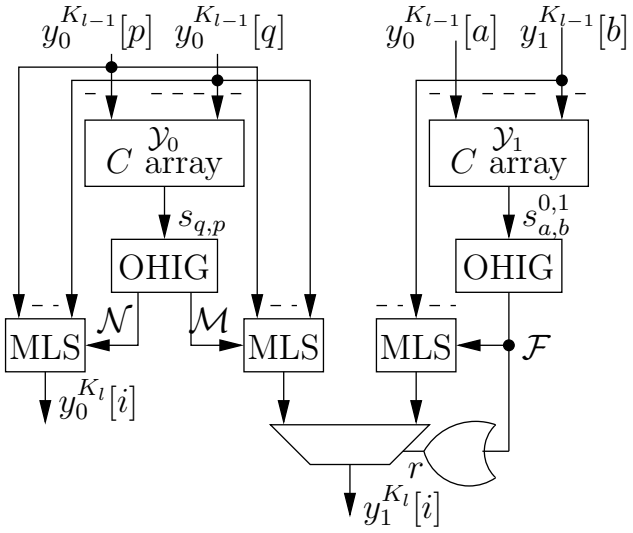

Figure 3. $l > 1$ CS architecture (radix-K)

the index of the element in $\mathcal{Y}_1^{K_{l-1}}[i]$ that is *max2* of the i-th CS, $\mathcal{F}$ is its OH binary representation, where the b-th element is $\mathcal{F}_b = \bigwedge_{a=0,a\neq b}^{K-1} s_{a,b}^{0,1}$. A further MLS selects $y_1^{K_l}[i]$ as the f-th element of $\mathcal{Y}_1^{K_{l-1}}[i]$. To complete the architecture (see Fig. 3) we need to infer if $y_1^{K_l}[i]$ belongs to $\mathcal{Y}_0^{K_{l-1}}[i]$ or to $\mathcal{Y}_1^{K_{l-1}}[i]$. This selection is accomplished by observing that if $y_1^{K_l}[i] \in \mathcal{Y}_1^{K_{l-1}}[i]$ then $r = \bigvee_{b=0}^{K-1} \mathcal{F}_b = 1$.

Table I
FRA POST SYNTHESIS RESULTS: A [μM^2] AND L [NS]

	$M=8$	$M=16$	$M=32$	$M=64$
$K_l=2$	11026/1.7	23105/2.2	51613/2.6	97906/3.2
$K_l=4$	-	23904/1.7	-	115310/2.4
$K_l=8$	16494/1.2	-	-	172474/2.3
$K_l=16$	-	70250/1.5	-	-
$K_l=32$	-	-	263270/1.8	-
$K_l=64$	-	-	-	819798/2.3

IV. EXPERIMENTAL RESULTS

The analytical approach proposed in section II can be used to identify the set of solutions with minimum C. This strategy is useful when the design space tends to be large, as in the IA-MRA case. In Table II area and latency for a 180 nm CMOS standard cell technology of one comparator (data represented on six bits) and one radix-2 CS at $l = 1$ and $l > 1$ are shown; the complexity of the comparator(s) is about half the complexity of the CS. Moreover, data in Table II allow for a reasonable estimation of the area and latency of radix-2 architectures summarized in Table I as $A_M^{(2)} = A_{C_{l=1}}^{(2)} \cdot \frac{M}{2} + A_{C_{l>1}}^{(2)} \cdot \left(\frac{M}{2} - 1\right)$ and $L_M^{(2)} = L_{C_{l=1}}^{(2)} + (N-1) \cdot L_{C_{l>1}}^{(2)}$. Similar formulas can be obtained to estimate A and L for higher radix and MRAs.

Table II

COMPARATOR AND RADIX-2 CS AT $l = 1$ AND $l > 1$ POST SYNTHESIS RESULTS: A [μM^2] AND L [PS]

comparator	CS @ $l = 1$	CS @ $l > 1$
A_C/L_C	$A^{(2)}_{C_{l=1}}/L^{(2)}_{C_{l=1}}$	$A^{(2)}_{C_{l>1}}/L^{(2)}_{C_{l>1}}$
467/400	880/500	2503/600

Table III

POST P&R FRA AND MRA RESULTS: A [μM^2] AND L [NS]

	FR (K_l)	MR N=2 (K_1/K_2)			MR N=3 ($K_1/K_2/K_3$)		
M=32		(16/2)	(2/16)	(4/8)	(4/4/2)	**(4/2/4)**	(2/4/4)
	(2)	157643/2.0	281772/2.0	96211/2.0	75780/2.2	**72651/2.2**	99709/2.2
	73355/2.6		**(8/4)**		(8/2/2)	(2/8/2)	(2/2/8)
			94130/2.0		929011/2.2	155873/2.2	121188/2.2

	FR (K_l)	MR N=4 ($K_1/K_2/K_3/K_4$)					FR (K_l)	MR N=3 ($K_1/K_2/K_3$)		
M=64		(4/4/2/2)	**(4/2/4/2)**	(4/2/2/4)	(2/4/2/4)	M=24		(6/2/2)	(2/6/2)	(2/2/6)
	(2)	155423/2.6	**150217/2.6**	151835/2.6	190267/2.6			56451/2.0	94163/2.0	74957/2.0
	137261/3.2	(2/4/4/2)		(2/2/4/4)			(2)	(2/3/4)	(2/4/3)	(3/2/4)
	(4)	206287/2.6		178323/2.6			50868/2.5	64922/2.0	73040/2.0	56480/2.0
	160219/2.4	(8/2/2/2)	(2/8/2/2)	(2/2/8/2)	(2/2/2/8)			(3/4/2)	**(4/2/3)**	(4/3/2)
		184369/2.6	304103/1.6	221290/2.6	194228/2.6			58376/2.0	**51560/2.0**	55276/2.0

To highlight the area/latency trade-off we define $\eta^*_M = (A^*_M \cdot L^*_M)/(A^{(2)}_M \cdot L^{(2)}_M)$ where A^*_M and L^*_M are the area and the latency of a generic radix architecture with M inputs. An architecture that halves the latency at the expense of doubling the area with respect to the radix-2 solution has $\eta^*_M = 1$. Thus, architectures of particular interest are the ones with $\eta^*_M < 1$.

Results shown in Table I highlight that FRAs do not permit to tune the A/L trade-off: for $M = 32$ there are only two solutions: radix-2 and radix-32. In the radix-2 solution area is 80% lower and latency 30% larger than radix-32 case. Moreover, the only FRAs with $\eta^*_M < 1$ are the radix-4 ones: $\eta^{(4)}_{16} = 0.8$ and $\eta^{(4)}_{64} = 0.88$.

To obtain more accurate results [19], VHDL developed architectures have been synthesized with Synopsys Design Compiler for shortest delay, Placed and Routed (P&R) with Cadence Encounter using a 180 nm CMOS standard cell technology at 0^{o}C and with supply voltage 1.95V.

Even if the expression of $\mathcal{M}_w$ (15) can be optimized by-hand (each element $t_{z,w}$ contains the common term $\overline{\mathcal{N}_w}$), we prefer to leave the task of logic minimization to the logic synthesizer to explore a larger space of complexity/performance trade-offs. Experimental results shown in the following for [14]–[16] have been reproduced for a fair comparison with the proposed solutions for six bit data width. We show in Table III the experimental results obtained for ($M = 32$, $N = 2$), ($M = 32$, $N = 3$), ($M = 64$, $N = 4$) and ($M = 24$, $N = 3$) respectively, where the best result is highlighted in bold. It is interesting to observe that IA-MRAs for the cases show in Table III exist. For the case $M = 32$, $N = 2$ the best solution requires $C = 130$ and 94130 μm^2. However, with four

radix-5 and two radix-6 CSs at $l = 1$ and $K_2 = 6$ we obtain $C = 115$. Even if this solution minimizes C it requires 5-input and 6-input *and/or-gates* leading to an area of 104903 μm^2, about the 10% more than the best IR-MRA. IA-MRAs for $M = 32$, $N = 3$ require in the best case $C = 78$ and 72651 μm^2, the same result obtained with $K_1 = 4$, $K_2 = 2$, $K_3 = 4$. When $M = 64$, $N = 4$ the best solution requires $C = 159$ and 150217 μm^2. However, with two radix-2 and twenty radix-3 at $l = 1$, $K_2 = 2$, one radix-2 and three radix-3 at $l = 3$ and $K_4 = 4$ we obtain $C = 143$ and $A = 145801$ μm^2, about the 3% less than the best IR-MRA.

Table IV
POST P&R EXPERIMENTAL RESULTS: A [μM^2] AND L [NS] COMPARISONS

M	[14]–[16]		Proposed		
6	[15]	[16]	$K_l = 6$	$K_1 = 2$ $K_2 = 3$	$K_1 = 3$ $K_2 = 2$
	13451/1.5	10158/1.6	13197/1.1	11329/1.4	8827/1.4
	$\eta_6^* > 1$	-	$\eta_6^* = 0.89$	$\eta_6^* = 0.98$	$\eta_6^* = 0.76$
7	[15]	[16]	$K_l = 7$	$K_1 = 4, 3$ $K_2 = 2$	
	13718/1.7	13390/1.7	15184/1.1	13472/1.3	
	$\eta_7^* > 1$	-	$\eta_7^* = 0.73$	$\eta_7^* = 0.77$	
8	[14]	[16]	$K_l = 8$	$K_1 = 2$ $K_2 = 4$	$K_1 = 4$ $K_2 = 2$
	17617/2.1	13640/1.7	23343/1.2	21938/1.4	13799/1.4
	$\eta_8^* > 1$	-	$\eta_8^* > 1$	$\eta_8^* > 1$	$\eta_8^* = 0.83$

The proposed architecture can also be employed to reduce L when M is not a power of two. As an example, if $M = 9$ a radix-2 solution imposes an unbalanced tree structure with $N = 4$. The implementation of such a structure leads to $A = 21025$ μm^2 and to $L = 2$ ns. On the other hand, with a FRA-3, corresponding to $N = 2$, we obtain $A = 14426$ μm^2 and $L = 1.6$ ns. It is worth noting that there is no IA-MRA for $M = 9$, $N = 2$ that performs better than $K_1 = K_2 = 3$. Similarly, $M = 24$ as a radix-2 solution has an unbalanced tree structure with $N = 5$. On the contrary, with $N = 3$ we have nine possible MRAs. As it can be observed, the 4/2/3 MRA improves the latency of 25% with respect to the FRA-2 and requires an area overhead of less than 1.1%. For $M = 24$, $N = 3$ IA-MRAs exist, however they require $C = 54$ as the best solution reported in Table III.

In Table IV we compare our MRAs with other approaches proposed for finding the first two min values in LDPC decoders [14], [15][3]: considered cases are $M = 8$ for [14], $M = 6$, $M = 7$ for [15]. For the cases $M = 6$ and $M = 7$ we consider also the unbalanced radix-2 tree proposed in [16] for a generic M, whereas, for the case $M = 7$ we use a IA-MRA: one radix-4 and one radix-3 at $l = 1$ ($K_1 = 4, 3$) and $K_2 = 2$.

For MRAs in Table III we observe that when $M = 32$ the best solution with $N = 2$ (8/4) leads to $\eta_{32}^* > 1$ whereas the best solution for $N = 3$ (4/2/4) achieves $\eta_{32}^* = 0.84$. When $M = 64$, $N = 4$ the best solution (4/2/4/2)

[3]Even if in [14] and [15] min values are found we will refer to max values.

gives $\eta^*_{64} = 0.89$ that is slightly worse than the FRA $K_l = 4$ ($\eta^{(4)}_{64} = 0.88$). Finally, for $M = 9$, $K_l = 3$ we have $\eta^{(3)}_9 = 0.55$; for $M = 24$ the best solution (4/2/3) achieves $\eta^*_{24} = 0.81$.

V. Conclusions

In this brief high speed architectures for finding the first two max/min values are presented. The proposed solution extends previous works based on radix-2 and radix-3 solutions to both higher and mixed radix solutions. As shown by experimental results MRAs achieve lower latency than radix-2 architectures with a limited area increase. Moreover, MRAs show better figures than other solutions proposed for LDPC decoders.

References

[1] M. M. Mansour and N. R. Shanbhang, "High-throughput LDPC decoders," *IEEE Trans. on VLSI*, vol. 11, no. 6, pp. 976–996, Dec 2003.

[2] O. Muller, A. Baghdadi, and M. Jezequel, "Exploring parallel processing levels for convolutional turbo decoding," in *IEEE International Conf. on Information and Communications Tech.: from Theory to Applications*, 2006, pp. 2353–2358.

[3] F. Guilloud, E. Boutillon, and J. L. Danger, "λ-min decoding algorithm of regular and irregular LDPC codes," in *International Symposium on Turbo Codes and Related Topics*, 2003, pp. 451–454.

[4] J. Chen, A. Dholakia, E. Eleftheriou, M. Fossorier, and X. Y. Hu, "Reduced-complexity decoding of LDPC codes," *IEEE Trans. on Communications*, vol. 53, no. 8, pp. 1288–1299, Aug 2005.

[5] D. Oh and K. K. Parhi, "Min-sum decoder architectures with reduced word length for LDPC codes," *IEEE Trans. on Circuits and Systems I*, vol. 57, no. 1, pp. 105–115, Jan 2010.

[6] A. Cevrero, Y. Leblebici, P. Ienne, and A. Burg, "A 5.35 mm^2 10GBASE-T ethernet LDPC decoder chip in 90 nm CMOS," in *IEEE Asian Solid-State Circuits Conf.*, 2010, pp. 1–4.

[7] S. Papaharalabos, M. Sybis, P. Tyczka, and P. T. Mathiopoulos, "Modified Log-MAP algorithm for simplified decoding of turbo and turbo TCM codes," in *IEEE Vehicular Tech. Conf. (Spring)*, 2009, pp. 1–5.

[8] Y. Sun and J. R. Cavallaro, "Low-complexity and high-performance soft MIMO detecion based on distributed M-algorithm through trellis-diagram," in *IEEE International Conf. on Acoustics, Speach and Signal Processing*, 2010, pp. 3398–3401.

[9] D. Patel, V. Smolyakov, M. Shabany, and P. G. Gulak, "VLSI implementation of a WiMAX/LTE compliant low-complexity high-throughput soft-output K-Best MIMO detector," in *IEEE International Symposium on Circuits and Systems*, 2010, pp. 593–596.

[10] P. Tsai, W. Chen, X. Lin, and M. Huang, "A 44 64-QAM reduced-complexity K-Best MIMO detector up to 1.5Gbps," in *IEEE International Symposium on Circuits and Systems*, 2010, pp. 3953–3956.

[11] D. Declercq and M. Fossorier, "Decoding algorithms for nonbinary LDPC codes over GF(q)," *IEEE Trans. on Communications*, vol. 55, no. 4, pp. 633–643, Apr 2007.

[12] R. M. Pyndiah, "Near-optimum decoding of product codes: block turbo codes," *IEEE Trans. on Communications*, vol. 46, no. 8, pp. 1003–1010, Aug 1998.

[13] C. Leroux, C. Jego, P. Adder, M. Jezequel, and D. Gupta, "A highly parallel turbo product code decoder without interleaving resource," in *IEEE Workshop on Signal Processing Systems*, 2008, pp. 1–6.

[14] K. Gunnam, G. Choi, and M. Yeary, "A parallel VLSI architecture for layered decoding for array LDPC codes," in *IEEE International Conf. on VLSI Design*, 2007, pp. 738–743.

[15] X. Y. Shih, C. Z. Zhan, C. H. Lin, and A. Y. Wu, "An 8.29 mm^2 52 mw multi-mode LDPC decoder design for mobile WiMAX system in 0.13 μm CMOS process," *IEEE Journal of Solid-State Circuits*, vol. 43, no. 3, pp. 672–683, Mar 2008.

[16] C. L. Wey, M. D. Shieh, and S. Y. Lin, "Algorithms of finding the first two minimum values and their hardware implementation," *IEEE Trans. on Circuits and Systems I*, vol. 55, no. 11, pp. 3430–3437, Dec 2008.

[17] K. Gunnam, G. Choi, W. Wang, and M. Yeary, "Parallel VLSI architecture for layered decoding," Texas A&M University, Tech. Rep., May 2007, available online at http://dropzone.tamu.edu/TechReports.

[18] Y. Matijasevic, "Enumerable sets are diophantine," *Doklady Akademiky Nauk SSSR*, vol. 11, pp. 279–282, 1970, English translation: Soviet Math Doklady 11, 354-357.

[19] A. Pulimeno, M. Graziano, and G. Piccinini, "UDSM trends comparison: From technology roadmap to UltraSparc Niagara2," *IEEE Trans. on VLSI*, 10.1109/TVLSI.2011.2148183, to appear.

ACKNOWLEDGMENT OF FINANCIAL SUPPORT

This work is partially supported by the NEWCOM++ NoE.

AFFILIATION OF AUTHORS

The authors are with Dipartimento di Elettronica - Politecnico di Torino - Italy.