Turkish Journal of Electrical Engineering & Computer Sciences

http://journals.tubitak.gov.tr/elektrik/

Research Article

Turk J Elec Eng & Comp Sci
(2022) 30: 713 – 729
© TÜBİTAK
doi:10.3906/elk-2105-175

Improving utilization rate of semi-parallel successive cancellation architecture for polar codes using 2-bit decoding

Dinesh Kumar DEVADOSS*, **Shantha Selvakumari RAMAPACKIAM**
Department of ECE, Mepco Schlenk Engineering College,Sivakasi, India

Received: 18.05.2021 • **Accepted/Published Online:** 06.12.2021 • **Final Version:** 21.03.2022

Abstract: Polar codes are the capacity-achieving error-correcting code proved to be a significant invention in coding theory. It can achieve channel capacity at infinite code length N due to its explicit code construction. However, the processing complexity along with the higher latency due to successive cancellation (SC) decoding is being a major design issue, which reduces the utilization rate in the decoder architectures. This paper presents a modified semi-parallel architecture for decoding polar code with a better decoding latency. *Precomputation* and *look-ahead* techniques are used to generate two bits in the final stage. Pipelined partial-sum unit with a less critical path reduces hardware complexity independent of code length. Hence, the fact that the proposed architecture reduces the latency by 2.7 times leads to increase in utilization rate than prior semi-parallel architecture. For a code length of $N = 2^{10}$, the proposed architecture shows 62.7% and 94% improved utilization rate compared to the conventional semi-parallel architecture and 2-bit SC decoder, respectively. Compared to the conventional semi-parallel decoder for $N = 2^{17}$, hardware resource such as look-up-tables (LUT) and flip-flops (FF) usage are reduced by 98% in field programmable gate array (FPGA) leads to reduction in processing complexity. Hence, very large efficient polar decoders with a high utilization rate can be implemented in FPGA.

Key words: Polar code, successive cancellation, semi-parallel, hardware architecture, field programmable gate array (FPGA)

1. Introduction

Polar code is one of the forward error-correcting codes, which has been adopted as a control channel for 5th generation (5G) new radio by the 3rd generation partnership project [1]. It is the only capacity-achieving code next to low-density parity-check (LDPC) codes [2] for a binary-input discrete memoryless channel (B-DMC). The channel capacity is achieved by the SC algorithm by Arikan [3] in his seminal work. Some papers address the theoretical aspects of polar codes [4–6]. A family of hardware architectures such as pipelined tree architecture and line architecture is proposed in [7], for SC decoding of polar codes to reduce the hardware complexity with a latency of *2N-2* clock cycles. The paper [8] implements polar decoder by an iterative decoding algorithm called belief propagation decoding which can support multiple code rates. In [9], log-likelihood ratio (LLR) based SC list (SCL) decoding architecture proposed to reduce the hardware complexity by half than conventional SCL, but with a latency of *3N-2* clock cycles. For high data rate applications, latency needs to be reduced further for achieving high throughput. Therefore, hardware architecture has to be revisited for improving the latency while

*Correspondence: dineshddk@mepcoeng.ac.in retaining the processing complexity. In [10], a semi-parallel architecture was proposed with drastically reducing processing complexity and a higher utilization rate than the previous architectures. However, it reports less throughput in comparison with the prior work due to a slight increase in latency. In the semi-parallel decoder, the partial-sum computation module is the limiting factor with a longer critical path. The complexity of this update logic increases with code length N, which leads to an increase in latency. A shift register-based partial-sum computation unit was reported in [11], in which registers are inserted in between the logic gates. It reduces the critical path delay and maximizes the operating frequency than the conventional partial-sum unit [10]. There are hardware architectures in [12, 13], which use *precomputation* and *look-ahead* decoding schedule to enhance the latency and throughput at the expense of higher gate count. It leads to an increase in processing complexity for the practical use of polar decoders at higher block lengths. In addition, the hardware utilization needs improvement for pipelined tree and line architecture in [12, 13]. A dedicated processor is reported in [14] to improve fast-simplified successive cancellation (F-SSC) implementation for decoding constituent codes with a significant rise in throughput, at the cost of augmenting LUT and FF. A high throughput energy-efficient combinational decoder in [15], reports maximum throughput of *2.5* Gbps for shorter block length and *1.24* Gbps for code length of 2^{10} with low power consumption. However, the hardware resources such as LUT, FF, and random access memory (RAM) usage is very high compared to the other conventional decoders, which leads to usage of this decoder only for limited applications. In [16], a radix-4 SC decoder uses partial-sum *look-ahead*, special subcode decoding to reduce latency and drastically enhance the throughput. A low complex merged processing element was implemented in [17] in which application-specific integrated circuits (ASIC) reported notable improvement in throughput and latency. Most of the above-mentioned work focuses on improving the latency and throughput performance of the decoder. However, the utilization rate is not reported in most of the work except [10], which plays a major role in implementing a hardware-efficient decoder.

In this work, a modified semi-parallel architecture is presented with *precomputation* and *look-ahead* techniques to reduce the latency. A pipelined partial-sum unit (PPU) with reduced critical path delay is incorporated to reduce the latency further, thereby enhancing the utilization rate and throughput. The analytical expressions for the architecture based on the scheduling principles are derived, which will validate the implementation results of the proposed decoder. The hardware analysis of the developed decoder is compared with other prior latency reduced architecture. The logic synthesis results of the implemented architecture for the polar code of length up to $N = 2^{17}$ with code rate R are reported using Virtex-6 FPGA.

This paper is organized as follows. Section 2 brief the background of polar code encoding and decoding. Section 3 provides the scheduling principles of the proposed architecture. Section 4 explains the proposed semi-parallel architecture and its operation. Hardware comparison with prior architecture along with implementation results discussed in Section 5. Then finally concluded in Section 6.

2. Background

This section provides preliminary concept of polar code generation from the information vector and generator matrix followed by a brief about successive cancellation decoding of polar code.

2.1. Polar code generation and encoding

Polar codes are operated using the concept of channel polarization for any given B-DMC, where the polar code decomposes the communication channel into several sub-channels whose probability of reliable transmission becomes *1.0*(noiseless) or *0.5*(noise) as N grows towards infinity. Hence, the sender and receiver can use a reliable channel for lower error rate communication. It can be fixed by the setting of sub-channels used to carry K information bits as A and the remaining *(N-K)* bits as the frozen set A_C. In [3], a bit-reversed indexing scheme was used while generating a codeword from the information vector. In this work, normal indexing scheme is followed in the encoding as shown in Figure 1. Let $\boldsymbol{u} = (u_0, u_1, u_2, u_3, ., u_{N-1})$ be the information vector and $\boldsymbol{X} = (X_0, X_1, X_2, X_3, , X_{N-1})$ be the codeword and G be the generator matrix of the polar code.

Generally, polar code '$\boldsymbol{X}$' can be generated from the information vector '$\boldsymbol{u}$' using the generator matrix G,

$$\boldsymbol{X} = \boldsymbol{uG} \tag{1}$$

Where the generator matrix can be derived using the n^{th} kronecker power of base matrix $F = \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix}$.

For example, (2) represents the generator matrix G for $n = 3$ and its equivalent graph representation is shown in Figure 1

$$G = F^{\otimes 3} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} \tag{2}$$

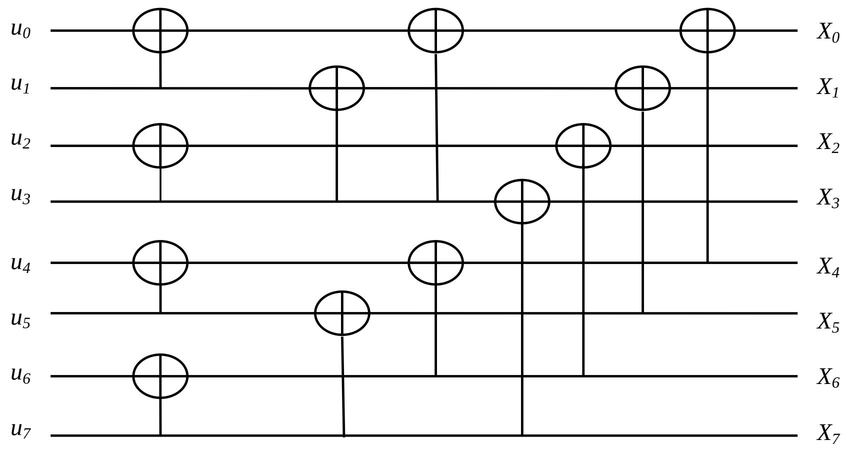

Figure 1. Encoder graph of polar code for $N = 8$.

2.2. Successive cancellation decoding

SC algorithm is used to decode the information vector from the LLRs received from the channel $L_{n,i}$. Here, the codeword after encoding, binary phase-shift keying (BPSK) modulation with additive white gaussian noise (AWGN) channel are used to generate LLR values. The estimated bit U_i is decoded successively provided all the LLR values and the previously decoded bits $U_{i-1} to U_0$ along with frozen bit information of the codeword.

Let $N = 2^n$ be the polar code length has n stages in the decoding graph shown in Figure 2. Let $L_{l,i}$ be the LLR position at each stage where $0 \leq l < n$ be the stages and i be the bit position ranges between $0 \leq i < N$.

The SC decoder successively estimates U_i according to the frozen set values,

$$U_i = \begin{cases} 0 & \text{if } i \epsilon A_c, \\ 0 & \text{if } L_{0,i} \geq 1, \\ 1 & \text{otherwise} . \end{cases} \tag{3}$$

where i be the currently decoded bit and $L_{0,i}$ be the likelihood values of an estimated bit U_i. In the modified semi-parallel architecture, 2b SC decoding techniques in [13] are employed where the last stage is replaced by p node, which successively decodes two bits at a time and leads to a reduction in the latency. Since notation P is used to indicate the number of implemented processing elements (PE) in the semi-parallel architecture, the p node is replaced by a Decision node (D node) to avoid conflict.

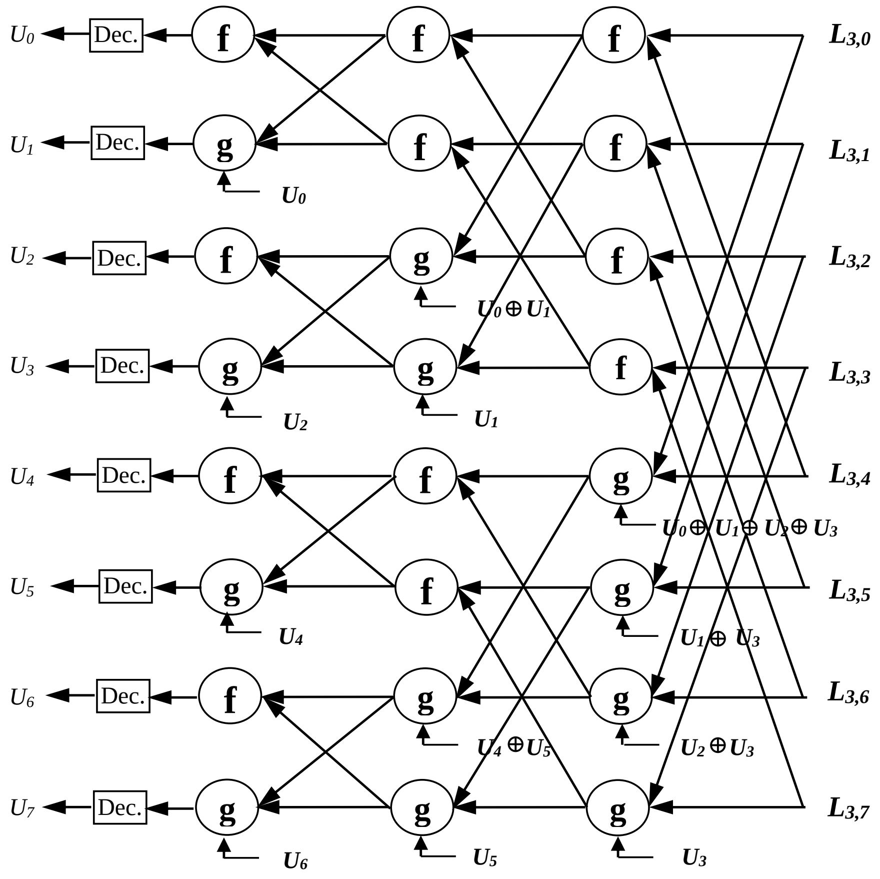

Figure 2. Conventional SC decoding graph of Polar Code for $N = 8$.

3. Scheduling principles of proposed semi-parallel SC decoder

The main flaw that all previous architectures such as tree and line SC decoder [7] faced is their utilization rate α decreases with the increase in code length. The utilization rate is the ratio of the total number of node updates to the product of hardware complexity and latency. Latency of the decoder is the total number of clock cycles required to decode a vector. The utilization rate of α can be improved by lowering the number of PEs. In [10], the implemented PEs are reduced. Hence, the stages require more node updates than the implemented PEs that need multiple clock cycles for the LLR computation, which leads to increase in latency. In the proposed method, the *precomputation* scheme is incorporated in which every stage l of the graph updated 2^{n-l-1} times. The last stage is replaced by D node, which takes frozen bit information from read only memory (ROM) and produces two bits in a single clock. Let P be the number of implemented PEs, where $P = 2^p$. In the stages which satisfies the condition $2^l \leq P$, the number of clock cycles required is reduced by half, and the remaining stages are not affected. Hence, the latency remains the same as that of the previous tree and line architecture. By incorporating the *look-ahead* technique, all the possible values of D node are computed. The corresponding values are selected by using multiplexer (MUX), which leads to the last stage updated 2^{n-l-2} times for $n > 2$ and updated in a single clock cycle. TThe total number of clock cycle *(NC)* can be derived from the decoding schedule as shown in Table 1 and Table 2 for the proposed decoder of *N=8*.

The NC for proposed decoder with *precomputation* is derived as follows:

$$\begin{aligned} NC &= \sum_{l=0}^{p} 2^{n-l-1} + \sum_{l=p+1}^{n-1} 2^{n-l-1}2^{l-p} \\ &= \frac{2^{n-1}(1-\frac{1}{2P})}{1-\frac{1}{2}} + 2^{n-p-1}(n-p-1) \\ &= N(1-\frac{1}{2P}) + \frac{N}{2P}(n-p-1) \\ &= N - \frac{N}{2P}\log_2 2 + \frac{N}{2P}(\log_2 \frac{N}{2P}) \\ &= N + \frac{N}{2P}(\log_2 \frac{N}{4P}) \end{aligned} \tag{4}$$

Similarly, the *NC* for proposed decoder with *precomputation* and *look-ahead* is given by

$$\begin{aligned} NC &= \sum_{l=0} 2^{n-l-2} + \sum_{l=1}^{p} 2^{n-l-1} + \sum_{l=p+1}^{n-1} 2^{n-l-1}2^{l-p} \\ &= \frac{3}{4}N + \frac{N}{2P}\log_2 \frac{N}{4P} \end{aligned} \tag{5}$$

For *precomputation*, utilization rate is given by

$$\alpha = \frac{N\log_2 N}{2P(N + \frac{N}{2P}\log_2 \frac{N}{4P})} = \frac{\log_2 N}{2P + \log_2 \frac{N}{4P}} \tag{6}$$

Similarly, for *precomputation* with *look-ahead* utilization rate is given by

$$\alpha = \frac{N\log_2 N}{2P(0.75N + \frac{N}{2P}\log_2 \frac{N}{4P})} = \frac{\log_2 N}{1.5P + \log_2 \frac{N}{4P}} \tag{7}$$

The proposed semi-parallel decoder achieves lower complexity with the reduction in latency. This approach uses P processing elements. However, a memory requirement increases for *precomputation* of soft bit LLR values.

Table 1. Scheduling of proposed decoder with *precomputation* for $N = 8$ and $P = 2$.

clock	1	2	3	4	5	6	7	8
PE1	$L_{2,0}$	$L_{2,2}$	$L_{1,0}$			$L_{1,4}$		
	$L_{2,4}$	$L_{2,6}$	$L_{1,2}$			$L_{1,6}$		
PE2	$L_{2,1}$	$L_{2,3}$	$L_{1,1}$			$L_{1,5}$		
	$L_{2,5}$	$L_{2,7}$	$L_{1,3}$			$L_{1,7}$		
D node				U_0 U_1	U_2 U_3		U_4 U_5	U_6 U_7

Table 2. Scheduling of proposed decoder with *precomputation* and *look-ahead* for $N = 8$ and $P = 2$.

clock	1	2	3	4	5	6
PE1	$L_{2,0}$	$L_{2,2}$	$L_{1,0}$		$L_{1,4}$	
	$L_{2,4}$	$L_{2,6}$	$L_{1,2}$		$L_{1,6}$	
PE2	$L_{2,1}$	$L_{2,3}$	$L_{1,1}$		$L_{1,5}$	
	$L_{2,5}$	$L_{2,7}$	$L_{1,3}$		$L_{1,7}$	
D node				U_0 U_1 U_2 U_3		U_4 U_5 U_6 U_7

4. Proposed semi-parallel decoder architecture

The proposed architecture comprises of various modules illustrated in Figure 3. The semi-parallel architecture to decode polar code sequentially was proposed in [10], which is focusing mainly on improving the hardware utilization rate. However, the complex partial-sum update logic of the decoder increases the latency with the increasing codeword length N and lesser throughput is observed due to single bit decoding. Complex multiplexing logic leads to complex control modules required to route the signal properly between the RAM module and the PE. Therefore, we propose a modified semi-parallel architecture that can decode two bits in parallel when compared with single bit decoding with reduced latency by using pipelined partial-sum computation inspired by the architecture in [12], where 2-bit decoded output is given as input, and it generates $N/2$ partial-sum as output.

Figure 3 shows the proposed semi-parallel architecture to decode the polar code of length N. In order to utilize the hardware resource, lesser than $N/2$ PEs are used to decode a larger codeword. Initially, parallel-in parallel-out shift registers are used to buffer the LLR received from B-DMC. Then LLRs are loaded via MUX logic into the dual-port RAM of width PQ for further processing by the PEs. The f node and g node of P PEs accept Q bit LLRs each, which equals *2PQ* bits. The P PEs produces three intermediate Q bit LLRs such as minimum, added and subtracted value, which equals *3PQ* bits. This LLRs are loaded into the dual-port RAMs of width PQ via a MUX. During the decoding phase, channel values and f node values stored in one RAM and other two RAMs are used to store added and subtracted value of g node separately. The decoded output U_{2i}, U_{2i+1} from the LLRs produced by the last stage is executed by p^{th} and $(p-1)^{th}$ PEs. Hence, the last two PEs are connected to the decision unit to produce decoded output. Then, the decoded output is fed to the PPU, which provides the necessary partial-sums required for the PEs.

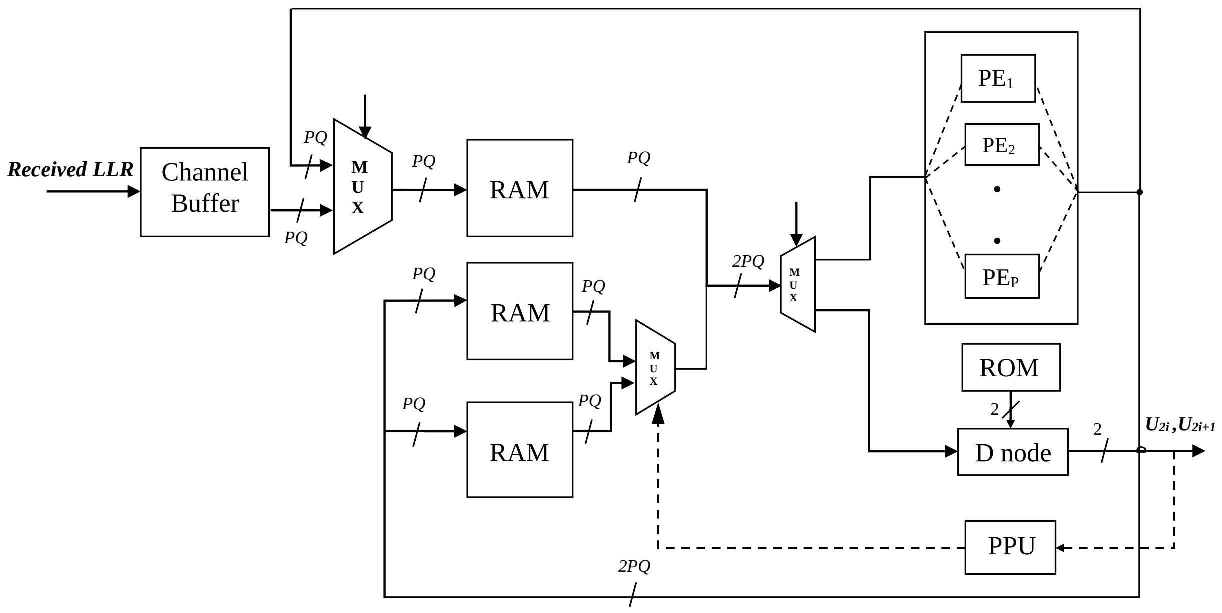

Figure 3. Proposed 2-bit semi-parallel decoder architecture.

4.1. Processing element and D node

Semi-parallel decoder performs their likelihood estimations using LLR [10], whereas previous existing architecture uses likelihood ratios (LR) that involve multiplications and divisions lead to an increase in hardware complexity. The merged PEs are used as shown in Figure 4a, which accept L_a, L_b as inputs and produce outputs L_f, L_g. The output L_f is the minimum value from f node, and L_g is added value to or subtracted value from g node, which is selected by the appropriate partial-sum generated by the PPU. In the LLR domain, functions f and g are performed using the simpler min-sum equations:

$$L_f(L_a, L_b) \approx sign(L_a)sign(L_b)min(\mid L_a \mid, \mid L_b \mid) \tag{8}$$

$$L_g(\hat{s}, L_a, L_b) = L_a(-1)^{\hat{s}} + L_b \tag{9}$$

D node produces outputs U_{2i}, U_{2i+1} based on the frozen conditions indicated by the signals *fr1*,*fr2* respectively. If *fr1* and *fr2* both are *1*, then U_{2i}, U_{2i+1} both are frozen bits. If *fr1* and *fr2* are equal to *0*, this indicates U_{2i}, U_{2i+1} are message bits that can be computed based on the $sign(L_a)$ and $sign(L_b)$. The sign bit produces *0* for non-negative LLR and *1* for negative LLR. The comparator output is denoted as *comp* signal, produces *1* when $|L_a| \geq |L_b|$ otherwise *0*. Based on the above signals, output of the D node is expressed as in [13].

$$U_{2i} = \overline{fr1}\,(sign(L_a) \oplus sign(L_b)) \tag{10}$$

$$U_{2i+1} = \overline{comp}\,\overline{fr2}\,sign(L_b) + comp\,\overline{fr1}\,\overline{fr2}\,sign(L_b) + comp\,fr1\,\overline{fr2}\,sign(L_a) \tag{11}$$

From the Figure 3, Table 1 reports that it takes two D node to produce outputs $U_0 - U_3$. Hence, the critical path of the decoder before *look-ahead* is denoted by $2T_{Dnode} + T_{PPU} + T_{MUX}$. As shown from the

Figure 4b, the critical path of the decoder with *look-ahead* is reduced to $T_{Dnode} + T_{PPU} + T_{MUX}$, which was achieved by connecting fewer extra D node and multiplexers in parallel.

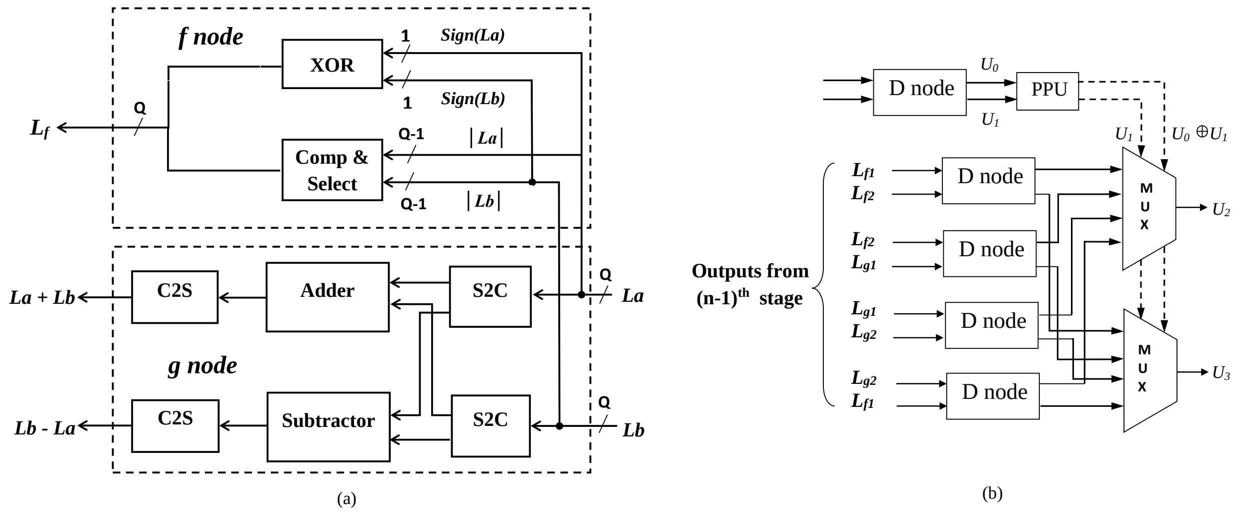

Figure 4. (a) Merged processing element. (b) D node with *look-ahead*.

4.2. LLR memory

The LLR memory stores the channel LLR values and the output of PEs. The internal LLRs produced by the PEs, need to be reused in subsequent steps of the decoding process. The PEs should read inputs and write outputs in a single clock cycle to avoid additional delays, which can be implemented by a dual-port RAM. It can be configured with a write port of width *3PQ* and read port of width *2PQ*. In a single clock cycle, each PE must take two LLRs of bit width *2Q* as input and produce outputs of *3Q* bit width. The channel LLRs are stored as words in the RAM. For a 2-bit semi-parallel decoder of $N = 8$ with $P = 2$, first word contains channel LLR processed by PE_1 and PE_2 produces $L_{2,0}L_{2,4}$ and $L_{2,1}L_{2,5}$ respectively. The second word contains channel LLR processed by the PEs producing $L_{2,2}L_{2,6}$ and $L_{2,3}L_{2,7}$ respectively. LLR memory following this fixed data path alignment scheme has a regular structure $L_{2,0}L_{2,2}L_{2,1}L_{2,3}$ and $L_{2,4}L_{2,6}L_{2,5}L_{2,7}$ are stored in this order follows a bit reversed scheme and LLRs generated are input to the PEs. This approach requires additional overhead bits of $3 \times 642 = 1926Q$ bits of memory for $P = 64$ irrespective of code length N. The memory requirement to store Q-bit LLRs for code length N is given by

$$Total\ memory = (\underbrace{3(N-2)}_{internal\ LLRs} + \underbrace{3*(2PlogP - 2P + 2)}_{additional\ overhead\ bits} + \underbrace{N}_{channel\ LLR}) \times Q \tag{12}$$

4.3. Channel buffer and RAM

We have used quantization bits Q of *5* for better error-correcting performance. For a polar code of higher block length, it is not practical to decode all the channel outputs simultaneously. The received LLRs are buffered in channel registers in a group of *64* LLRs. Clock cycles of *16* are required to load one frame in the channel registers for a code length of $N = 1024$ with $P = 64$. The dual-port RAM is used to write the new frame and decode the current frame simultaneously and the decoder read *2PQ bits*, which requires $2 \times 64 \times 5 = 640bit$ bus. Hence, multiple RAM banks of the same input width used in which data can be written one RAM at a time but read from all simultaneously and the decoder requires one bank to store channel LLRs and internal LLRs for f node with a depth of *42* and a width of $64 \times 5 = 320 bits$ and two banks with a depth of *26* and a width of *320 bits* to store the internal LLRs of g node.

4.4. Pipelined partial-sum computation unit

During the decoding process, a maximum of P numbers of f and g functions can be performed in parallel. PPU used is recursive in nature leads to the PPU for code length 2^n can be constructed by using 2^{n-1} with additional *N/4* FFs, XOR gates, and MUX as shown in Figure 5. The critical path of the PPU is two XOR gates, since one exor is performed for the two bits obtained from D node say $U_2 + U_3$, and this resultant is exor with previous value $U_0 + U_1$ stored in the register to generate a partial-sum of $U_0 + U_1 + U_2 + U_3$. The *N/2* outputs of PPU are mapped to P numbers of PEs by using an extra MUX logic at the PPU output. Select lines have chosen to route the partial-sum generated from PPU to each PE.

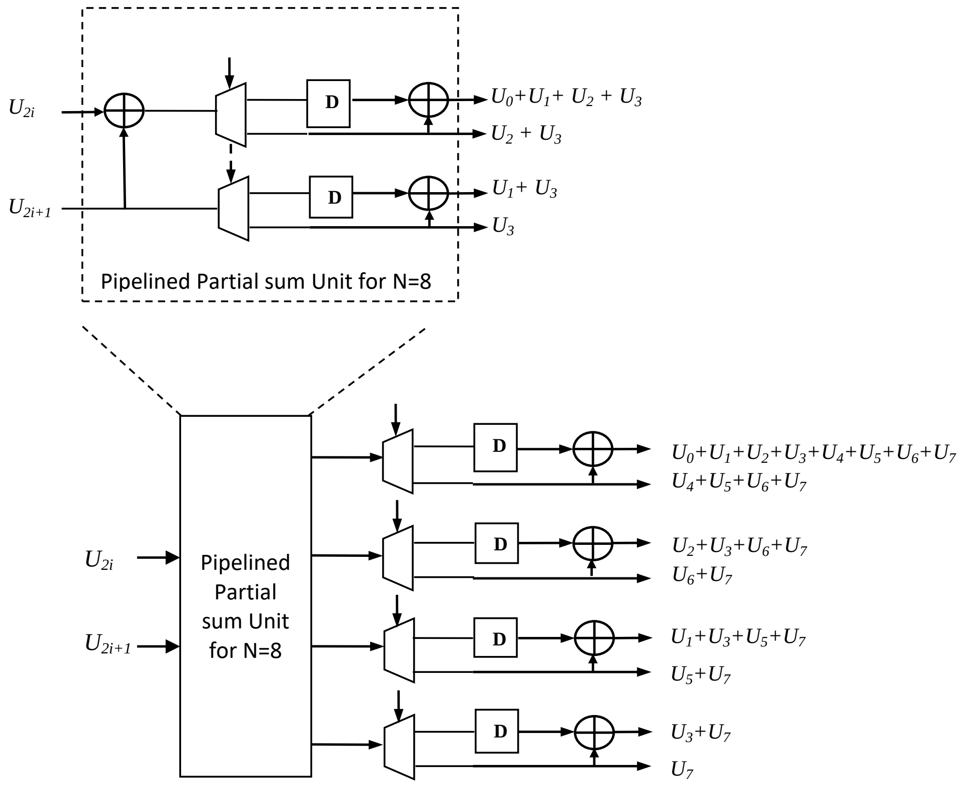

Figure 5. Pipelined partial-sum unit for $N = 16$.

4.5. Frozen channel ROM

The output of PE passes through the D node, which produces two bits based on the frozen inputs from the ROM. A two-bit ROM of size *N/2* stores the indices of frozen bits. Whenever the decoder reaches the final stage, the $L_{1,i}$ values of PE processed by D node. It sets the outputs U_{2i}, U_{2i+1} to *0* if the corresponding frozen indices of ROM are *1*, else produces the estimated bits based on D node. ROM can be replaced by RAM to adapt to different channel conditions by replacing the contents of RAM with the different sets of frozen bits.

4.6. Operation of the proposed architecture

Table 3 shows the operation of proposed decoder architecture and the data propagation through the different hardware module for $N = 8$. Initially, the likelihood ratios are loaded in the buffer registers in the bit reversed order. The MUX between the buffer and RAM selects buffer containing LLR as input for the N/P clock cycles. It selects the output of the PE as input for the remaining clock cycle belongs to the decoding phase. The LLRs are loaded simultaneously as a block, while the decoding of LLR takes place by the PE in a pipelined manner. Buffer register allows the decoder to accept the received LLR serially, while the output of the PE is connected to RAM. Then, the LLR is written into the RAM as a memory word, which depends on the quantization bits chosen. The read and write address is given to the RAM to store the LLR without overlapping. Since it is a dual-port RAM, read and write operation takes place in the same clock cycle. The decoder has P PEs depends on the code length. Each PE takes *2Q* bits LLR as input for f and g unit and produces LLR, which is to be written into the RAM as memory word. Since two bits are decoded at the same clock cycle, in the first clock cycle, LLRs $L_{2,0}, L_{2,4}, L_{2,1}, L_{2,5}$ are computed, then $L_{2,2}, L_{2,6}, L_{2,3}, L_{2,7}$ and $L_{1,0}, L_{1,2}, L_{1,1}, L_{1,3}$ are computed in second and third clock cycles, respectively. In fourth and fifth clock cycle, U_0, U_1 and U_2, U_3 are decoded using the D node. Similarly, U_4, U_5 and U_6, U_7 are decoded in the remaining clock cycles. With *look-ahead*, $U_0 - U_3$ and $U_4 - U_7$ can be computed in a single clock cycle leads to two clock cycles can be saved. The PPU can be constructed recursively for an N length SC decoder from the basic module. Here in Figure 5, PPU for $N = 16$ is constructed from PPU for $N = 8$. Whenever the bits U_{2i}, U_{2i+1} are decoded in a particular clock cycle, it is fed to PPU. The corresponding partial-sums from the PPU are stored in the D-FFs for successive decoding. For PPU, though the number of input remains the same, the number of outputs of PPU increases with the code length *N*. Hence, there are P MUXs need to be selected at the appropriate clock cycle, to map the *N/2* partial-sum outputs to partial-sums required for P PEs.

Table 3. Data flow through different hardware modules in the proposed decoder for $N = 8$.

Clk	Read (addr)	Write (addr)	Channel Buffer	RAM	PE	D node	PPU
0			$L_{3,0}....L_{3,7}$				
1		0000		$L_{3,0}, L_{3,4}$			
2		0000		$L_{3,2}, L_{3,6}$			
3		0001		$L_{3,1}, L_{3,5}$			
4		0001		$L_{3,3}, L_{3,7}$			
5	0000	0010		$L_{2,0}L_{2,4}\ L_{2,1}L_{2,5}$	4LLR		
6	0001	0010		$L_{2,2}L_{2,6}\ L_{2,3}L_{2,7}$	4LLR		
7	0010	0011		$L_{1,0}L_{1,2}\ L_{1,1}L_{1,3}$	4LLR		
8	0011					U_0U_1	$U_0 + U_1, U_1$
9	0011					U_2U_3	$U_0 + U_1 + U_2 + U_3, U_1 + U_3$ $U_2 + U_3, U_3$
10	0010			$L_{1,4}L_{1,6}\ L_{1,5}L_{1,7}$	4LLR		
11	0100					U_4U_5	$U_4 + U_5, U_5$
12	0100					U_6U_7	$U_4 + U_5 + U_6 + U_7, U_5 + U_7$ $U_6 + U_7, U_7$

5. Hardware analysis and comparison

Table 4 shows the hardware comparison of the proposed architecture with the decoders reported in the previous literature. In the decoders, $Q = \{4, 5, 6\}$ bit quantizations can be used for the channel LLR. It is inferred that there is an increase in utilization rate in the proposed decoder as the code length increases since the implemented PEs is limited to $P(\ll N)$. The throughputs are obtained for all the decoders and normalized to the conventional SC decoder [7]. Among the proposed scheme, the *precomputation* and *look-ahead* has a better utilization rate and normalized throughput comparable to the tree and line-based architecture. And the latency of the proposed decoder with *precomputation*, *precomputation* with *look-ahead* are *N*, *0.75N* approximately obtained by neglecting the insignificant logarithmic component for up to (1024, 512) polar code with $P = 64$ PEs.

Table 5 shows the performance comparison of the proposed semi-parallel decoder with existing decoders. The latency and utilization rate for different code length up to 1024 are compared. The performance parameters of the conventional semi-parallel decoder are obtained from [10]. The latency of the *2b-SC* decoder [13] of code length N is *0.75N-1*. The utilization rate of *2b-SC* decoder is obtained from $\alpha = \frac{Required\ node\ updates}{Resource\ complexity\ available*Latency} = \frac{N log_2 N}{2(N-2)*(0.75N-1)}$. For proposed semi-parallel decoder, the latency and utilization rate for any code length N can be obtained from eqns(5) and (7), respectively. As the code length varies from *64* to *1024*, significant improvement in utilization rate is achieved at the cost of an increase in latency by fewer cycles negligible for smaller code length. It shows that, for a code length of *1024*, utilization rate of proposed semi-parallel decoder α is improved by $\frac{utilization\ rate_{prop.} - utilization\ rate_{17}}{utilization\ rate_{prop.}} = 94\%$ than *2b-SC* decoder [17] but at the cost of an increase in latency by *2%* which is insignificant as compared to utilization rate achieved as shown in Figure 6 and Figure 7. Similarly, the proposed work shows a *62.7%* improvement in utilization rate than conventional semi-parallel architecture [10]. Figure 8 shows the proposed decoder has an improved utilization rate than the semi-parallel decoder in [10] when a number of PEs P varied from 1 to 128 for code length up to 2^{20} . It shows that the utilization rate increases as the code length increases and when the number of PE decreases. Here, selecting the appropriate number of PE is necessary, since choosing a lesser PEs increases the latency, and higher PE increases the processing complexity.

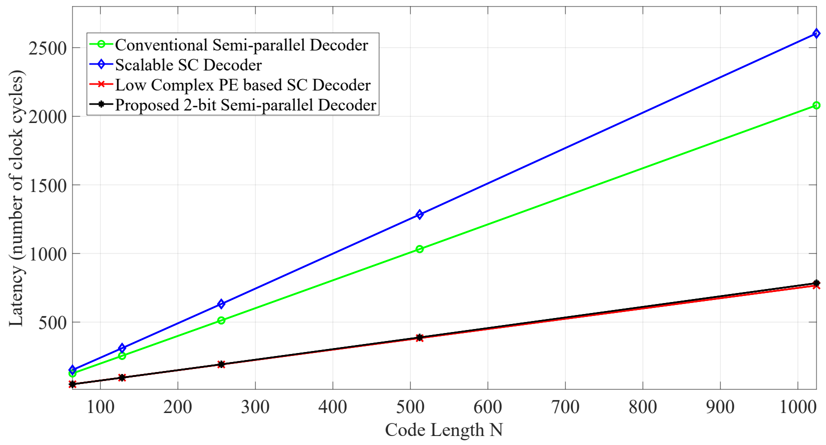

Figure 6. Latency comparison of proposed decoder with prior decoders for varying code length *N*.

Table 6 reports the device utilization of the decoder for the target device and compared only with the

Table 4. Hardware comparison of the proposed semi-parallel architecture with prior architectures.

Hardware		[7]	[13]			Proposed Architecture	
		SC	*2b-SC*	*2b-SC* with overlapping	*2b-SC* *precomputation*	*precomputation*	*precomputation* with *look-ahead*
No. of PE	Tree	$N-1$	$N-2$	$N-2$	$N-2$	$P=64$	$P=64$
	Line	$N/2$	$N/2$	$N/2$	$N/2$		
No. of D node		0	1	1	5	1	5
No. of 1 bit Registers		$\sim QN$	$\sim QN$	$\sim QN$	$\sim 3QN$	$\sim 3QN$	$\sim 3QN$
Latency(cycle)		$2N-2$	$1.5N-2$	$N-1$	$0.75N-1$	$\sim N$	$\sim 0.75N$
Utilization rate (α)	Tree	$\frac{\log_2 N}{2N}$	$\frac{\log_2 N}{3N}$	$\frac{\log_2 N}{2N}$	$\frac{\log_2 N}{1.5N}$	$\frac{\log_2 N}{2P}$	$\frac{\log_2 N}{1.5P}$
	Line	$\frac{\log_2 N}{N}$	$\frac{\log_2 N}{1.5N}$	$\frac{\log_2 N}{N}$	$\frac{\log_2 N}{0.75N}$		
Normalized Throughput[a]		1	1.33	2	2.67	2	2.67

[a]Throughput of architecture in [13] and the proposed architecture are normalized with respect to the SC decoder in [7].

Table 5. Performance comparison of the proposed decoder with conventional semi-parallel and *2b-SC* decoders.

Code length N	Latency (No. of clock cycles)				Utilization rate (α)			
	[10]	[19]	[17]	Proposed decoder	[10]	[19]	[17]	Proposed decoder
64	126	151	47	47	0.023	0.039	0.066	0.064
128	254	310	95	95	0.026	0.023	0.037	0.074
256	512	632	191	192	0.031	0.025	0.021	0.083
512	1032	1284	383	388	0.035	0.028	0.012	0.093
1024	2080	2604	767	784	0.038	0.031	0.006	0.102

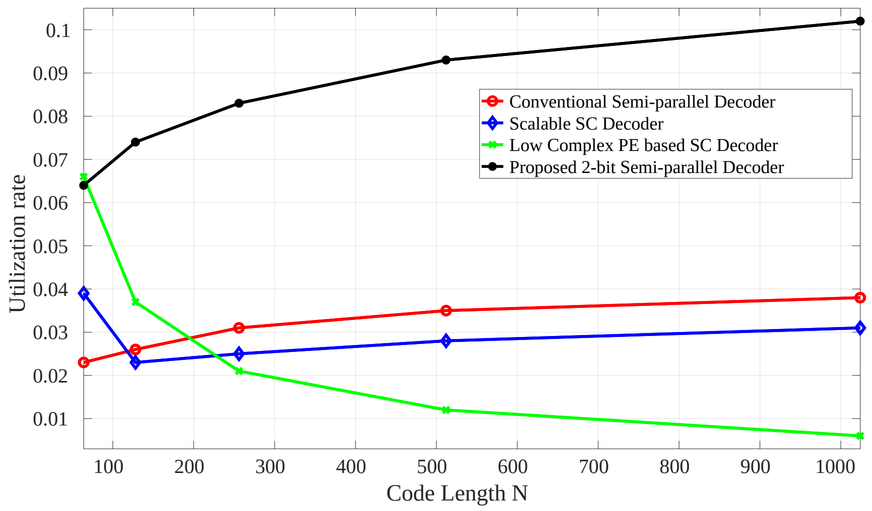

Figure 7. Utilization rate comparison of proposed decoder with prior decoders for varying code length N.

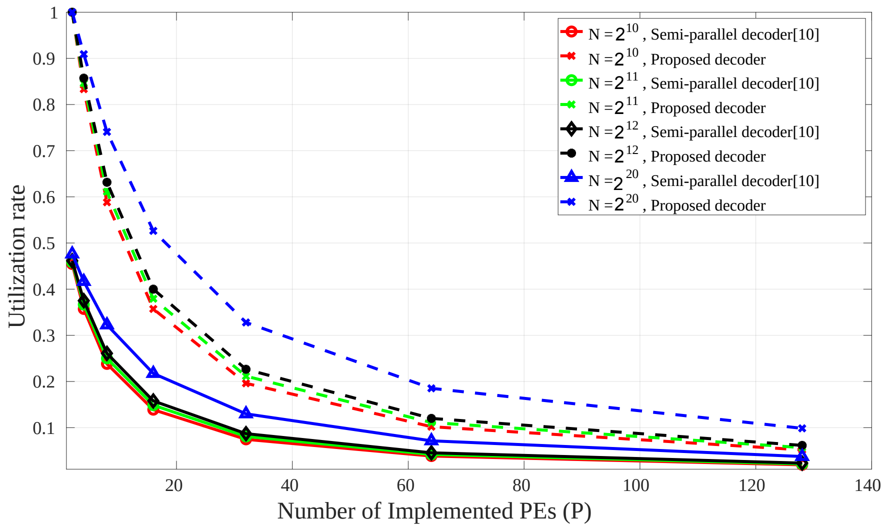

Figure 8. Utilization rate comparison of proposed decoder with conventional semi-parallel decoder in [10] for different number of PEs P.

existing work on Virtex-6 FPGA. For a code length of $N = 2^{10}$, the proposed decoder utilized only 1.69% of LUTs, 0.28% of FFs and 0.24% of BRAM, in XC6VLX240T, which is 88.4%, 89.2% and 98.3% lesser than decoder in [14] respectively. The decoder in [15] uses XC6VLX550T, which is two times bigger than XC6VLX240T. The decoder is also implemented for a code length of $N = 2^{17}$ and utilized only 1.87% of LUTs, 0.28% of FFs, but 18.5% of BRAM utilized in XC6VLX240T. Higher RAM usage may impact power consumption and energy efficiency. Hence, the power consumption and energy-per-bit are also reported for the proposed design. The on-chip power can be estimated from the Xilinx power estimator tool for the logic resources used in our design. The energy-per-bit can be obtained from the expression $E_{bit}(J/b) = \frac{Power(W)}{Throughput(b/s)}$. As the code length increases, the power consumption increases which increases the energy-per-bit E_{bit} and thus reduces the energy efficiency. Power consumption and energy-per-bit are discussed for ASIC in [15], but for FPGA it is not reported in [14] and [15].

Logic synthesis, mapping, placement, and routing were performed in Virtex-6 FPGA. It has similar 40nm technology with Stratix-IV FPGA has been chosen to compare with prior architectures. The tools used are Matlab to generate LRs for decoder input, Modelsim for functional verification, Xilinx ISE 15.4 for synthesis and implementation. Moreover, worse case environment such as maximum junction temperature of $85°C$, low-voltage complementary metal oxide semiconductor (LVCMOS) 2.5V 12mA (Slow) model for I/O standards and internal supply voltage of 0.87 V has been chosen for implementation in FPGA. Different quantization schemes were used in the decoders denoted by (Q_i, Q_{ic}, Q_f) as shown in Table 7, where Q_i is the number of integer bits in internal LLR, Q_{ic} is the number of integer bits in channel LLR and Q_f is the number of fractional bits in both. The proposed decoder is validated by choosing LLR of Quantization bits $(Q_i, Q_{ic}, Q_f) = (5, 5, 0)$ as input to the decoder. LLR is generated in Matlab by constructing polar code optimized for specific signal-to-noise ratio (SNR) and encoding a random message vector of code length $N = 1024$ with varying code rate $R = K/N$ passed through an AWGN channel. Noisy codewords were generated for the range of SNR values of *0* dB to *3.5* dB. The received LLR from the channel is saturated to the range [-2,2] having acceptable error correction

Table 6. Comparison of device utilization of the proposed decoder with the existing decoder in Virtex-6 FPGA.

Target Device	Logic utilization	Used	Available	Utilization %	Power consumption (W)	Energy-per-bit E_{bit} (nJ/bits)
Proposed *look-ahead* decoder for $N = 2^{17}$						
XC6VLX240T	LUT	2,812	150,720	1.87	6.457	58.3
	FF	871	301,440	0.28		
	BRAM_36K	77	416	18.5		
Proposed *look-ahead* decoder for $N = 2^{10}$						
XC6VLX240T	LUT	2,544	150,720	1.69	6.380	37.7
	FF	830	301,440	0.28		
	BRAM_36K	1	416	0.24		
decoder from [14] for $N = 2^{10}$						
XC6VLX240T	LUT	22,115	150,720	14.6	N/A	N/A
	FF	7,941	301,440	2.6		
	BRAM_36K[b]	59	416	14.2		
decoder from [15] for $N = 2^{10}$						
XC6VLX550T	LUT	190,127	343,680	55.3	N/A	N/A
	FF	22,928	687,360	3.3		
	BRAM_36K[b]	1	632	0.15		

[b]The block RAM count is estimated by $BRAM\_36K = \lceil \frac{RAM(bits)}{36000} \rceil$.

performance as shown in Figure 9 for different quantization bits. It is inferred that bit error rate (BER) and frame error rate (FER) performance is degraded by only *0.25* dB for *5* bit quantization compared to floating point decoder. The equivalent quantization bits of $Q = 5$ for the received LLR are given along with necessary inputs to the decoder and the estimated bits are compared with the message vectors for the error correction performance. The critical path of the proposed architecture consists of RAM, PE, D node, PPU and MUX logic which takes *8.9 ns* to produce the decoded output leads to the maximum operating frequency of *112 MHz* and after applying the *look-ahead* technique with D node reduces the critical path delay to *6.9 ns* and improves the maximum operating frequency to *144 MHz*. Also, the proposed architecture shows improved throughput performance in terms of code rate R compared to the architecture [10, 19].

Table 7 shows the comparison of resource utilization and performance of proposed architecture with the prior architecture. The maximum throughput of the proposed decoder approaches *188 Mbps* for a code rate R approaches the maximum value. The proposed decoder is compared with the different methods implemented in the literatures, the work [14, 20–22] and [15] achieves very high throughput using Fast-SSC algorithm and combinational pipelined architecture, respectively, but at a cost of higher LUTs and FFs usage than the proposed decoder. However, the developed semi-parallel design is compared with prior semi-parallel implementation in [10] for $N = 2^{10}$. The proposed method achieves 2.2 times higher throughput than [10] with the *look-ahead* techniques. Both the proposed scheme requires fewer LUTs and FFs than the prior architecture shows that only

Table 7. Comparison of implementation results of proposed decoder with prior decoders in FPGA.

References	N	Algorithms	FPGA	(Q_i, Q_{ic}, Q_f)	LUT	FF	RAM (bits)	f (MHz)	Throughput (Gbps)
[14]	2^{10}	Fast SSC	Virtex-6	(6,5,1)	22,115	7,941	2,106,000	70.2	0.436
[22]		Fast SSC	Stratix-V	(5,6,1)	81,498	96,762	2,367,488	300	307.2
[21]		Fast SSC	Stratix-IV	N/A	155,858	158,185	285,120	206	105.3
[20]		Fast SSC	Stratix-IV	(6,4,1)	14,300	1,216	18,350	89.57	0.431
[18]		SCL	Stratix-V	(7,7,1)	33,502	5,515	11,264	N/A	N/A
[15]		SC	Virtex-6	(5,5,0)	190,127	22,928	13,312	N/A	1.24
[23]	2^{15}	SC	Stratix-IV	(7,7,0)	17,980	6,241	557,568	130	$0.133R$
	2^{16}			(6,6,0)	13,630	5,240	984,064	125	$0.124R$
[19]	2^{15}	SC	Stratix-IV	(6,4,0)	3,263	1,304	411,648	167	$0.062R$
	2^{16}			(5,5,0)	2,866	1,254	820,992	170	$0.062R$
	2^{17}				2,714	1,263	1,640,192	160	$0.058R$
[10]	2^{10}	SC	Stratix-IV	(5,5,0)	4,130	1,691	15,104	173	$0.085R$
	2^{15}				58,480	33,451	364,288	66	$0.031R$
	2^{16}				114,279	66,223	724,736	56	$0.026R$
	2^{17}				221,471	131,764	1,445,632	10	$0.0046R$
Proposed decoder	2^{10}	SC	Virtex-6	(5,5,0)	**2,529**	**830**	31,104	112	$0.11R$
	2^{15}				**2,612**	**846**	697,728	97	$0.091R$
	2^{16}				**2,689**	**858**	1,385,856	89	$0.084R$
	2^{17}				**2,797**	**871**	2,762,112	77	$0.072R$
Proposed *look-ahead* decoder	2^{10}	SC	Virtex-6	(5,5,0)	**2,544**	**830**	31,104	144	$0.188R$
	2^{15}				**2,627**	**846**	697,728	128	$0.159R$
	2^{16}				**2,704**	**858**	1,385,856	116	$0.143R$
	2^{17}				**2,812**	**871**	2,762,112	101	$0.123R$

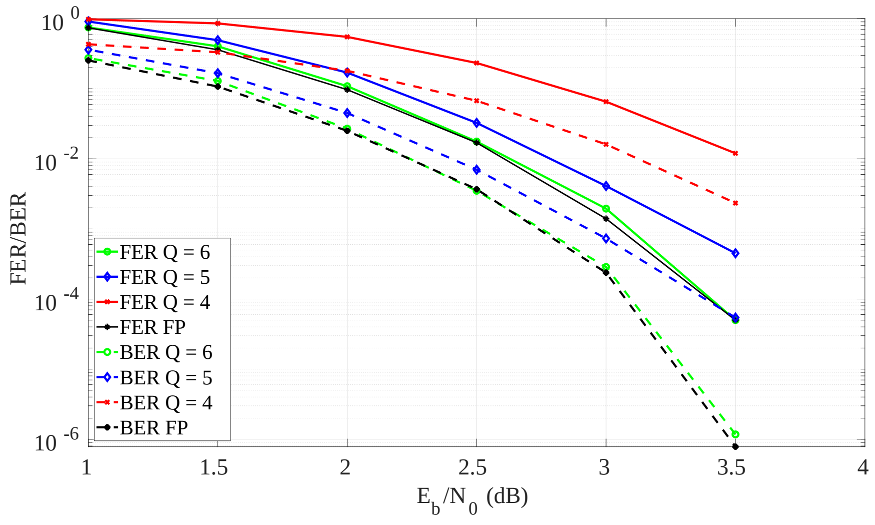

Figure 9. Effect of quantization on BER performance of polar code with code length $N = 1024$ and code rate $R = 0.5$.

87% of LUTs with *49*% of FFs are required compared to what was required for architecture [10]. The proposed architecture shows 98% and 89% lower RAM usage than the ones in [14, 22] and [21], respectively. On the other hand, the decoder shows *50 - 60%* more RAM usage than other decoder implementations in [10, 15, 18, 20] because of *precomputation* techniques employed in the proposed method. Moreover, the proposed design is also implemented for higher code length. LUTs and FFs are reduced by 98%; throughput increased by 26 times and the RAM utilization is increased by 48% as compared to semi-parallel architecture in [10] for $N = 2^{17}$. As compared to the decoder in [19], it is realized in our decoder that FFs are reduced by 34%, throughput increased by twice, and RAM usage is increased by 40%. The work [23] reports implementation results of the decoder up to $N = 2^{16}$. It shows an increase in LUT and FF usage by 80-84% and a reduction in RAM usage by 29% with comparable throughput than the proposed decoder for the same code length.

6. Conclusion

In this work, a modified semi-parallel SC decoder with *precomputation* and *look-ahead* techniques are implemented. *Precomputation* of the internal LLRs reduces the latency of each stage by half and the *look-ahead* further reduces the latency for each decision in the last-stage. The PPU is used to compute partial-sum for every incoming two bits reduces the critical path delay. The analytical expressions provided for the latency and utilization rate of the proposed decoder based on the scheduling schemes verifies the implementation results. Hardware analysis and performance comparison is done for the proposed decoder. It shows improved utilization rate and latency than the prior decoder architecture. The developed architecture for code length up to 2^{17} with varying code rate are synthesized in Virtex-6 FPGA and compared with other semi-parallel architectures. It is evident from the results that a very large efficient decoder can be implemented with fewer LUTs and FFs.

References

[1] 3rd Generation Partnership Project; Technical Specification Group Radio Access Network; NR; Multiplexing and channel coding (Release 15)

[2] Hsu C, Anastasopoulos A. Capacity Achieving LDPC Codes Through Puncturing. IEEE Transactions on Information Theory 2008; 54 (10): 4698-4706. doi: 10.1109/TIT.2008.928274

[3] Arikan E. Channel Polarization: A Method for Constructing Capacity-Achieving Codes for Symmetric Binary-Input Memoryless Channels. IEEE Transactions on Information Theory 2009; 55 (7): 3051-3073. doi: 10.1109/TIT.2009.2021379

[4] Arikan E, Telatar E. On the rate of channel polarization. In: IEEE 2009 International Symposium on Information Theory; Seoul; 2009. pp. 1493-1495. doi: 10.1109/ISIT.2009.5205856

[5] Hussami N, Korada SB, Urbanke R. Performance of polar codes for channel and source coding. In: IEEE 2009 International Symposium on Information Theory; Seoul; 2009. pp. 1488-1492. doi: 10.1109/ISIT.2009.5205860.

[6] Tal I, Vardy A. List Decoding of Polar Codes. IEEE Transactions on Information Theory 2015; 61 (5): 2213-2226. doi: 10.1109/TIT.2015.2410251

[7] Leroux C, Tal I, Vardy A, Gross WJ. Hardware architectures for successive cancellation decoding of polar codes. In: IEEE 2011 International Conference on Acoustics, Speech and Signal Processing; Prague; 2011. pp. 1665-1668. doi: 10.1109/ICASSP.2011.5946819

[8] Pamuk A. An FPGA implementation architecture for decoding of polar codes. In:2011 8th International Symposium on Wireless Communication Systems; Aachen; 2011. pp. 437-441. doi: 10.1109/ISWCS.2011.6125398

[9] Yuan B, Parhi KK. Successive cancellation list polar decoder using log-likelihood ratios. In: 2014 48th Asilomar Conference on Signals, Systems and Computers. Pacific Grove, CA; 2014. pp. 548-552. doi:10.1109/ACSSC.2014.7094505

[10] Leroux C, Raymond AJ, Sarkis G, Gross WJ. A Semi-Parallel Successive-Cancellation Decoder for Polar Codes. IEEE Transactions on Signal Processing 2013; 61 (2): 289-299. doi: 10.1109/TSP.2012.2223693

[11] Berhault G, Leroux C, Jego C, Dallet D. Partial sums generation architecture for successive cancellation decoding of polar codes. SiPS 2013 Proceedings; Taipei City; 2013. pp. 407-412. doi: 10.1109/SiPS.2013.6674541

[12] Zhang C, Yuan B, Parhi KK. Reduced-latency SC polar decoder architectures. In:2012 IEEE International Conference on Communications; Ottawa, ON; 2012. pp. 3471-3475. doi: 10.1109/ICC.2012.6364209

[13] Yuan B, Parhi KK. Low-Latency Successive-Cancellation Polar Decoder Architectures Using 2-Bit Decoding. IEEE Transactions on Circuits and Systems I: Regular Papers 2014; 61 (4): 1241-1254. doi: 10.1109/TCSI.2013.2283779

[14] Giard P, Sarkis G, Thibeault C, Gross WJ. A 638 Mbps low-complexity rate 1/2 polar decoder on FPGAs. In: IEEE 2015 Workshop on Signal Processing Systems; Hangzhou; 2015. pp. 1-6. doi: 10.1109/SiPS.2015.7345007.

[15] Dizdar O, Arıkan E. A High-Throughput Energy-Efficient Implementation of Successive Cancellation Decoder for Polar Codes Using Combinational Logic. IEEE Transactions on Circuits and Systems I: Regular Papers 2016; 63 (3): 436-447. doi: 10.1109/TCSI.2016.2525020

[16] Hussein GH Hassan, Amr MA Hussien, Hossam AH Fahmy. A simplified radix-4 successive cancellation decoder with partial sum lookahead, AEU - International Journal of Electronics and Communications 2018; 96: 267-272. doi: 10.1016/j.aeue.2018.09.027

[17] Geethu SB, Lakshmi RM, Lakshminarayanan G, Mathini S. Low-complex processing element architecture for successive cancellation decoder. Integration the VLSI Journal 2019; 66 : 80-87. doi: 10.1016/j.vlsi.2019.01.005

[18] Zhou H, Liang X, Li L, Zhang Z, You X et al. Segmented Successive Cancellation List Polar Decoding with Tailored CRC. Journal of Signal Processing System 2019; 91: 923–935. doi: 10.1007/s11265-018-1425-0

[19] Raymond AJ, Gross WJ. A Scalable Successive-Cancellation Decoder for Polar Codes. IEEE Transactions on Signal Processing 2014; 62 (20): 5339-5347. doi: 10.1109/TSP.2014.2347262

[20] Ercan F, Condo C, Gross WJ. Reduced-memory high-throughput fast-SSC polar code decoder architecture. In: 2017 IEEE International Workshop on Signal Processing Systems; Lorient; 2017. pp. 1-6. doi: 10.1109/SiPS.2017.8110014

[21] Giard P, Sarkis G, Thibeault C, Gross WJ. 237 Gbit/s unrolled hardware polar decoder. Electronics Letters 2015; 51 (10): 762-763. doi: 10.1049/el.2014.4432

[22] Zhang X, Yan X, Zeng Q, Cui J, Cao N et al. High-Throughput Fast-SSC Polar Decoder for Wireless Communications. Wireless Communications and Mobile Computing 2018; Article ID 7428039, 10 pages. doi: 10.1155/2018/7428039

[23] Delomier Y, Gal BL, Crenne J, Jego C. Model-based Design of Hardware SC Polar Decoders for FPGAs. ACM Transactions on Reconfigurable Technology and Systems (TRETS) 2020; 13 (2): 1-27. doi: 10.1145/3391431