CHALMERS
UNIVERSITY OF TECHNOLOGY

Unfolded SiBM BCH Decoders for High-Throughput Low-Latency Applications

Citation for the original published paper (version of record):
Wang, X., Fougstedt, C., Svensson, L. et al (2024). Unfolded SiBM BCH Decoders for High-Throughput Low-Latency Applications. 2024 IEEE Computer Society Annual Symposium on VLSI (ISVLSI). http://dx.doi.org/10.1109/ISVLSI61997.2024.00048

(article starts on next page)

Unfolded SiBM BCH Decoders for High-Throughput Low-Latency Applications

Xu Wang*, Christoffer Fougstedt†, Lars Svensson‡, Per Larsson-Edefors‡
*Dept. of Computer Science and Engineering, Chalmers University of Technology, Gothenburg, Sweden
†Ericsson Research, Gothenburg, Sweden
‡Dept. of Microtechnology and Nanoscience, Chalmers University of Technology, Gothenburg, Sweden
xuwang@chalmers.se

***Abstract*—Low-latency and area-efficient forward error correction is crucial in high-throughput communication scenarios, such as die-to-die connections. Using t to denote error correction capability, we propose a low-latency t-unfolded simplified inverse-free Berlekamp-Massey (SiBM) decoder, which for $t > 3$ offers a shorter critical path compared with area-efficient Peterson-based decoders. Synthesized in a 22-nm CMOS process, our unfolded SiBM decoders with $t = 4$ and 5 provide up to 1.39× higher throughput than their Peterson-based counterparts, at comparable area efficiencies.**

***Index Terms*—Application specific integrated circuits, forward error correction, Berlekamp-Massey, unfolded decoding.**

I. Introduction

Forward error correction (FEC) is commonly utilized in communication systems [1] to push the data rate higher and enhance transmission reliability, and in memory systems [2] to improve storage integrity. Nowadays, there is also a growing interest in integrating FEC into die-to-die (D2D) connections, aiming at mitigating insertion losses and distortions caused by crosstalk [3], [4]. In order to integrate FEC functionality in a D2D connection, the FEC circuit must 1) be resource efficient due to the limited area and power budget, 2) provide high data throughput, and 3) offer low transmission latency. To address these issues, hard-decision (HD) codes may provide a possible solution because they are less complex than soft-decision codes [5]. Among HD codes, the Bose-Chaudhuri-Hocquenghem (BCH) codes can provide multiple error correction capabilities while maintaining lower hardware complexity compared to Reed-Solomon codes [6]. Additionally, binary BCH codes permit tradeoffs involving iteration time and calculation complexity [2].

In binary BCH decoders, the Berlekamp-Massey (BM) [7], [8] and Peterson [9] algorithms are commonly employed. When prioritizing latency, the Peterson algorithm is preferred [1], [10], [11], especially for lower error correction capabilities t, because it directly solves the Newton identities without the recurrent calculation associated with BM algorithms. While it has been shown in [11] that Peterson-based decoders can provide efficient implementations for $t \leq 2$, it is however unclear whether this is still the best approach for higher error correction capabilities, such as $t \geq 3$.

In this paper, we propose a t-unfolded decoder structure based on the simplified inverse-free BM (SiBM) algorithm and show that it is possible to reduce the computational delay of the key equation solver to only one clock cycle by unfolding the iterations. Therefore, the proposed circuit can provide high throughputs; up to 3.71 Tb/s with $t = 4$ in a 22-nm CMOS process. Unfolding exposes additional avenues of optimization to reduce the hardware complexity, such as reducing the number of required adders and multipliers. To investigate efficient FEC decoder implementations for application scenarios with $t \geq 3$, we contrast our implementation with decoders based on the Peterson algorithm: Our proposed t-unfolded architecture scales better with t than the Peterson algorithm does and it can provide up to 1.39× higher throughput, while maintaining competitive area efficiency.

II. Berlekamp-Massey Algorithms

Thanks to different techniques to optimize the primary BM algorithm [2], [6], [12]–[14], BM-based decoders implementations can approach the resource efficiency of implementations based on the Peterson algorithm. These enhancing techniques focus on reducing timing and hardware complexity. Inversionless BM (iBM) replaces the hardware-consuming inversions by multiplications [12], decreasing the critical path to two Galois field (GF) multipliers and $1 + \lceil \log_2(t+1) \rceil$ adders. Based on this technique, a reformulated look-ahead inversionless BM (riBM) was proposed [12], [13] to further reduce the critical path. This improvement is achieved by simultaneously calculating the discrepancy and error location polynomial, resulting in a critical path with only $T_{mult} + T_{add}$ (see Table I), where T_{mult} and T_{add} denote the delays associated with a GF multiplier and a GF adder, respectively.

TABLE I. Timing Complexity of BM Algorithms

KES	Logic depth	Delay
iBM [12]	$2 \cdot T_{mult} + (1 + \lceil \log_2(t+1) \rceil) \cdot T_{add}$	$2t$
riBM [12]	$T_{mult} + T_{add}$	$2t$
SiBM [2]	$T_{mult} + T_{add}$	t

However, the improved timing performance comes at the cost of integrating more computational units, among them $t-1$ GF multipliers, which in turn increases the area overhead (see Table II). It should be noted that the BM algorithms discussed above are primarily proposed for general BCH codes. For binary BCH codes, however, the computational delay can be reduced to t [6]. In addition, the calculations

of error evaluation polynomials are redundant for binary BCH code since the error value is constant one. By utilizing these properties, the simplified inverse-free BM (SiBM) [2], [14] has lower hardware overhead and shorter computational time compared to its predecessors.

TABLE II. HARDWARE COMPLEXITY OF BM ALGORITHMS

KES	Adders	Mults	Regs	Muxes
iBM [12]	$3t+1$	$5t+3$	$6t+2$	$2t+1$
riBM [12]	$3t+1$	$6t+2$	$6t+2$	$3t+1$
SiBM [2]	$2t$	$4t$	$4t$	$4t$

III. IMPLEMENTATION OF BINARY BCH DECODER

Binary BCH codes are cyclic error-correcting codes constructed over a Galois field GF(2^m) [6], where m is the primitive power. Binary BCH codes can be denoted by BCH(n,k,t), where n is block length, k is useful message length, and t is the maximal number of bits that can be corrected, with $n-k=m\cdot t$ and $n=2^m-1$. A binary BCH FEC system comprises an encoder and a decoder. The encoder computes and appends the parity check bits on the original message. This is achieved either through a linear feedback shift register (LFSR) or simple exclusive OR (XOR) trees derived from the generator matrix.

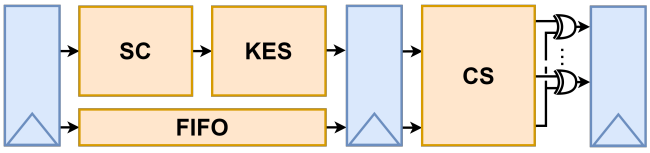

Figure 1. A typical structure of the algebraic-based decoder.

Compared to the encoder, the decoder is more complex. Figure 1 shows a typical structure of the algebraic decoder that contains three main components: syndrome calculation (SC), key equation solving (KES), and Chien search (CS). The syndrome vector $\mathbf{S}$ is calculated by multiplying the received codeword with a transposed parity check matrix $\mathbf{H}$ [6]. If all the elements of vector $\mathbf{S}$ are zero, no errors are detected in the received code; otherwise, errors are detected. The error location polynomial $\Lambda(x)$ can then be determined by solving the Newton identities [6]. Once the $\Lambda(x)$ is calculated, the error location can be identified using the Chien search algorithm [15]. Leveraging the cyclic property of binary BCH codes, the Chien search evaluates the roots individually to determine whether they are the roots of $\Lambda(x)$. To speed up the circuit, a parallel Chien search is implemented in the below architectures by unfolding the iterations and examining all the roots concurrently. Consequently, the error can be corrected by XOR operations between the received message and error patterns [6]. Registers are inserted between the blocks, effectively pipelining the circuit and reducing the logic depth. To synchronize the input signals of CS, a First-In-First-Out (FIFO) buffer is integrated. The buffer should mirror the latency inherent to KES since there is no register inserted in SC. Clock gating logic is implemented to disable the KES and CS components when an error-free message is received, improving hardware efficiency.

Due to the high degree of parallelism in the SC and CS components, the main latency contributor of the decoder is the KES unit. The Peterson algorithm provides a possible solution to reduce the calculation overhead involved in KES because of its direct solving property, reducing the latency to only one cycle. Although the SiBM implementations in [2], [14] decrease computational delay to half that of the riBM, the KES unit still requires t cycles to complete the iterative calculation of $\Lambda(x)$. This recurrent calculation reduces the throughput of the whole system. It is possible to incorporate more KES units to concurrently calculate $\Lambda(x)$ to reduce the latency. Clearly, selecting suitable KES algorithms and architectures becomes imperative to realizing a high-throughput and low-latency decoder system.

IV. TRADEOFF BETWEEN AREA USAGE AND THROUGHPUT

This section discusses two basic decoder architectures, demonstrating a tradeoff between area usage and throughput. The conventional way to build the KES unit involves utilizing a single core to iteratively calculate the coefficients of $\Lambda(x)$.

A. One-core SiBM decoder

The one-core SiBM architecture utilizes a single SiBM-based KES core to iteratively compute the error location polynomial $\Lambda(x)$. One multiplexer (MUX) is added before the input registers to select between the input syndrome and the previous output of the SiBM core (see Figure 2). Therefore, the KES critical path becomes $T_{mux}+T_{mult}+T_{add}$, which is one MUX delay[1] longer than the original SiBM [2]. A controller is also integrated to control the input MUX. Since only one core is used, the resulting design should be highly area efficient (see Table III). One computation of $\Lambda(x)$, however, takes t clock cycles, resulting in high latency and thus low throughput. To be more specific, the SC unit has to hold for t clock cycles to prepare the next syndrome vectors, and the CS unit has to wait for t clock cycles to update the error locations. To synchronize the received signals, a t-stage FIFO should be used which increases area usage.

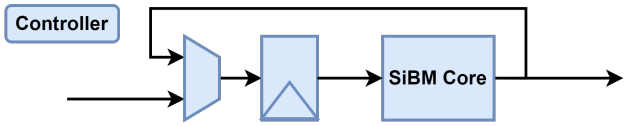

Figure 2. Structure of the one-core SiBM KES.

B. t-parallel SiBM decoder

The throughput of the overall decoder can be increased by parallelizing t SiBM cores, where each core handles one $\Lambda(x)$ computation. A three-core example is shown in Figure 3. Similar to the one-core structure, each core recursively computes one $\Lambda(x)$. While each core is still engaged for t

[1] T_{mux}, T_{mult}, and T_{add} represent the delays of a 2-input MUX, a GF multiplier, and a GF adder, respectively.

TABLE III. TIMING AND HARDWARE COMPLEXITY OF DIFFERENT KES IMPLEMENTATIONS

KES		Timing complexity		Hardware complexity							
		Logic depth	Delay	Adders		Mults		Regs	Muxes		
One-core SiBM		$T_{mult}+T_{add}+T_{mux}$	t	$2t$		$4t$		$4t$	$8t$		
t-parallel SiBM		$T_{mult}+T_{add}+\lceil\log_2 t\rceil\cdot T_{mux}$	$t+1$	$2t^2$		$4t^2$		$4t^2$	$8t^2$		
t-unfolded SiBM	$t=3$	$5\cdot T_{mult}+3\cdot T_{add}$	1	$2t^2-4t+1$	7	$4t^2-5t+1$	22	t	$2t^2-4t+2$	8	
	$t=4$	$6\cdot T_{mult}+4\cdot T_{add}$			17		45			18	
	$t=5$	$8\cdot T_{mult}+5\cdot T_{add}$			31		76			32	
Peterson	$t=3$	$3\cdot T_{mult}+2\cdot T_{add}$	1	3		6		t	2		
	$t=4$	$7\cdot T_{mult}+4\cdot T_{add}$		15		20			3		
	$t=5$	$11\cdot T_{mult}+6\cdot T_{add}$		27		40			6		

clock cycles, overhead can be concealed as there is always at least one SiBM core available to update $\Lambda(x)$ based on the new syndrome. Input demultiplexer (DEMUX) and output MUX are incorporated to facilitate the delivery of the syndrome and $\Lambda(x)$. Because of the output MUX, the critical path turns into $T_{mult}+T_{add}+\lceil\log_2 t\rceil\cdot T_{mux}$. The controller contains information about the current status, such as which core is available, and regulates the DEMUX and MUXes. Therefore, the SC and CS units do not need to stall, leading to a t-times higher throughput than the one-core topology. The FIFO, however, needs $t+1$ stages for synchronization.

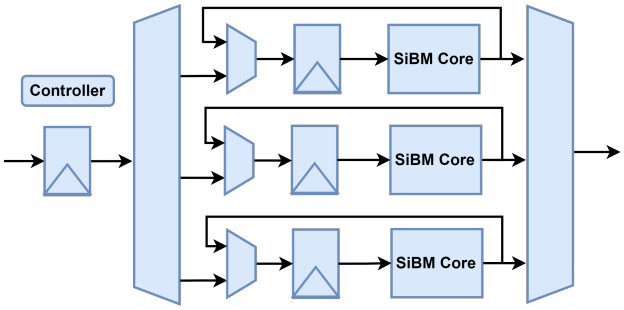

Figure 3. Structure of the t-parallel SiBM KES.

Although the parallel architecture can increase the throughput by t times, the resulting area of the KES unit is t times larger than the one-core KES architecture (see Table III), only considering the SiBM cores and their input registers and MUXes. One-core and t-parallel architectures present a trade-off between area usage and throughput, which is illustrated in the synthesis results in Figure 4. Neither of the architectures can provide a solution that balances area efficiency with high throughput. Hence, a more efficient architecture is desired.

V. HIGH-THROUGHPUT LOW-LATENCY DECODERS

To maintain high throughput and reduce the hardware complexity of t-parallel SiBM architectures, we propose a t-unfolded architecture that utilizes t optimized SiBM cores. In order to compare with the direct Peterson solution, we also present a Peterson-based decoder for $t=5$.

A. t-unfolded SiBM decoder

Replicating parallel SiBM cores leads to excessive hardware usage. Here, we instead unfold the SiBM algorithm, which allows us to simplify the control logic while also reducing hardware overhead as no muxing is required. This is possible

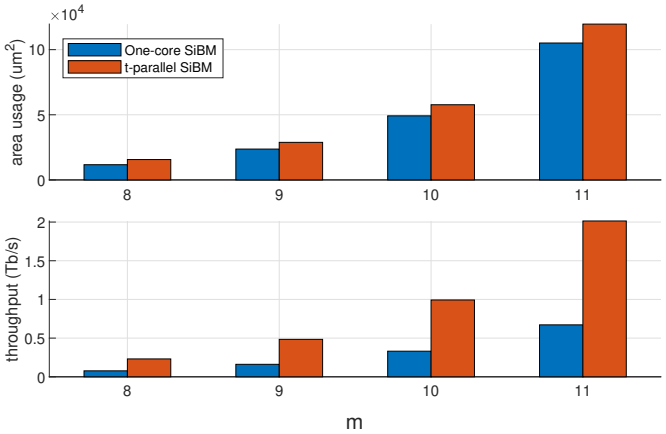

Figure 4. Comparison of the $t=3$ one-core and t-parallel SiBM-based decoders, synthesized at 1-GHz frequency using a 22-nm design kit. It should be noted that the area usage here is the full decoder, not only the KES.

as the main iteration adheres to a predictable pattern without complex interdependencies. In contrast to the parallel structure, t SiBM-cores are directly cascaded; see Figure 5 for a $t=3$ example. The unfolded SiBM decoder offers low latency because the computation of $\Lambda(x)$ can be finished in only one clock cycle. The critical path increases to $t\cdot(T_{mult}+T_{add})$, but the input MUX can be removed. Based on the principles of GF arithmetic, the even syndromes can be calculated from the odd syndromes. Therefore, to improve the hardware efficiency, the even-syndrome calculations are moved from the SC unit to the first SiBM core. This relocation means that the calculation of even syndromes is activated only when errors are detected. Consequently, the final critical path grows by two or three GF multipliers depending on the error correction capability, as shown in Table III.

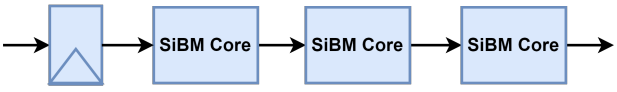

Figure 5. Structure of the 3-unfolded SiBM KES.

The unfolded architecture has a lower hardware complexity than the parallel counterpart since some input operands and output results of each core are deterministic. The number of input registers is reduced to t and the required MUXes for each core decreases to $2t$. For the first core, as $\gamma(0)=1$, the main computation [14]

$$\Delta_i(r+1)=\gamma(r)\Delta_{(i+2)}(r)+\Delta_0(r)\Theta_{(i+1)}(r), \qquad (1)$$

where $r \in \{0, 1, \ldots, t-1\}$, can be simplified to

$$\Delta_i(r+1) = \Delta_{(i+2)}(r) + \Delta_0(r)\Theta_{(i+1)}(r). \quad (2)$$

Therefore, $2t$ GF multipliers can be replaced by direct mappings of the input syndrome. In addition, three adders, two multipliers, and two MUXes can be eliminated because of the constant zero and one inputs; see Table IV, for each r, the above row is $\Delta_i^1(r+1) = \gamma(r)\Delta_{(i+2)}(r)$ and the below is $\Delta_i^2(r+1) = \Delta_0(r)\Theta_{(i+1)}(r)$. However, $t-1$ multipliers are added for even syndrome calculations. The last core can also be optimized as the computation of $\Theta(t)$ is redundant. Hence, the last iteration only contains the calculation of $\Delta(t)$:

$$\Delta_i(t) = \gamma(t-1)\Delta_{(i+2)}(t-1) + \Delta_0(t-1)\Theta_{(i+1)}(t-1). \quad (3)$$

Besides the elimination of $2t$ assignment MUXes, the last core can still be improved due to two constant zero coefficients of $\Theta_{(i+1)}(t-1)$ and $\Delta_{(i+2)}(t-1)$ (see Table IV). Similar to the last core and first core, the inner cores can be optimized for constant zero operations. Therefore, $4 \cdot (t-2)$ multipliers and adders, and $2 \cdot (t-2)$ MUXes can be omitted in inner cores.

TABLE IV. 3-UNFOLDED $\Delta_i(r+1)$ CALCULATIONS

$r=0$	S_3	S_4	S_5	0	1	0
	$\Delta_0^2(1)$	$\Delta_1^2(1)$	$\Delta_2^2(1)$	$\Delta_3^2(1)$	0	S_1
$r=1$	$\Delta_0^1(2)$	$\Delta_1^1(2)$	$\Delta_2^1(2)$	$\Delta_3^1(2)$	0	0
	$\Delta_0^2(2)$	$\Delta_1^2(2)$	0	0	$\Delta_4^2(2)$	$\Delta_5^2(2)$
$r=2$	$\Delta_0^1(3)$	$\Delta_1^1(3)$	$\Delta_2^1(3)$	$\Delta_3^1(3)$	0	0
	0	0	$\Delta_2^2(3)$	$\Delta_3^2(3)$	$\Delta_4^2(3)$	$\Delta_5^2(3)$

As all cores are directly cascaded without any pipeline registers, the overhead introduced by KES is only one clock cycle and a one-stage FIFO is sufficient, reducing the hardware complexity further. Consequently, the t-unfolded implementation can provide efficient hardware compared to the t-parallel SiBM and achieve high throughput with low latency.

Figure 6. Structure of Peterson-based KES.

B. Peterson-based decoder

The Peterson-based [9] decoders provide a low-latency solution as the coefficients of $\Lambda(x)$ can be directly calculated by reformulating the Newton identities. A structure of the Peterson-based KES is shown in Figure 6. Inversionless Peterson equations with $t = 1-4$ which eliminate hardware-costly GF inversions are described in [1], [11]. Based on these, we develop Peterson equations for $t = 5$ to explore the scenario of higher error correction capabilities.

To remove the inversions, the scaling part (SP) should be first calculated

$$\begin{aligned}
\mathrm{SP}_1 &= (S_1^3 + S_3)[(S_1 S_3^2 + S_7) + S_1^4(S_1^3 + S_3)] \\
&\quad + (S_5 + S_1^5)(S_1^2 S_3 + S_5) \\
\mathrm{SP}_2 &= (S_1^4 S_3^2 + S_5^2) + (S_1^3 + S_3)[S_1^2(S_5 + S_1^5) \\
&\quad + (S_1 S_3^2 + S_7)],
\end{aligned}$$

where S_i is the syndrome. Then, the coefficients of the error location polynomial Λ_i can be derived by

$$\begin{aligned}
\Lambda_0 &= \mathrm{SP}_1\mathrm{SP}_2 \\
\Lambda_1 &= S_1\Lambda_0 \\
\Lambda_2 &= \mathrm{SP}_2\{(S_1^3 + S_3)[S_3^2(S_1^3 + S_3) + S_1^4(S_5 + S_1^2 S_3) \\
&\quad + (S_1^9 + S_9)] + (S_5 + S_1^5)[(S_1^2 S_5 + S_3^2 S_1) \\
&\quad + (S_7 + S_1^7)]\} \\
\Lambda_3 &= (S_1^3 + S_3)\Lambda_0 + S_1\Lambda_2 \\
\Lambda_4 &= \mathrm{SP}_1\{(S_1^2 S_3 + S_5)[(S_1^2 S_7 + S_9) + S_3(S_1^6 + S_3^2)] \\
&\quad + [S_1^2(S_5 + S_1^5) + (S_1 S_3^2 + S_7)] \\
&\quad \cdot [(S_1^7 + S_7) + S_1(S_1 S_5 + S_3^2)]\} \\
\Lambda_5 &= (S_5 + S_1^2 S_3)\Lambda_0 + (S_1^3 + S_3)\Lambda_2 + S_1\Lambda_4.
\end{aligned}$$

If three errors have occurred in the codeword, $\mathrm{SP}_1 = \mathrm{SP}_2 = 0$. In this case, all coefficients will become zero and the above equations are insufficient. To address this case, the set of equations should be substituted with [1]

$$\begin{aligned}
\Lambda_0 &= S_1^3 + S_3 & \Lambda_2 &= S_1^2 S_3 + S_5 \\
\Lambda_1 &= S_1\Lambda_0 & \Lambda_3 &= \Lambda_0^2 + S_1\Lambda_2.
\end{aligned}$$

and $\Lambda_0 = 1$, $\Lambda_1 = S_1$ to tackle a single-error case.

Table III shows that the critical path of the Peterson-based decoder is longer than that of the t-unfolded SiBM for $t = 4$ and 5. This means that the t-unfolded SiBM may provide higher speed and higher throughput than the Peterson approach. Regarding area usage, however, the Peterson algorithm may still perform competitively.

VI. RESULTS

All configurations, including both SiBM and Peterson, were synthesized on a 22-nm fully-depleted silicon-on-insulator (FD-SOI) CMOS technology, at $V_{dd} = 0.9$ V, typical corner. To limit the code overhead to below 20%, decoders with $t = 3-5$ and $m = 8-11$ were designed and evaluated. We focus on the area usage as the power dissipation is expected to be similar at the low input bit-error rates that the decoders will operate at. The overall power will be dominated by the SC unit and the FIFO buffer since the KES and CS units are rarely activated.

Figure 7 shows the area usage of different architectures synthesized at 1 GHz. The resulting area usage considers the whole decoder system, including SC, KES, and CS computation components, FIFO, and pipeline registers in between. In general, the area usage rises when m and t increase, because more logic components, such as GF multipliers and adders, are integrated. Our proposed t-unfolded structure performs better than the parallel structure due to the optimization described in Section V: (1) unfolding the iteration helps expose the logic to more optimizations, and (2) a less complex FIFO buffer is needed to synchronize the signals; in fact one stage is sufficient. Even compared with the efficient Peterson-based decoder, the t-unfolded SiBM can offer competitive or better area complexity; see e.g. $t = 5$ in Figure 7.

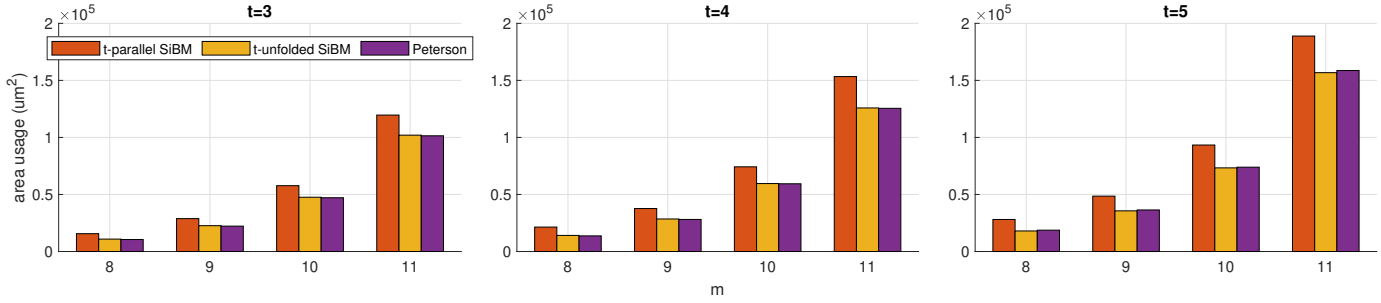

Figure 7. Area usage of t-parallel vs t-unfolded SiBM-based vs Peterson-based decoders.

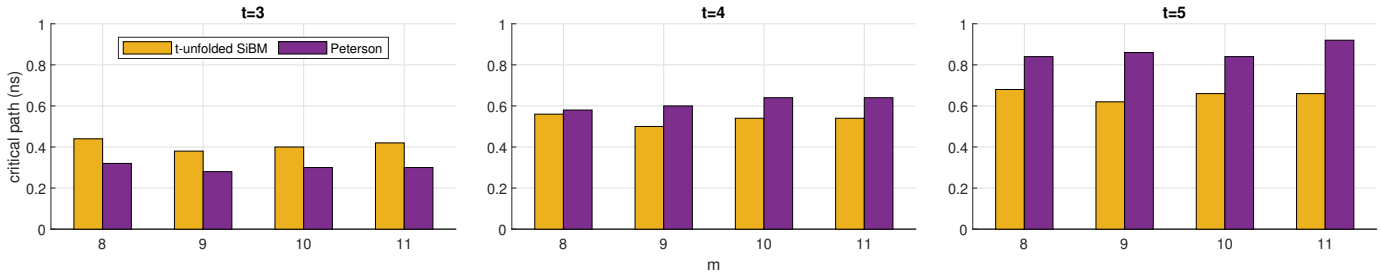

Figure 8. Critical path of the t-unfolded SiBM-based and Peterson-based decoders.

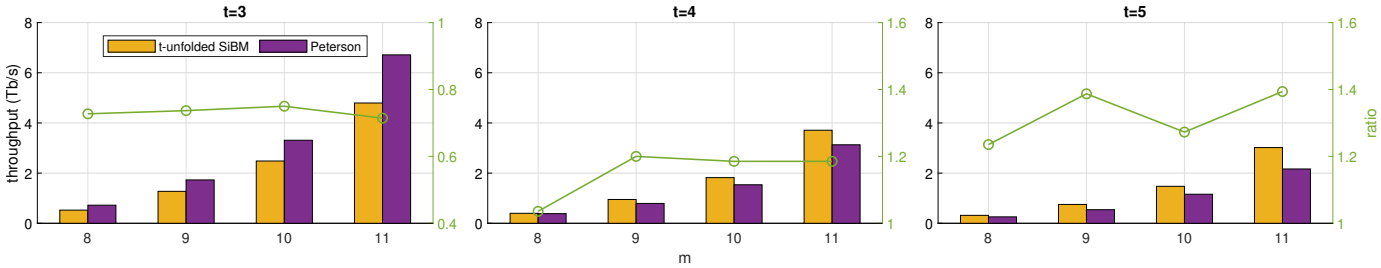

Figure 9. Throughput of the implemented decoder configurations. The left-hand y-axis shows the absolute throughput of the t-unfolded SiBM-based and Peterson-based decoders. The right-hand y-axis shows the speedup ratio of the t-unfolded SiBM-based decoders over Peterson-based decoders.

In terms of timing, t-unfolded SiBMs with $t = 4$ and 5 decoders exhibit a short critical path compared to their Peterson-based counterparts (see Figure 8). This is because the coefficient calculation of the Peterson algorithm heavily relies on S_1, especially for higher ts. For a Peterson-based KES with $t = 5$, the highest power of S_1 is nine, resulting in a long critical path, $11 \cdot T_{mult} + 6 \cdot T_{add}$, and high fanout. The t-unfolded KES, however, only has a $8 \cdot T_{mult} + 5 \cdot T_{add}$ logic depth (see Table III), $3 \cdot T_{mult} + T_{add}$ less than the Peterson counterpart. Nevertheless, for $t = 3$, the Peterson-based decoder provides a better timing performance due to the shorter critical path, $3 \cdot T_{mult} + 2 \cdot T_{add}$, in contrast to $5 \cdot T_{mult} + 3 \cdot T_{add}$ of the t-unfolded SiBM.

Better timing performance also means that higher throughput is possible. The throughput is calculated by multiplying the useful message length, k, with the maximal clock rate from synthesis. Therefore, for a specific message length, the highest throughput of the decoder is determined by its maximal synthesized clock rate. Figure 9 shows that the unfolded SiBM architectures can offer higher throughput compared to the Peterson-based ones for $t = 4$ and 5. For an $m = 11$ and $t = 5$ configuration, the unfolded SiBM can provide 3.02-Tb/s throughput, which is 1.39× times higher than the Peterson-based (2.17 Tb/s) design. In contrast, the $t = 3$ unfolded SiBM decoders have lower throughput than their Peterson counterparts because the longer critical path leads to a lower maximal clock rate.

VII. Conclusion

To resolve the dilemma posed by conventional simplified inverse-free Berlekamp-Massey (SiBM) decoders, which struggle to reconcile low latency with efficient area usage demanded by e.g. multiple-lane high-throughput communication links for chiplet interfaces, we propose the t-unfolded SiBM decoder. This decoder offers higher throughput for $t = 4$ and 5 and competitive area usage compared with the corresponding Peterson-based decoders. Synthesized in a 22-nm CMOS process, an unfolded SiBM decoder for BCH(2047,1992,5) achieves a data rate of 3.02-Tb/s, which is 1.39× higher than its Peterson counterpart.

ACKNOWLEDGEMENT

The authors would like to thank GlobalFoundries for design kit access.

REFERENCES

[1] C. Fougstedt and P. Larsson-Edefors, "Energy-Efficient High-Throughput VLSI Architectures for Product-Like Codes," *Journal of Lightwave Technology*, vol. 37, pp. 477–485, 2019.

[2] W. Liu, J. Rho, and W. Sung, "Low-Power High-Throughput BCH Error Correction VLSI Design for Multi-Level Cell NAND Flash Memories," *IEEE Workshop on Signal Processing Systems Design and Implementation*, pp. 303–308, 2006.

[3] "Universal Chiplet Interconnect Express (UCIe) Specification Rev 1.0." https://www.uciexpress.org. Online.

[4] D. Das Sharma, G. Pasdast, Z. Qian, and K. Aygun, "Universal Chiplet Interconnect Express (UCIe): An Open Industry Standard for Innovations With Chiplets at Package Level," *IEEE Transactions on Components, Packaging and Manufacturing Technology*, vol. 12, no. 9, pp. 1423–1431, 2022.

[5] P. Larsson-Edefors, C. Fougstedt, and K. Cushon, "Implementation Challenges for Energy-Efficient Error Correction in Optical Communication Systems," in *Advanced Photonics, Signal Processing in Photonic Communications (SPPCom)*, p. SpTh4F.2, 2018.

[6] W. Ryan and S. Lin, *Channel Codes: Classical and Modern*. Cambridge University Press, 2009.

[7] E. Berlekamp, "Nonbinary BCH Decoding," *IEEE Transactions on Information Theory*, vol. 14, no. 2, pp. 242–242, 1968.

[8] J. Massey, "Shift-Register Synthesis and BCH Decoding," *IEEE Transactions on Information Theory*, vol. 15, no. 1, pp. 122–127, 1969.

[9] W. Peterson, "Encoding and Error-Correction Procedures for the Bose-Chaudhuri Codes," *IRE Transactions on Information Theory*, vol. 6, no. 4, pp. 459–470, 1960.

[10] S. An, H. Tang, and J. Park, "A Inversion-Less Peterson Algorithm Based Shared KES Architecture for Concatenated BCH Decoder," *International SoC Design Conference (ISOCC)*, pp. 281–282, 2015.

[11] C. Fougstedt, K. Szczerba, and P. Larsson-Edefors, "Low-Power Low-Latency BCH Decoders for Energy-Efficient Optical Interconnects," *Journal of Lightwave Technology*, vol. 35, no. 23, pp. 5201–5207, 2017.

[12] D. Sarwate and N. Shanbhag, "High-Speed Architectures for Reed-Solomon Decoders," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 9, no. 5, pp. 641–655, 2001.

[13] X. Zhang, *VLSI Architectures for Modern Error-Correcting Codes*. CRC Press, 2015.

[14] M. Yin, M. Xie, and B. Yi, "Optimized Algorithms for Binary BCH Codes," in *IEEE International Symposium on Circuits and Systems (ISCAS)*, pp. 1552–1555, 2013.

[15] R. Chien, "Cyclic Decoding Procedures for Bose-Chaudhuri-Hocquenghem Codes," *IEEE Transactions on Information Theory*, vol. 10, no. 4, pp. 357–363, 1964.