Hardware Software Co-design of 5G NR Polar Encoder and Decoder on System-on-Chip

A THESIS

SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR

THE DEGREE OF

M.Tech

BY
MANSI

Electronics and Communication Engineering
(Specialization - VLSI and Embedded Systems)

INDRAPRASTHA INSTITUTE OF INFORMATION TECHNOLOGY DELHI
NEW DELHI– 110020

June 30, 2022

Certificate

This is to certify that the thesis titled **"Hardware Software Co-design of 5G NR Polar Encoder and Decoder on System-on-Chip"** submitted by **Mansi** for the partial fulfillment of the requirements of *Master of Technology in Electronics and Communications Engineering* is a record of the bonafide work carried out by her under my guidance and supervision at Indraprastha Institute of Information Technology, Delhi. This work has not been submitted anywhere else for the reward of any other degree.

Dr. Sumit J Darak **Date: June 30, 2022**
Associate Professor
Department Of Electronics and Communication
Indraprastha Institute of Information Technology, Delhi

Acknowledgement

I would like to take this opportunity to express my sincere gratitude to the people who have supported me during my thesis work. Foremost, I would like to express my sincere gratitude to my advisor, Dr. Sumit Darak, for his invaluable guidance, encouragement, and support throughout my thesis work. His constructive feedback and regular work discussions helped me accomplish the task with great clarity. I would like to thank Mr. Khagendra Joshi for the quick access to the required resources for my thesis work. I would also like to thank Daksh Kumar and Aryan Govil for assisting in completing my thesis work. Last but not least, I would like to thank my family and friends for supporting me through challenging times and for their guidance in managing my thesis work efficiently.

Abstract

Channel encoder and decoder are critical components of the wireless physical layer (PHY), enabling the transceiver to overcome transmission errors due to wireless channel fading and interference. In 5G, polar encoder and decoder are chosen for control information, while Low-Density Parity Check (LDPC) encoder and decoder are chosen for data information. Depending on the deployment platform of wireless PHY, software and hardware IP cores of these channel coders are desired. To the best of our knowledge, such IP cores are available commercially from a couple of sources only. These cores are expensive with fewer features, encrypted source codes, and hence, limited flexibility. The availability of alternate low-cost, efficient open-source IP cores is critical for indigenous 5G activities in Industry and research activities in academia. The work presented in this thesis focuses on the design and experimental validation of software and hardware IP cores for 5G polar encoder and decoder on system-on-chip (SoC).

In this work, we have developed IP cores for polar encoder and decoder, rate matching and dematching, data modulator, and demodulator for downlink and uplink links. These IP cores are flexible to support various application scenarios in the 5G 3GPP specifications. The software IP cores are compatible for realization on any processor, and we have done the performance validation on dual-core ARM Cortex A9 and quad-core ARM Cortex A53 processors. The hardware IP cores are synthesizable on 7-series and Ultra-scale families of the field-programmable gate arrays (FPGA). The hardware IP cores are optimized via pipelining, memory tiling, and word length optimization. We have experimentally validated the functional correctness of the proposed IP cores for a wide range of signal-to-noise ratio (SNR), coding rates, and word lengths.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Motivation

Channel encoder and decoder are critical components of the wireless physical layer (PHY), enabling the transceiver to overcome transmission errors due to wireless channel fading and interference. This is done by adding the redundant bits along with the information bits and randomizing their positions in the transmitted signal. Various types of channel coders such as convolutional coder, turbo coder [2], and Low Density Parity Check (LDPC) coder [3] have been used in wireless standard. In 5G [4], 3^{rd} Generation Partnership Project (3GPP) has chosen polar codes for control channels and LDPC codes for data channels compared to Turbo coders used in 4G [5]. Owning to significant changes in 5G channel coders compared to 4G, there are various open research and design challenges related to software and hardware IP cores of these channel coders.

Polar Codes, invented by Arikan in 2008 [6], are based upon channel polarization, where the physical channels are transformed into virtual noiseless channels. Depending on the deployment platform of wireless PHY, software and hardware IP cores of these channel coders are desired. To the best of our knowledge, such IP cores are available commercially from a couple of sources only. Please refer to Sec. 1.3 for more details. These cores are expensive with fewer features, encrypted source codes, and hence, limited flexibility. The availability of alternate low-cost, efficient open-source IP cores is critical for indigenous 5G activities in Industry and research activities in academia. The work presented in this thesis focuses on the design and experimental validation of software and hardware IP cores for 5G polar encoder and decoder on system-on-chip (SoC).

1.2 Objectives and Contribution

The main objectives of the thesis are -

1. To understand the mathematical framework and application scenarios of the polar encoder and decoder algorithms as per 3GPP specifications.
2. Design and optimize software and hardware IP cores for channel encoder and decoder on the chosen platform
3. Experimental performance and complexity analysis of the proposed IP cores.

The contribution of the thesis can be summarized as follows. In this work, we have developed IP cores for polar encoder and decoder, rate matching and dematching,

data modulator, and demodulator for downlink and uplink links. These IP cores are flexible to support various application scenarios in the 5G 3GPP specifications. The software IP cores are compatible for realization on any processor, and we have done the performance validation on dual-core ARM Cortex A9 and quad-core ARM Cortex A53 processors. The hardware IP cores are synthesizable on 7-series and Ultra-scale families of the field-programmable gate arrays (FPGA). The hardware IP cores are optimized via pipelining, memory tiling, and word length optimization. Complete system-level demonstration of wireless PHY in the presence of a wireless channel is done on Zynq SoC. We have experimentally validated the functional correctness of the proposed IP cores for a wide range of signal-to-noise ratio (SNR), coding rates, and word lengths.

1.3 Literature Review

After invention of Polar codes in 2008, various works have been done at the algorithm and architecture levels. Most of the works have been focused on improving the performance and complexity of the Polar decoders. As an improvement to the Successive Cancellation (SC) decoding algorithm, the list decoding was proposed in [7]. In Successive Cancellation List (SCL) decoding, a certain list of candidate codewords is maintained by pruning the less likely paths. Selecting the final codeword from the list of candidate codewords can be done via CRC-Aided SCL decoder [8]. Various work have discussed the hardware implementation of the polar decoding and encoding algorithm [9–17]. The existing work focused on the hardware implementation of SC decoder [9–13], Believe Propagation(BP) [14] and soft-decision decoder [15]. In [16], the authors proposed a hardware architecture of the SCL decoding algorithm. In [17], the hardware implementation of an Asymmetric Adaptive SCL Decoder design is proposed. In this, the authors have used both SC and SCL decoders together. The particular codeword is returned if the incoming packet decoded by the SC decoder satisfies the CRC. If it does not satisfy the CRC, the SCL decoding is applied. This can reduce the overall latency for some blocks by avoiding the complexity of SCL. To the best of our knowledge, limited work has been done to have software and hardware IP cores for complete Polar encoder and decoder algorithms. As shown in Fig. 1.1, commercial IP cores available from Intel, AMD-Xilinx and Accelercomm are expensive with limited features, and encrypted source codes results in limited flexibility.

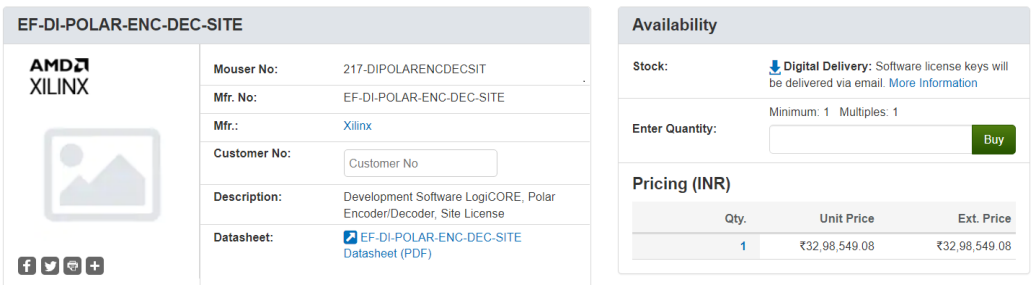

Figure 1.1: Price Quotation for Polar Encoder and Decoder IP from Xilinx [1]

1.4 Thesis Organization

The thesis is organized as follows - Chapter 2 gives an overview of the NR Polar Encoder and Decoder Algorithm. Chapter 3 presents the proposed hardware architectures for Polar Encoder and Polar Decoder along with the respective IP documentations. In Chapter 4, the hardware validation results and word length analysis is discussed in detail. Chapter 5 concludes the thesis along with the future research directions.

Chapter 2

NR Polar Coding Algorithm

In this chapter, we discuss the polar coding technique chosen for 5G New Radio(NR) communication systems according to the 3GPP specifications. With 5G, an average throughput of 100 $Mbps$ and a maximum throughput of 20 $Gbps$ can be acheived.

2.1 Block Diagram

The polar channel coding is used for encoding control information in 5G NR. The polar codes are applied to the uplink and downlink control information (UCI/DCI) for the enhanced mobile broadband (eMBB). Fig. 2.1 illustrates the block diagram overview of the polar coding.

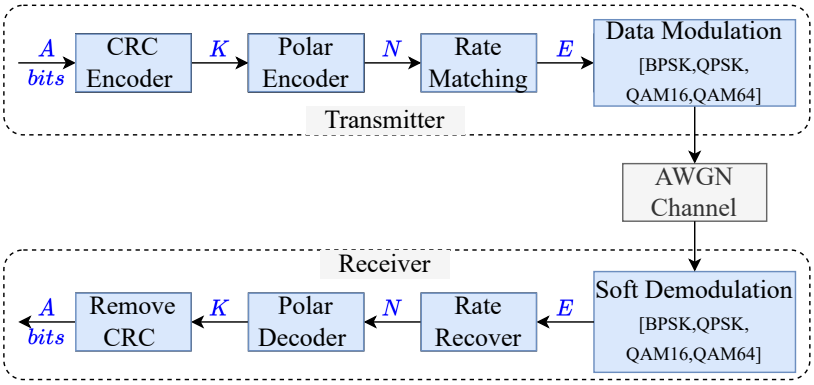

Figure 2.1: Block Diagram for Polar Coding

Any Polar Code is given by P(N, K), where K is the dimension of the code and N is the length of the code. First, the Uplink or Downlink information block, **blk** of length A is appended with $crcLen$ Cyclic Redundancy Check (CRC) parity bits. The CRC encoded information block, **blkCRC** of length K ($A + crcLen$) is polar encoded to a codeword, $\mathbf{c} = c_0^{N-1}$ of length N. As per the input Rate(K/E), rate-matching is done to transmit rate-matched block, $\mathbf{c}_{RM}$ of length E. The rate-matched output is data modulated, and the Additive White Gaussian Noise (AWGN) is added according to the specified Signal to Noise Ratio (SNR) value. In the modulation scheme, we can choose between BPSK, QPSK, QAM16, and QAM64. The signal is then soft demodulated to obtain E log-likelihood ratios (LLRs). The rate-recovery is done to output LLR vector, $\mathbf{y}$ of length N. Successive Cancellation List (SCL) decoding is performed on the received values, $\mathbf{y}$ to retrieve the information block of length K. We get the recovered information block of length A on the removal of CRC bits. As per the 3GPP specifications, the limit on the parameters is listed in table 2.1.

Parameter	Limits		Description
	UCI	DCI	
K	18-1023	36-164	For UCI, 26<=K<=30 is not allowed
crcLen	6,11	24	For downlink 24C polynomial is used for CRC Encoding.
N	1024	512	Must be greater than K and always a power of 2
E	8192		Must be greater than K
iIL	False	True	Interleave before encoding
iBL	True	False	Interleave after encoding
nMax	10	9	
L	2-8		List Size for SCL decoding

Table 2.1: Paramter Limits for UCI and DCI

2.2 Transmitter End Processing

2.2.1 Polar Encoder

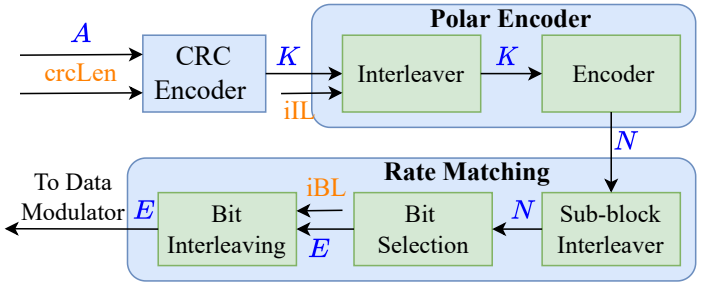

Figure 2.2: Transmitter End Processing

Based upon the $crcLen$, the CRC bits are appended to the input block, **blk**. The possible values of $crcLen$ for UCI and DCI are mentioned in table 2.1. Polar Encoder is divided into Interleaver and Encoder. The parameter iIL specifies if interleaving must be done prior to encoding. For DCI, the input bits are interleaved prior to polar encoding, hence $iIL = 1$ for DCI. Interleaving is required to prevent burst errors. In interleaving, symbols are rearranged as per the specified interleaving pattern in the 3GPP standard. The interleaved output is then polar encoded to a codeword of length N. The codeword, **c**, is given as the Kronecker Product of the $1 \times N$ information vector, **u**=$(u_\phi)_{\phi=0}^{n-1}$ and $N \times N$ Generator Matrix, $\mathbf{G}_N$.

$$c = u * G_N \tag{2.1}$$

Based upon the value of K, we can classify the encoders into CRC Aided Polar (CA Polar) and Parity Check Polar (PC Polar). In the CA Polar, only CRC bits are appended as parity at the end of the input message. In PC Polar, additional parity bits and weighted parity bits are also taken into account. In polar codes, the reliability sequence implies channel polarization. In the case of CA Polar, the N bit information vector is formed by taking into account channel polarization as follows -

- First, the frozen bit locations are estimated based on the $N - K$ least reliable channels from the reliability sequence.
- At the frozen bit locations, assign 0 and at other K locations, add the CRC encoded information bits.

In case of PC Polar, the additional parity bits are also added at certain locations. The Generator Matrix $\mathbf{G}_N$ for polar code of length N is computed as 2.2. Here, $n = log_2(N)$ and the maximum value of n, $nMax$ is 9 for DCI and 10 for UCI.

$$G_N = \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix}^{\otimes n} \tag{2.2}$$

2.2.2 Rate Matching

Rate Matching is done for the codeword of length N to transmit E bits. Since N is usually very large, rate matching reduces the number of transmitted bits, thereby saving power and bandwidth resources. The rate matching is divided into three steps -

1. Sub-block Interleaver: The codeword is divided into 32 sub-blocks. The bits are sub-block interleaved, and the output is passed to a circular buffer of length N.
2. Bit Selection: In this, depending upon the value of E and N, repetition, puncturing, or shortening is done. If $E > N$, the last $E - N$ bits are the repeated values. For $E < N$, either puncturing is done where E bits are taken from the end of the codeword or shortening, where the E bits are taken from the start of the codeword.
3. Bit Interleaving: It is done if the parameter, $iBL = 1$, hence block is valid only for uplink. For UCI, $iBL = 1$ and $iIL = 0$, the interleaving is done after polar encoding.

2.3 Receiver End Processing

After the Soft Demodulation, the LLRs are transmitted for the Receiver End Processing. It consists of two main blocks, Rate Recover and Polar Decoder, discussed below.

2.3.1 Rate Recover

The Rate Recover block is divided into three steps -

1. Bit Deinterleaving: Bit deinterleaving is the opposite of the bit interleaving operation and is done for UCI ($iBL = 1$).

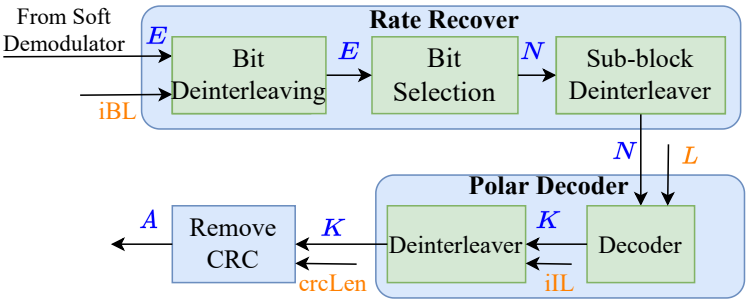

Figure 2.3: Receiver End Processing

2. Bit Selection is done as follows -
   - If $E < N$, either puncturing or shortening is done. In puncturing, $N - E$ zeros are appended at the end of E bits. While in shortening, $N - E$ large values are added at the start of the E bits.
   - If $E > N$, only the first N bits are taken at the output.
3. Sub-block Deinterleaver: In this, the N bits from the circular buffer are individual sub-block is deinterleaved.

2.3.2 Polar Decoder

The LLR vector of length N, $\mathbf{y}$ (output from the Rate Recover block) is then polar decoded via CRC Aided Successive Cancellation decoder [7]. The polar decoder algorithm retrives estimate of information vector, $\mathbf{u}$, $\hat{\mathbf{u}}$ from the input N LLR values. First, we will discuss the Successive Cancellation (SC) decoder and then extend it for Successive Cancellation List (SCL) decoder.

Successive Cancellation Decoder: The SC decoder consists of N bit-channels and the ϕ^{th} bit-channel takes $\mathbf{y}$ and $u_0^{\phi-1}$ as input and computes $\hat{u}_\phi$. At each bit-channel, the probability of the bit being 0 (P_0) of 1 (P_1) is evaluated by iterating over each layer. The greater of the two probabilities decides the estimate at the corresponding bit channel. Fig. 2.5 illustrates the $N = 4$ length SC polar decoding algorithm, stepwise via a tree structure. The terminology used for the N length SC decoder is as follows -

- The various levels of the tree are referred to as layers and layer (λ) range from 0 to m, where, $m = log_2(N)$.
- For each layer, λ, phase, ϕ is defined as $\phi \in [0, 2^\lambda]$.
- Each node of the tree acts as a bit-channel and is defined by layer, λ and phase, ϕ as W_λ^ϕ.
- The LLR input to each bit-channel is denoted by P_λ^ϕ and the estimate at each bit-channel is denoted by B_λ^ϕ. The length of these is equal to $\beta_\lambda = 2^{m-\lambda}$.

The operations for a general $P(N, K)$ polar code shown in Fig. 2.4 can be described as follows -

1. L Operation (Parent,Left Child): In Operation L on (W_{λ}^{ϕ},$W_{\lambda+1}^{\phi^{'}}$), the LLR output $P_{\lambda+1}^{\phi}$ from root is evaluated by the minsum operation in 2.3 is passed to left child, $W_{\lambda+1}^{\phi^{'}}$.
$$f(P_{\lambda}^{\phi}) = f(P_{\lambda\,[0:\frac{\beta_\lambda}{2}-1]}^{\phi}, P_{\lambda\,[\frac{\beta_\lambda}{2}:\beta_\lambda]}^{\phi}) \tag{2.3}$$

2. R Operation (Parent, Right Child): In operation R on (W_{λ}^{ϕ},$W_{\lambda+1}^{\phi^{'}+1}$), the LLR output, $P_{\lambda+1}^{\phi+1}$ is evaluated by the g operation in 2.4.
$$g(P_{\lambda+1}^{\phi}, B_{\lambda+1}^{\phi}) = g(P_{\lambda+1\,[0:\frac{\beta_\lambda}{2}-1]}^{\phi}, P_{\lambda+1\,[\frac{\beta_\lambda}{2}:\beta_\lambda]}^{\phi}, B_{\lambda+1}^{\phi}) \tag{2.4}$$

3. B Operation (Parent) : In Operation B on parent node W_{λ}^{ϕ}, the hard decision values at parent, B_{λ}^{ϕ} are evaluated as $B_{\lambda}^{\phi} = [B_{\lambda+1}^{\phi^{'}} + B_{\lambda+1}^{\phi^{'}+1} \;\; B_{\lambda+1}^{\phi^{'}+1}]$.
In the above description of operations, for phase, ϕ , $\phi^{'} = 2\phi$.

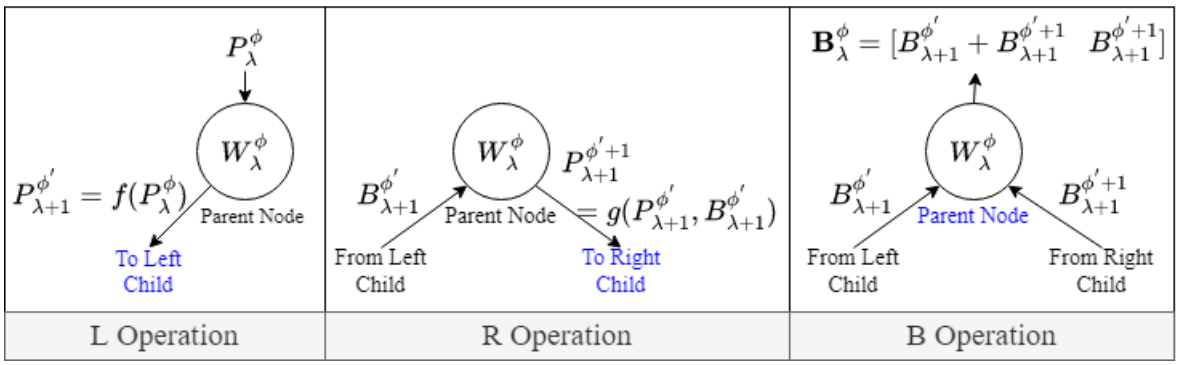

Figure 2.4: Operations for SC Decoder

At each bit-channel, the LLRs are estimated based upon the previously estimated bits and input LLRs. Fig. 2.5 illustrates the step-wise decoding process for $N = 4$. The decoding starts from the root node (W_0^0), where the initial LLR, P_0 is defined based upon the input. After initialization, the steps as in Fig. 2.5 are explained below -

1. In the first step ($\phi = 0$), the evaluation of P_2^0 is done by iterating over each layer. The L operation on (W_0^0, W_1^0) and (W_1^0, W_2^0) gives P_2^0. At the leaf node, the decision for the $\hat{u}_0$ is made based upon the value of P_2^0. If the value is greater than 0, $\hat{u}_0 = 0$ else $\hat{u}_0 = 1$.

2. In the second step, the evaluation for the next phase, $\phi = 1$. Here, the estimates made from the last layer will be used for the calculation of LLR, P_2^1 (R operation). The hard decision for $\hat{u}_1$ is made at the leaf node as per the value P_2^1. For leaf nodes with an odd phase value, the estimate vector, $B_2^1 = [\hat{u}_0 + \hat{u}_1 \; \hat{u}_1]$ is updated by the B-operation on W_2^1.

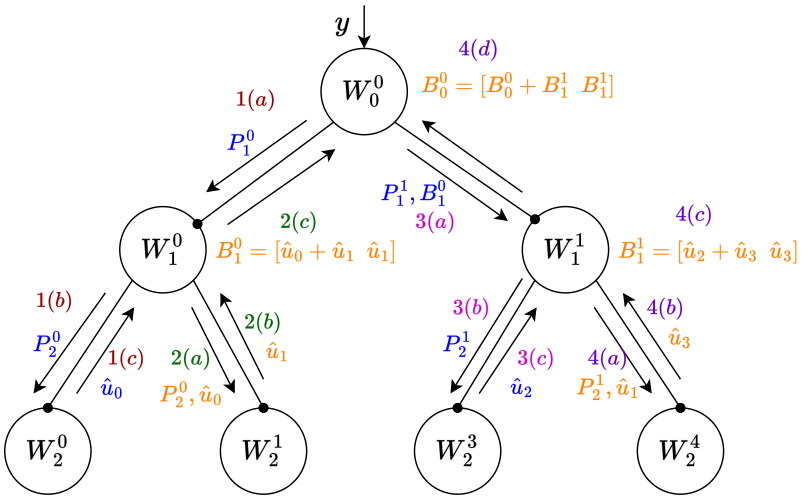

Figure 2.5: SC Decoder for N=4

3. In the third step, the R operation is applied for (W_0^0,W_1^1) followed by L operation for (W_1^1,W_2^2) to evaluate P_2^2. The hard decision for $\hat{u}_1$ is made at the leaf node as per the value P_2^2.

4. In step four, R operation is applied for (W_1^1,W_2^2), followed by the hard decision based on the value of P_2^3. The B operation on W_1^1 updates $B_1^1 = [\hat{u}_2 + \hat{u}_3 \; \hat{u}_3]$. Lastly B-operation on (W_0^0) gives the final codeword estimate, $\hat{\mathbf{c}}$=$[\hat{u}_0 + \hat{u}_1 + \hat{u}_2 + \hat{u}_3 \;\; \hat{u}_1 + \hat{u}_3 \;\; \hat{u}_2 + \hat{u}_3 \;\; \hat{u}_3]$.

Successive Cancellation List Decoder: In the SC decoder, the decision is made as per the LLR evaluated at that bit-channel. In list decoding, we assume both the paths for each u_ϕ, i.e, $u_\phi = 1$ and $u_\phi = 0$. The SCL decoder has another important parameter - the list size, L. The list size, L denotes the maximum number of paths that can be maintained in the decoder. When the current number of paths exceeds L, pruning is done to eliminate the less likely paths based on another parameter known as Path Metric (PM). The PM is defined for each path, and for the i^{th} path; it is updated as $PM_i = PM_i + LLR$ only if the decision is made as per the belief on that node. In the pruning process, the L paths with the highest PM are kept, and the rest are eliminated. From the selection of the final codeword from the list of L candidate codewords, the CRC check is done. The paths for which the CRC is invalid are eliminated. Among the paths that satisfy CRC, the path with the highest PM (most likely path) is chosen as the final codeword. Since we use CRC bits to select the final codeword, the decoder is called CRC Aided Successive Cancellation List (CA-SCL) decoder.

Chapter 3

Hardware Implementation

In this chapter, we discuss the hardware implementation of the polar coding algorithm discussed in Chapter 2. The implementation of the entire algorithm is divided into three separate IPs - 1) Polar Encoder 2) Rate Recover 3) Polar Decoder. The IPs have been developed by the Vivado High Level Synthesis (HLS) Tool. The hardware implementation and interface of each IP is discussed below.

3.1 Polar Encoder

Polar Encoder IP includes the complete transmitter end processing from CRC Encoder to Modulation as discussed in Section 2.2. The implementation of the Polar Encoder IP can be divided into the following sub-blocks - 1) CRC Encoder 2) Interleaver 3) Encoder 4) Rate Matching 5) Modulation

CRC Encoder -
In the CRC Encoder, the CRC bits are appended to the $1 \times A$ input message block, **blk**. Based upon the $crcLen$, we assign the encoding polynomial, **gPoly** of dimension $1 \times gLen$, where $gLen = crcLen + 1$. For example **gPoly=**[1 1 0 0 0 0 1] for $crcLen = 6$. CRC bits are the remainder of modulo 2 division of input, **blk** and **gPoly**. (3.1).

$$crcBits = blk\%gPoly \tag{3.1}$$

The $crcLen$ CRC bits are appended to the input message in CRC Encoding. The output of CRC Encoder, is $1 \times K$ vector, **blkCRC=**[**blk crcBits**]. Fig. 3.1 shows the hardware architecture for the initiliazation of **gPoly** as per the input $crcLen$. The polynomial values corresponding to each $crcLen$ is stored in the local memory, Block Random Access Memory (BRAM). As per the select signals generated via matching $crcLen$, **gPoly** is initialized. The control unit generates the appropriate read and write addresses to read/write into the memory. For the optimization purpose, the memories gPoly6, gPoly11, gPoly24 and gPoly has been partitioned completely i.e, data for each memory stored in registers. The partitioning enables the parallel initialization of all $crcLen$ polynomial values.

After the gPoly initialization (Init gPoly), the CRC bits are calculated. The hardware architecture for CRC bits calculation is shown in Fig. 3.2. In Fig. 3.2, all the memories are intialized in the C0 state which is executed only once. In the C1 state, the dividend is initialized with the remainder, followed by the check if division is possible or not in the C2 state. In the C3 state, if the division is possible then the upper block updates the remBits by modulo 2 division of divBlk with gPoly. The execution of $C1 \rightarrow C2 \rightarrow C3$ K times (iteration over the complete **blk**) completes

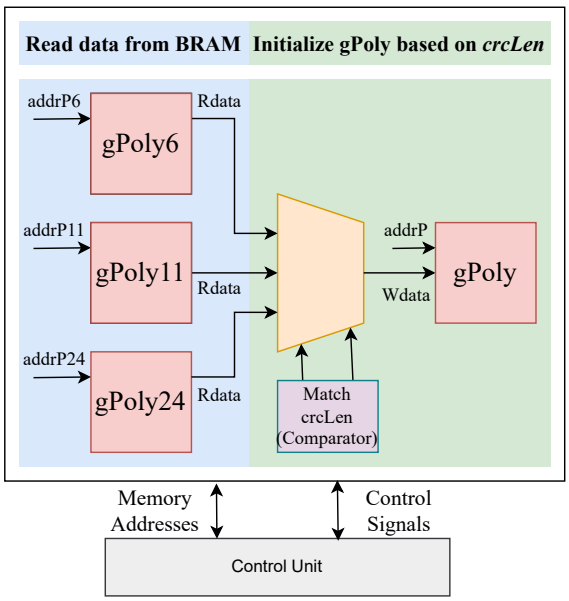

Figure 3.1: CRC Encoder Architecture: Init gPoly

the division and the $crcLen$ remBits are appended to the **blk** to get the CRC encoded message, **blkCRC**.

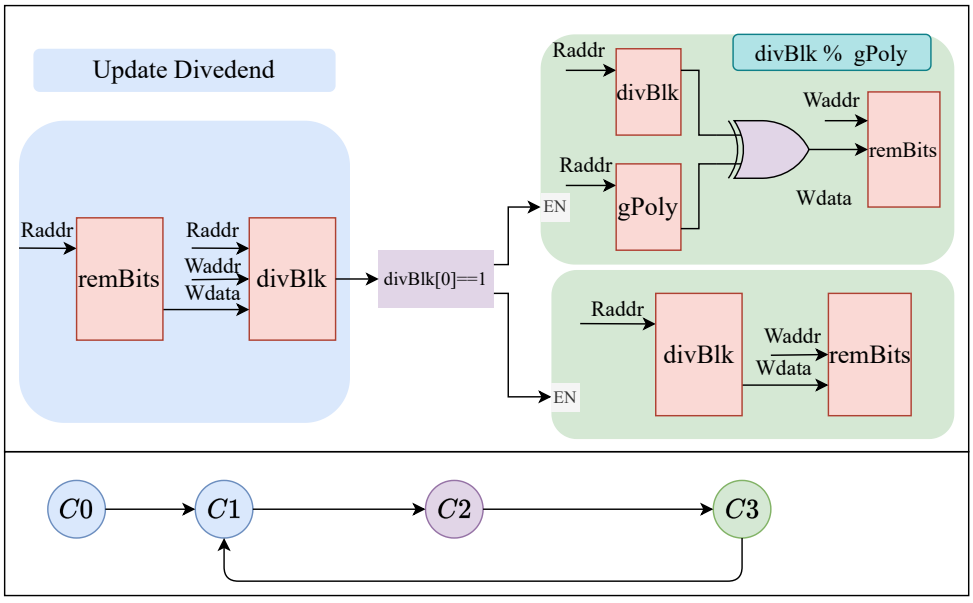

Figure 3.2: CRC Encoder Architecture: Calculate CRC bits

Interleaving - In interleaving, the $1 \times K$, **blkCRC** is rearranged according to the specified interleaving pattern. The interleaved output is denoted by $1 \times K$ vector, **blkCRCInt**. The hardware implementation of this involves read operation from **blkCRC** and write operation **blkCRCInt**.

Encoder - In encoder, first based upon the value of K and E, we set the PC Polar parameter, i.e, the number of parity check bits (nPC) and weighted parity check bits (nPCwm). The information bit vector, **u** of dimension $1 \times N$ is initialized as discussed in section 2.2.1. In **u**, the $N - K$ are information bit locations, initialized with **BlkCRCInt** and rest are frozen bit locations which are set to 0. The final codeword, **c** is obtained by matrix multiplication of $1 \times N$, **u** with $N \times N$, generator matrix, $\mathbf{G}_N$. For the hardware implementation, G_N is stored in local memory for the largest value of $N = 1024$. The generator matrix is partitioned across the rows and **u** is partitioned across the columns by a factor of 4 to parallelize $\frac{N}{4}$ multiplications to generate one value of **c**.

Rate Matching -
Rate Matching takes the $N \times 1$ codeword, **c** and transmit $1 \times E$ rate matched output, $\mathbf{c}_{RM}$. First, the input is sub-block interleaved as per the interleaving pattern. Then the bit selection is done to transmit E bits by bit repetition ($E > N$), shortening or puncturing ($E < N$) as discussed in 2.2.2, followed by bit interleaving where the input is stored column-wise and then row-wise. The hardware implementation of the rate-matching is just read and write operations from memory.

Modulation -
For modulation, we have provided the option to choose from four modulation schemes; BPSK, QPSK, QAM16, and QAM64 as listed in table 3.1. The parameter $modscheme$ decides the modulation scheme to be implemented. The hardware architecture of QPSK modulation scheme is shown in Fig. 3.3. The bits $[a_1 a_0]$ are modulated to one complex symbol $I + Qj$. Similar architecture applies for other modulation schemes where the constant, c and the bits per symbol (bps) differ.

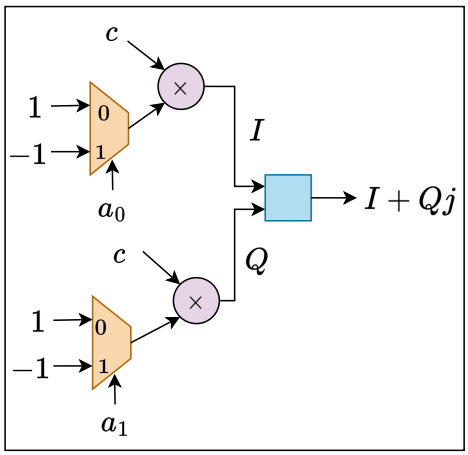

Figure 3.3: Hardware Implementation of QPSK Modulation

modscheme	Modulation Scheme	Bits-per-Symbol (bps)
0	BPSK	1
1	QPSK	2
2	QAM16	4
3	QAM64	6

Table 3.1: Modulation Scheme Options

3.2 Rate Recover

In the rate recover, first the soft demodulation is done to get $1 \times E$ LLR vector, **rSig**. Then the process is exact opposite of the rate matching. First, the **rSig** is sub-block deinterleaved as per the interleaving pattern. Then in bit selection, for $E >= N$, the first N values are taken whereas for $E < N$, puncturing($\frac{K}{E} \leq \frac{7}{16}$) or shortening is done to transmit the recovered N LLR values. Finally, the block is sub-block deinterleaved by storing LLR values, first row-wise then column-wise. The hardware implementation of each sub-block of rate recover IP is memory read operation as per some pattern and writing it to another memory.

3.3 Polar Decoder

The rate recovered LLR vector, **y** of length N from the rate-recover block is the input to decoder IP. Polar decoder uses the probabilistic estimates of encoded bits to retrieve the input message block. The Polar Decoder IP uses the CA-SCL [7] decoding algorithm discussed briefly in Section 2.3.2. The step-wise flow of CA-SCL decoding is shown in Fig. 3.4. We will discuss the details and hardware implementation of each block in Fig. 3.4 below.

Initialize Data Structures (Init DS) -
Since the polar decoding is done sequentially by iterating over each bit-channel. Also, decoding at ϕ^{th} bit-channel requires the estimates of $\phi - 1^{th}$ bit-channel hence LLRs and decisions have to stored for each bit-channel of every path separately. Table 3.2 lists the arrays, their dimensions and description. In Init DS, these arrays are initialized to default values when none of the path is active. All the data-structures are stored in the local memory, BRAM.

Assign Initial Path (AIP) and Initialization of P_0^0 -
At the start of the decoding process, initially all the paths are inactive. First, in AIP, initial path is assigned by setting that path as active and associating the corresponding arrays for that path. The hardware implementation of this is memory write operation for path initialization. After this, The $P_0^0(l)$ is initialized with the received values, **y** for the l^{th} active path.

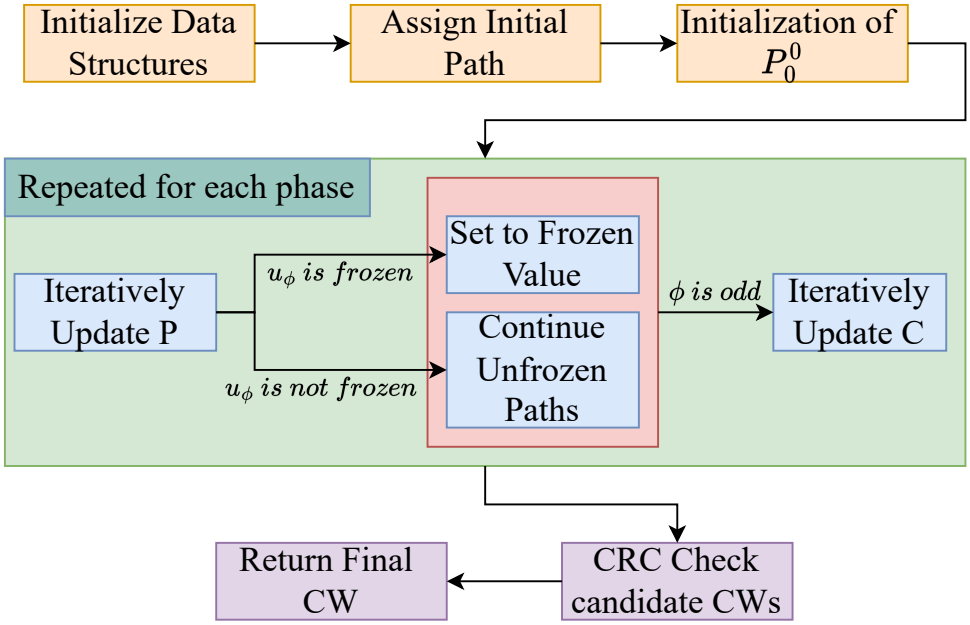

Figure 3.4: Step-wise Flow for List Decoding

Data Structure	Dimension	Description
inactivePathIndices	1 x L	Stores the path index that are inactive
activePath	1 x L	activePath[i]=1 if ith path is active.
llrPathMetric	1 x L	Stores the Path Metric for all the paths
ParrayPtrLLR (P)	(m+1) x L x N	Stores the LLRs for m^{th} layer, l^{th} path for n^{th} bit estimate.
ParrayPtrC (B)	(m+1) x L x N x 2	Stores the bit estimates for m^{th} layer, l^{th} path for n^{th} bit estimate (both 0 and 1 are taken).
pathIdxToArrayIdx	(m+1) x L	Mapping of pathIndex to arrayIndex
inactiveArrayIndices	(m+1) x L	Stores the inactive array indices for each path
inactiveArrayIndicesLen	1 x L	The no. of inactive array indices remaining for each path
arrayReferenceCount	(m+1) x L	arrayReferenceCount[i][j] Denotes the number of paths currently using the array pointed by ParrayPtrLLR[i][j]
savedCWs	N x L	arrayReferenceCount[i][j] Denotes the number of paths currently using the array pointed by ParrayPtrLLR[i][j]

Table 3.2: Data Structures for List Decoding

Iteratively Update P -
This step is repeated for each phase, ϕ from 0 to N. In this step, if the input phase is

ϕ the LLRs are updated for P_m^ϕ for all the layers and each active path. The computation of P_m^ϕ involves iteration over bit-channels, as discussed in section 2.3.2 via an example of SC decoder (Fig. 2.5). The computation at each bit-channel can be either minsum, $f(a,b)$ (3.2) or $g(a,b,\hat{u})$ (3.3). The architecture for f and g functions is shown in Fig. 3.5 and Fig. 3.6, respectively. After this if the u_ϕ is frozen location, then it is directly set to 0 else we move to the other block, continue path unfrozen.

$$f(a,b) = sgn(a) * sgn(b) * min(a,b) \tag{3.2}$$

$$g(a,b,\hat{u}) = (1-2\hat{u}) * a + b \tag{3.3}$$

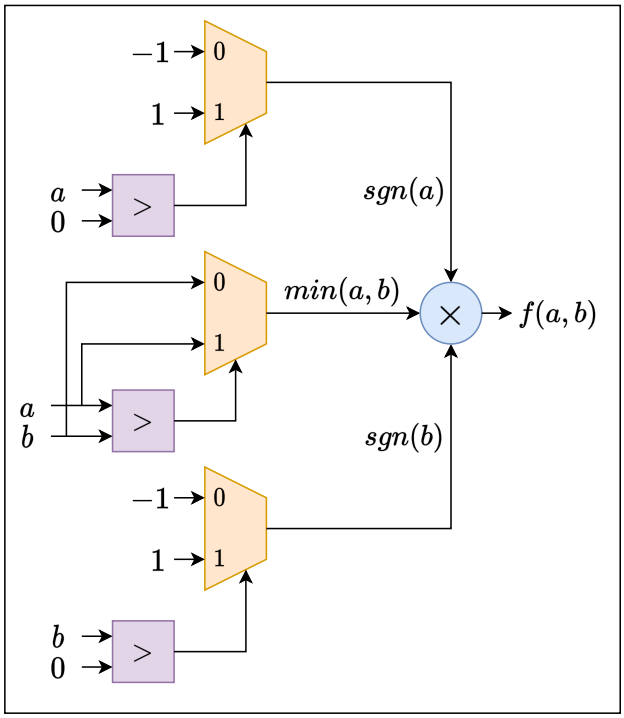

Figure 3.5: Hardware Implementation of f(a,b)

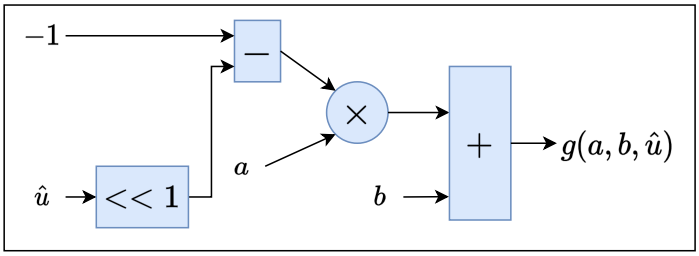

Figure 3.6: Hardware Implementation of g(a,b)

Continue Unfrozen Paths (CUP)
In CUP, for phase input, ϕ, we branch out to paths for the $\hat{u}_\phi$. If the current active paths exceeds the list size then decide on which paths to be continued by associating arrays for that paths. The step-wise implementation of the same as follows -

1. For the current phase, ϕ, we branch out to both the possible estimates, i.e, $u_\phi = 0$ and $u_\phi = 1$. The probabilistic estimate of each branch of the path is stored in **probF** array of dimension $L \times 2$. Fig. 3.7 shows the path splitting for $\hat{u}^0$ and $\hat{u}^1$. Each node in this graph represents an SC decoder itself.

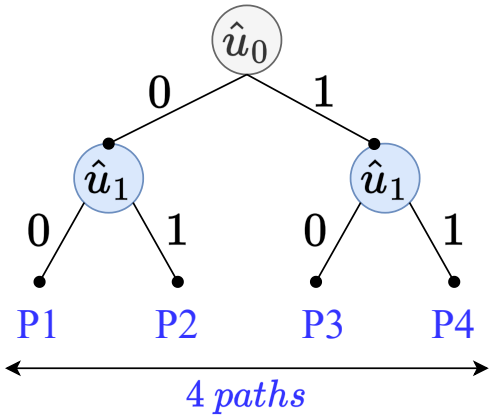

Figure 3.7: SCL Decoding Overview

2. For the probabilistic estimate of the l^{th} path, we check LLR value, $P_m^\phi(l)$. If LLR is > 0 then believe is as per the $u^\phi = 0$ path and the **probF** is initialized with the updated Path Metric (PM). Fig. 3.8 shows the architecture to update the **probF**. Here to parallelize the operations, **probF** is partitioned along the column and split into two separate memories, **probF0** and **probF1**.

3. If the number of current active paths (L_c) is greater than the list size, L, then pruning is done to eliminate the less likely paths. Fig. 3.9 shows the case when current number of paths, 8 exceeds the list size of 4, pruning is done to eliminate 4 paths. Pruning is done as per the updated PM. Only the paths with the highest PM are continued. The hardware implementation of this involves sorting of **probF** array and storing the indices of the path of be continued.

4. Lastly, continue to the relevant path by associating corresponding arrays.

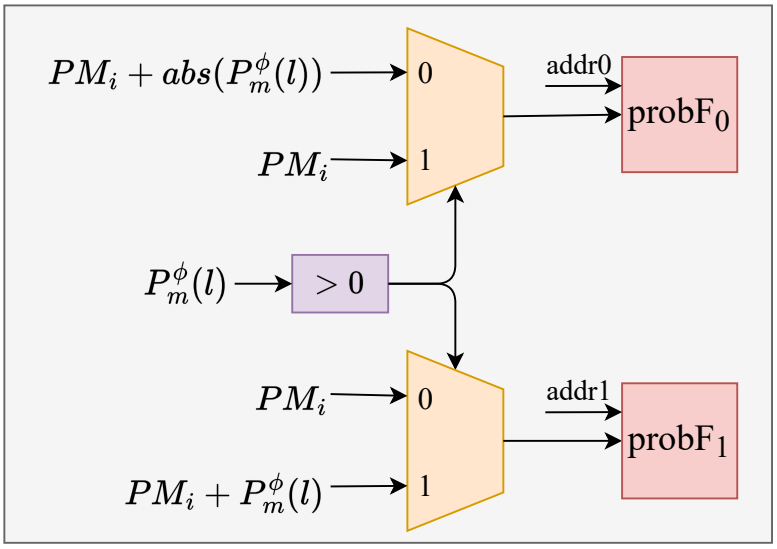

Figure 3.8: Architecture for probF Update

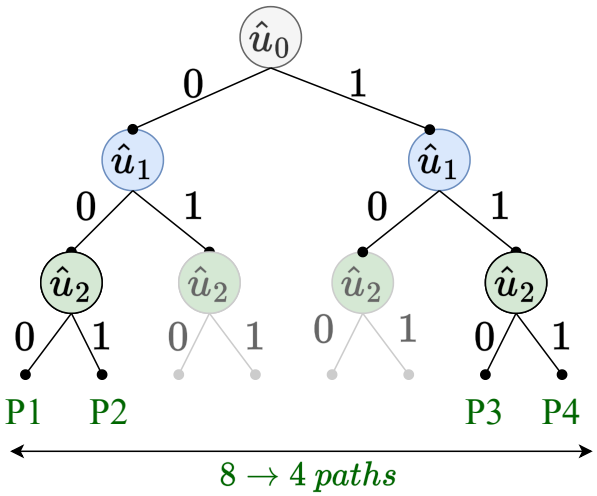

Figure 3.9: Pruning in SCL Decoder

Iteratively Update C -
At this point, we have atmost L active paths. Next, if ϕ is odd, the estimates(B_λ^ϕ) are by the B operation discussed in section 2.3.2 (Fig. 2.4). The architecture for this is simply xor operation between the given estimates.

CRC Check for Candidate Codewords (CWs)
From the L candidate Cws, CRC Check is done and the path that donot satisfy CRC are completely eliminated. From the paths that satisy CRC, the most likely path is chosen based upon the PM. For CRC Check, the starting K bit message is extracted from the N bit codeword. This K bit message is encoded via CRC Encoder. The CWs for each the calculated CRC matches the actual CRC, passes the CRC check. The hardware implementation involves CRC Encoder block (discussed in

section 3.1) and comparator to compare the CRC bits.
The resource utilization and power for the proposed hardware for the three IPs is shown is listed in table 3.3. These results are for the single precision floating point (SP-FP) implementation. Polar Decoder has the highest hardware complexity followed by Polar Encoder and Rate Recover.

IP	BRAM	DSP	LUT	FF	Power (in Watts)
Polar Encoder	82.5	98	19364	16652	0.529
Rate Recover	108	154	14178	12870	0.255
Polar Decoder	340.5	222	35992	26495	0.624

Table 3.3: Resource Utilization of IPs for FP-SP

3.4 Implementation On SoC

The proposed architecture for Polar Encoder, Rate Recover, and Polar Decoder on SoC is shown in Fig. 3.10. The Zynq SoC consists of the Arm Processing System (PS) and the Programmable Logic (PL). The data to be transmitted is stored in the Double Data Rate (DDR) memory. The data from the DDR memory is passed to the AXI Direct Memory Access (DMA) in the PL via the AXI Accelerated Coherency Port (ACP). The data from the AXI DMA is sent Polar Encoder block. The output from the Polar Encoder Block is sent over the AWGN Channel in PS via the AXI DMA. The output from AWGN is sent to the Rate Recover IP. The output from the Rate Recover IP is passed to the Polar Decoder IP via the AXI Stream protocol. The pre-processed data from this block is stored back into the DDR memory from AXI DMA via the AXI ACP port. Each of the individual IPs is tested separately on the Zynq SoC.

The execution time of the hardware IPs has been compared with its ARM-based software implementation. The acceleration factor is the ratio of hardware execution time over the software execution time. For the results in table 3.4, 3.5, and 3.6 the modulation scheme is QPSK. In table 3.4, the acceleration factor for the Polar Encoder IP is higher for large values of K and E.

K	E	crcLen	Acceleration Factor
1018	2048	11	1.343
307	360	24	0.815
54	124	24	0.927
20	124	11	0.858

Table 3.4: Acceleration Factor for Polar Encoder IP

In table 3.5, for Rate Recover IP, the value of $crcLen = 24$ and we get a good acceleration ~ 2.5. Also, it increases with the value of K and E. In table 3.6, the

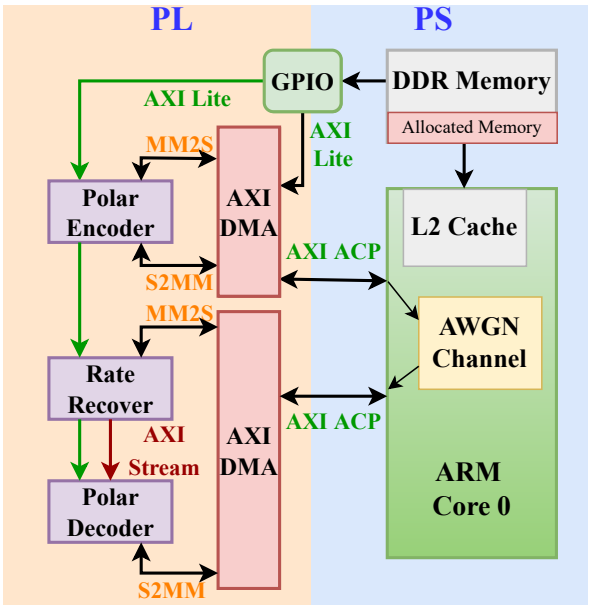

Figure 3.10: Proposed Architecture for Polar Codes on SoC

K	E	Acceleration Factor
56	864	2.489
56	124	2.154
67	128	1.918
307	360	2.172

Table 3.5: Acceleration Factor for Rate Recover IP

K	N	crcLen	Acceleration Factor
256	164	24	0.72
1024	1011	11	0.627
512	128	24	0.568
43	256	24	0.499

Table 3.6: Acceleration Factor for Polar Decoder IP

acceleration factor is higher for the cases where the value K is comparable to N. The throughput of polar encoder, rate recover and polar decoder IPs is $1.28, 271$ and $1\ Mbps$. The acceleration factor for Polar Encode and Polar Decoder IP can be increased by reducing the data communication overhead from DMA to IP and vice versa and paralleling the operations within the IP. Optimising the IPs to get an acceleration with respect to software is an important research direction that we would like to explore in the thesis in future.

3.5 IP Documentation

3.5.1 Polar Encoder IP

The Polar Encoder IP encodes the $K - crcLen$ length message block to a $\frac{2*E}{bps}$ length rate matched modulated output. The IP has an AXI Stream Interface. The Port Description of the IP is listed in table 3.10. The AXI4-Lite parameter bus is used to assign the control information and the parameter details. The parameters of the IP assigned via the AXI4-Lite port is listed in table 3.7.

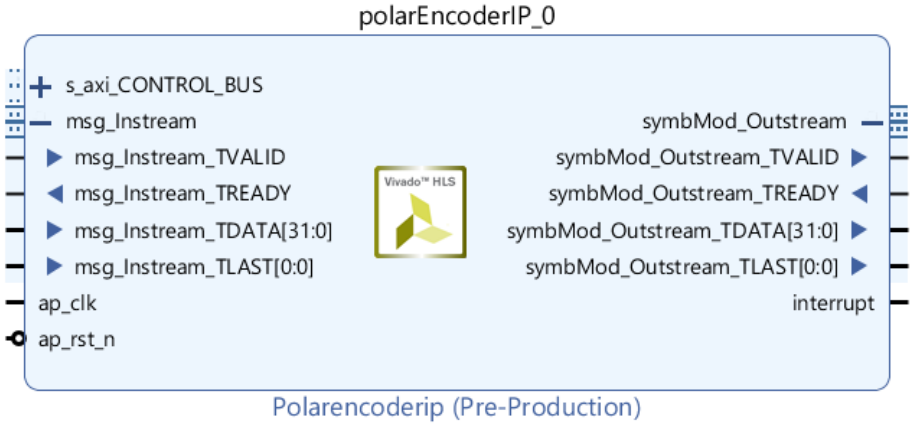

Figure 3.11: Interface of Polar Encoder IP

Field	Bits	Description
msgLen	10	Length of input message block
E	14	Rate matched length
crcLen	4	Number of CRC bits
nMax	4	9 for DCI and 10 for UCI
iIl	1	1 for DCI and 0 for UCI
iBIL	1	0 for DCI and 1 for UCI
mod_scheme	2	Choose from BPSK (0), QPSK (1), QAM16 (2), QAM64 (3)

Table 3.7: Parameters Description for Encoder IP

3.5.2 Rate Recover IP

The Rate Recover IP takes the $\frac{2*E}{bps}$ received values, performs demodulation and rate recovering and outputs E LLR values. The port and parameter description of the IP is listed in table 3.8 and 3.11.

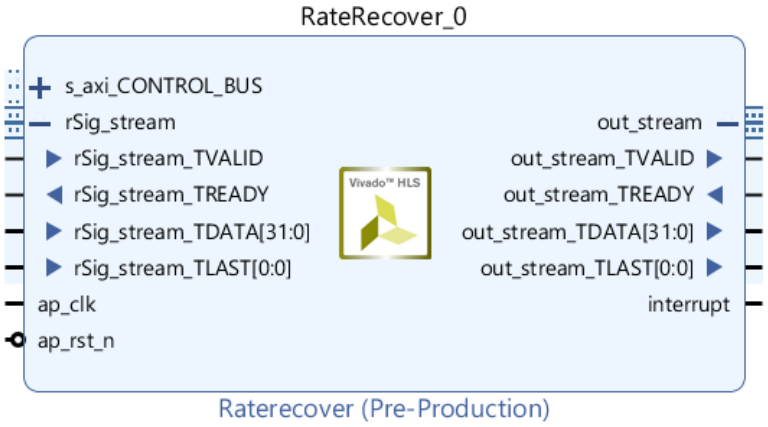

Figure 3.12: Interface of Rate Recover IP

Field	Bits	Description
K	10	Length of input message block
E	14	Rate matched length
N	11	Length of Codeword
iBIL	1	0 for DCI and 1 for UCI
mod_scheme	2	Choose from BPSK (0), QPSK (1), QAM16 (2), QAM64 (3)

Table 3.8: Parameter Description of Rate Recover IP

3.5.3 Polar Decoder IP

The Polar Decoder IP performs the CRC Aided Successive Cancellation List (CA-SCL) decoding for the maximum list size of 4. The input to the IP is N LLR values and the output is K length recovered CRC encoded message. The first $K - crcLen$ bits of the output is the recovered message block. The port and parameter description of the IP is listed in table 3.9 and 3.12.

Field	Bits	Description
K	10	Length of input message block
E	14	Rate matched length
N	11	Length of Codeword
L	4	List Size
crcLen	4	Number of CRC bits
nMax	4	9 for DCI and 10 for UCI
iIl	1	1 for DCI and 0 for UCI

Table 3.9: Parameter Description of Polar Decoder IP

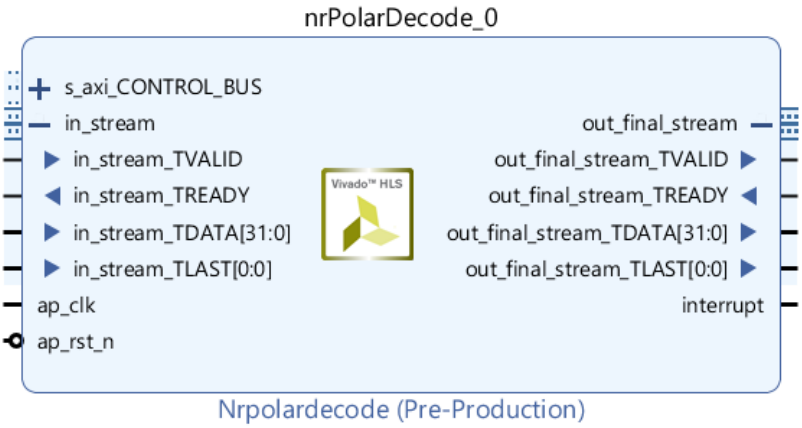

Figure 3.13: Interface of Polar Decoder IP

Signal	Direction	Width	Description
ap_clk	Input	1	Clock Signal for the IP Core
ap_rst_n	Input	1	Synchronous Reset with positive polarity
interrupt	Output	1	Indicates error conditions
s_axi_CONTROL_BUS_AWADDR	Input	7	Parameter Bus - AXI4-Lite Interface
s_axi_CONTROL_BUS_AWVALID	Input	1	
s_axi_CONTROL_BUS_AWREADY	Output	1	
s_axi_CONTROL_BUS_WDATA	Input	32	
s_axi_CONTROL_BUS_WSTRB	Input	1	
s_axi_CONTROL_BUS_WVALID	Input	1	
s_axi_CONTROL_BUS_WREADY	Output	1	
s_axi_CONTROL_BUS_BRESP	Output	2	
s_axi_CONTROL_BUS_BVALID	Output	1	
s_axi_CONTROL_BUS_BREADY	Input	1	
s_axi_CONTROL_BUS_ARADDR	Input	7	
s_axi_CONTROL_BUS_ARVALID	Input	1	
s_axi_CONTROL_BUS_ARREADY	Output	1	
s_axi_CONTROL_BUS_RDATA	Output	32	
s_axi_CONTROL_BUS_RRESP	Output	1	
s_axi_CONTROL_BUS_RVALID	Output	1	
s_axi_CONTROL_BUS_RREADY	Input	1	
msg_Instream_TVALID	Input	1	Data Input Bus - AXI4-Stream Slave Interface
msg_Instream_TREADY	Output	1	
msg_Instream_TDATA	Input	32	
msg_Instream_TLAST	Input	1	
symbMod_Outstream_TVALID	Output	1	Data Output Bus - AXI4-Stream Master Interface
symbMod_Outstream_TREADY	Input	1	
symbMod_Outstream_TDATA	Output	32	
symbMod_Outstream_TLAST	Output	1	

Table 3.10: Port Description Of Polar Encoder IP

Signal	Direction	Width	Description
ap_clk	Input	1	Clock Signal for the IP Core
ap_rst_n	Input	1	Synchronous Reset with positive polarity
interrupt	Output	1	Indicates error conditions
s_axi_CONTROL_BUS_AWADDR	Input	7	Parameter Bus - AXI4-Lite Interface
s_axi_CONTROL_BUS_AWVALID	Input	1	
s_axi_CONTROL_BUS_AWREADY	Output	1	
s_axi_CONTROL_BUS_WDATA	Input	32	
s_axi_CONTROL_BUS_WSTRB	Input	1	
s_axi_CONTROL_BUS_WVALID	Input	1	
s_axi_CONTROL_BUS_WREADY	Output	1	
s_axi_CONTROL_BUS_BRESP	Output	2	
s_axi_CONTROL_BUS_BVALID	Output	1	
s_axi_CONTROL_BUS_BREADY	Input	1	
s_axi_CONTROL_BUS_ARADDR	Input	7	
s_axi_CONTROL_BUS_ARVALID	Input	1	
s_axi_CONTROL_BUS_ARREADY	Output	1	
s_axi_CONTROL_BUS_RDATA	Output	32	
s_axi_CONTROL_BUS_RRESP	Output	1	
s_axi_CONTROL_BUS_RVALID	Output	1	
s_axi_CONTROL_BUS_RREADY	Input	1	
rSig_stream_TVALID	Input	1	Data Input Bus - AXI4-Stream Slave Interface
rSig_stream_TREADY	Output	1	
rSig_stream_TDATA	Input	32	
rSig_stream_TLAST	Input	1	
out_stream_TVALID	Output	1	Data Output Bus - AXI4-Stream Master Interface
out_stream_TREADY	Input	1	
out_stream_TDATA	Output	32	
out_stream_TLAST	Output	1	

Table 3.11: Port Description Of Rate Recover IP

Signal	Direction	Width	Description
ap_clk	Input	1	Clock Signal for the IP Core
ap_rst_n	Input	1	Synchronous Reset with positive polarity
interrupt	Output	1	Indicates error conditions
s_axi_CONTROL_BUS_AWADDR	Input	7	Parameter Bus - AXI4-Lite Interface
s_axi_CONTROL_BUS_AWVALID	Input	1	
s_axi_CONTROL_BUS_AWREADY	Output	1	
s_axi_CONTROL_BUS_WDATA	Input	32	
s_axi_CONTROL_BUS_WSTRB	Input	1	
s_axi_CONTROL_BUS_WVALID	Input	1	
s_axi_CONTROL_BUS_WREADY	Output	1	
s_axi_CONTROL_BUS_BRESP	Output	2	
s_axi_CONTROL_BUS_BVALID	Output	1	
s_axi_CONTROL_BUS_BREADY	Input	1	
s_axi_CONTROL_BUS_ARADDR	Input	7	
s_axi_CONTROL_BUS_ARVALID	Input	1	
s_axi_CONTROL_BUS_ARREADY	Output	1	
s_axi_CONTROL_BUS_RDATA	Output	32	
s_axi_CONTROL_BUS_RRESP	Output	1	
s_axi_CONTROL_BUS_RVALID	Output	1	
s_axi_CONTROL_BUS_RREADY	Input	1	
in_stream_TVALID	Input	1	Data Input Bus - AXI4-Stream Slave Interface
in_stream_TREADY	Output	1	
in_stream_TDATA	Input	32	
in_stream_TLAST	Input	1	
out_final_stream_TVALID	Output	1	Data Output Bus - AXI4-Stream Master Interface
out_final_stream_TREADY	Input	1	
out_final_stream_TDATA	Output	32	
out_final_stream_TLAST	Output	1	

Table 3.12: Port Description Of Polar Decoder IP

Chapter 4

Hardware Results and Analysis

This chapter presents the Block Error Rate (BLER) results of the proposed architecture for varying SNR, rates and word lengths. For the validation of the proposed architecture, the results are compared with the double-precision floating point (DP-FP) implementation of Polar Encoder and Polar Decoder provided in the MATLAB 5G Toolbox.

4.1 BLER Vs SNR

The BLER is calculated by 4.1, where N_{blks} are the total number of blocks for which the design is simulated and N_{err} are the number of blocks for which the recovered block doesn't match with the original block. Fig. 4.1 and Fig. 4.2 shows the variation of BLER at varying SNR for different rate (K/E) values for downlink and uplink, respectively. These results are computed by simulating for a total of 10000 blocks ($N_{blks} = 10000$).

$$BLER = \frac{N_{err}}{N_{blks}} \tag{4.1}$$

Fig. 4.1 is the plot for DCI where the $crcLen = 24$ and modulation scheme is QPSK. In Fig. 4.1, we can see that the BLER for each rate values decreases with increasing SNR. Moreover, the BLER at a particular SNR value increases with increasing rate. The BLER ~ 1 at lower SNR values and ~ 0 at higher SNR values and the similar trend is followed for all rate values. Moreover, the BLER at a particular SNR value increases with increasing rate. Similarly, Fig. 4.2 are the BLER results obtained for UCI where the $crcLen = 11$ and modulation scheme is QPSK..

4.2 Word Length Analysis

In the Section 3, we discussed the Single-Precision Floating point (SP-FP), i.e, 32-bit floating point implementation of the polar codes. Although, the accuracy for SP-FP matches exactly with DP-FP MATLAB implementation, the memory utilization and hardware complexity of operation can be huge. In this section, we present the detailed word length (WL) analysis through which we can decide on the optimal word length for the design at comparable accuracy.

4.2.1 Integer and Fractional Part Requirement

The notation (W, I) presents fixed point data-type of W bits where I is the number of integer bits and $W - I$ are the number of fractional bits. Since in the Polar En-

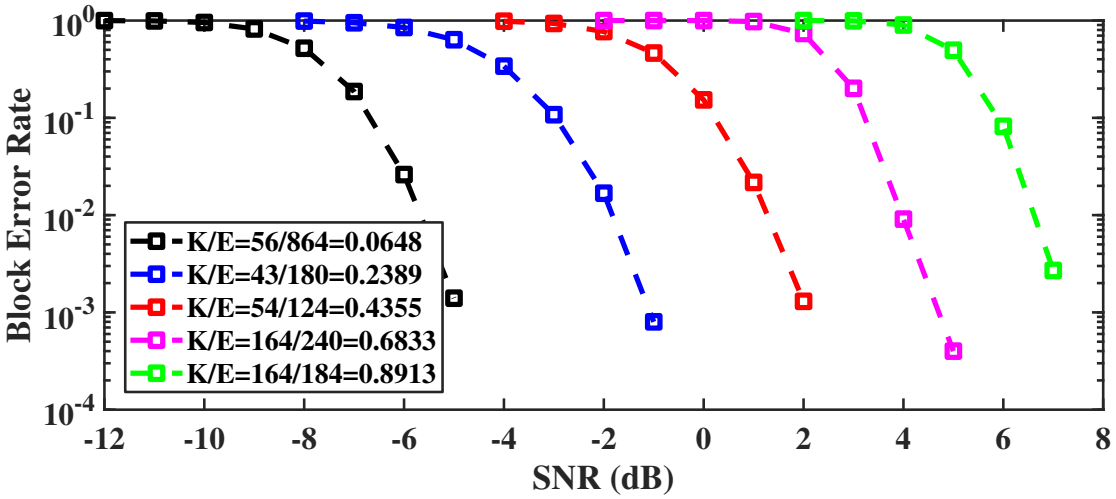

Figure 4.1: BLER Vs SNR at different Rates for Downlink

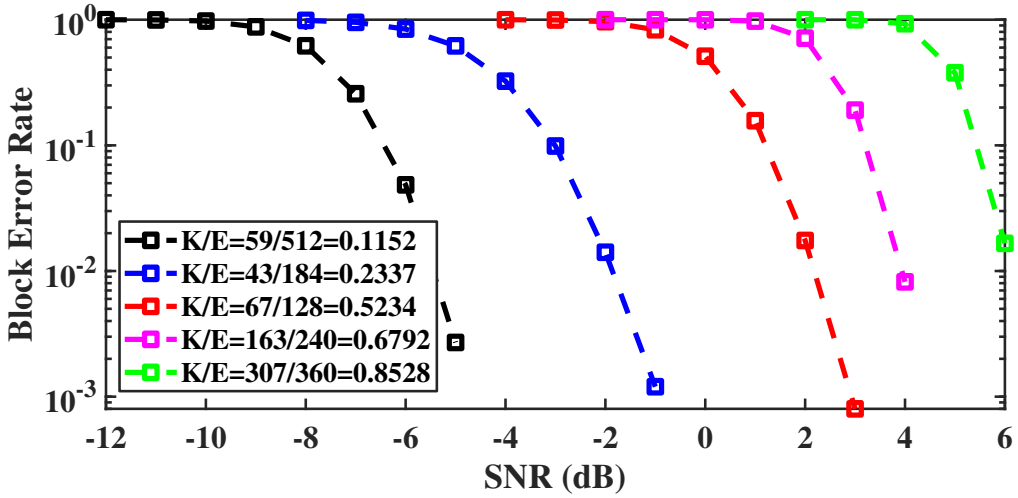

Figure 4.2: BLER Vs SNR at different Rates for Uplink

coder IP, the computation is done on 1-bit integer values, the word length analysis is done for the Rate-Recover and Polar Decoder IPs and the results for the same are discussed below.

Integer Bit Requirement: For the estimation of the minimum integer bit requirement,the number of fractional bits $(W - I)$ is kept constant to 12 and the number of integer bits are varied to 7, 8, 9. In table 4.1, since the BLER for $I < 8$ is ~ 1, the minimum integer bit requirement for the polar codes is 8. Appropriate scaling of LLRs was done in the Rate Recover block to reduce the integer bit requirement.

Fractional Bit Requirement: Similarly, for the estimation of fractional bit requirement, the number of integer bits is kept constant$(= 8)$ and the number of fractional bits are varied. Fig. 4.3 and Fig. 4.4 shows the variation of BLER with varying number of fractional bits for DCI and UCI, respectively. These results are taken

WL	DCI	UCI
SP-FP	0.7709	0.0965
(21,9)	0.7709	0.0965
(20,8)	0.7709	0.0965
(19,7)	0.9554	1

Table 4.1: BLER for varying Integer bits

at $SNR = 0\ dB$ and rate of 0.4355 for DCI and 0.5234 for UCI. The modulation scheme for these results is QPSK. The plot of BLER deviates from the SP-FP for $F < 6$. Moreover, since the the BLER for $F \geq 6$ remains almost the same, we need not go for higher fractional bits.

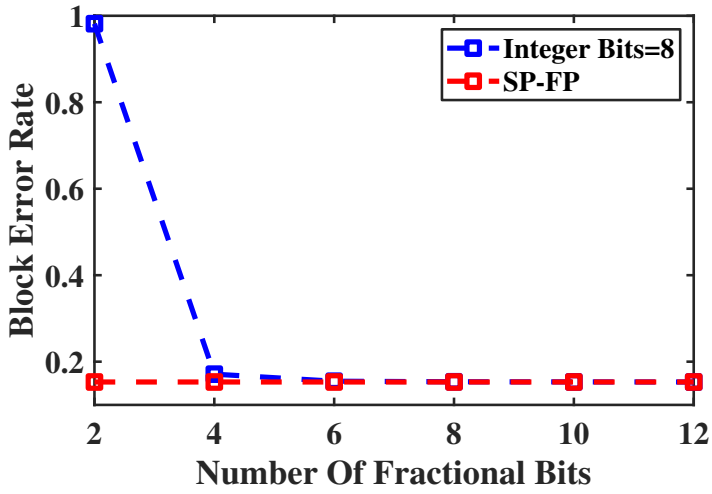

Figure 4.3: BLER Vs Number of Fractional Bits for Downlink

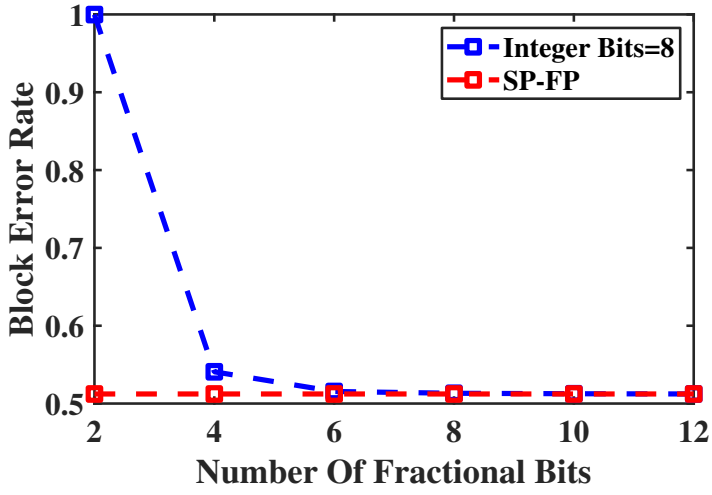

Figure 4.4: BLER Vs Number of Fractional Bits for Uplink

From the presented analysis, the minimum integer and fractional bit requirement are 8 and 6, respectively. Additionally, the variation of BLER with varying SNR is shown

in Fig. 4.5 and 4.6 for DCI and UCI, respectively. The BLER for WLs $(14, 8)$, $16, 8$ and SP-FP decreases with increasing SNR, while for $(10, 8)$, the BLER is remains almost constant (~ 1). Hence, $(14, 4)$ is an appropriate choice of word length for the proposed design.

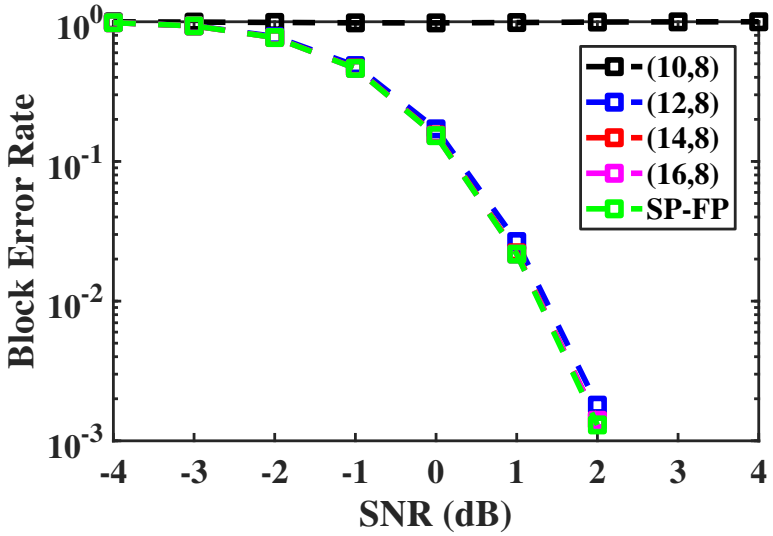

Figure 4.5: BLER Vs SNR at different WL for Downlink

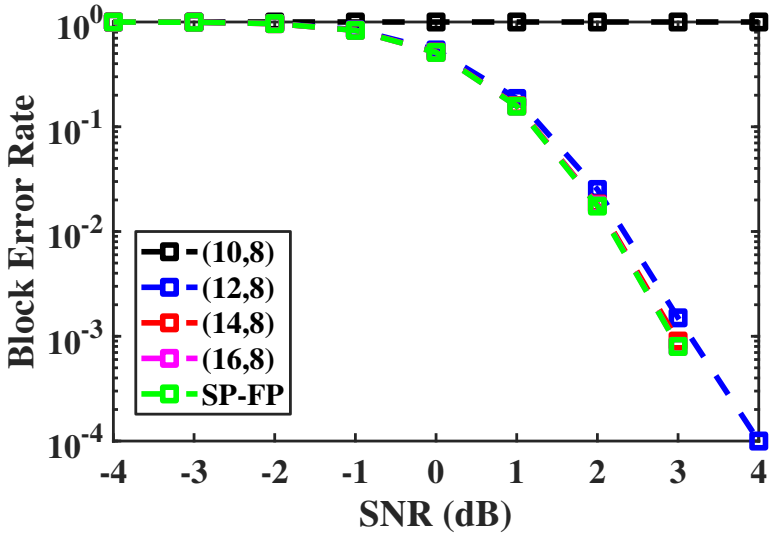

Figure 4.6: BLER Vs SNR at different WL for Uplink

4.2.2 Resource Utilization

In Section 4.2.1, we discussed the accuracy of the design at different word lengths. In this section, the hardware resource utilization for different WLs is discussed. Table 4.2 and 4.3 shows the comparison of hardware complexity at three different word lengths for the Rate-Recover and Decoder IP, respectively. The BRAM and DSP utilization decrease with decreasing WL as the memory requirement is much less, and the operations are performed on a smaller data width. The LUT utilization for Rate Recover IP is higher at lower WLs because LUTs are used as memory instead of BRAM. With the word length analysis, we can conclude that resource utilization and power consumption decreases significantly with a decrease in WL at comparable accuracy. As per the constraints of hardware and accuracy, one can opt for the required word length.

WL	**BRAM**	**DSP**	**FF**	**LUT**	**Dynamic Power (in W)**
SP-FP	108	154	13278	12639	0.255
(14,8)	52	145	12870	14178	0.181
(12,8)	47.5	145	12718	13721	0.181

Table 4.2: Rate Recover: Complexity Comparison for different WL

WL	**BRAM**	**DSP**	**FF**	**LUT**	**Dynamic Power (in W)**
SP-FP	340.5	222	26495	35992	0.624
(14,8)	285.5	216	23295	33025	0.615
(12,8)	279.5	216	23239	33511	0.608

Table 4.3: Polar Decoder: Complexity Comparison for different WL

Chapter 5

Conclusion and Future Scope

Polar Codes are adopted for the channel coding of control channels for eMBB in 5G NR by the 3GPP standardization group. The thesis presents an overview of the blocks of polar coding for 5G NR for uplink and downlink control channels. The end-to-end hardware architecture of polar codes is implemented via three IPs - Polar Encoder, Rate Recover, and Polar Decoder. The block error rate of the proposed architecture is analyzed for varying SNR at different rate values. Moreover, the complexity of the proposed architecture is presented for different word lengths. The presented analysis showed the reduced hardware complexity at comparable accuracy with the fixed-point implementation. The proposed architecture has been integrated with the Zynq Processing system and validated on the Zynq system-on-chip.

In the future, we will be working on optimizing the hardware to achieve a good acceleration. Also, we will explore the hardware-software co-designing to exploit the properties of both hardware and software in the system. In hardware-software co-design, we can move the extensive hardware part of the algorithm to software by careful analysis of each hardware block. With hardware-software co-designing, overall, a low latency system can be developed. Also, we will explore on the possibility of dynamic partial reconfiguration in the current design to further improve the resource and power utilization.

References

[1] Last accessed - June 2022. AMD-Xilinx IP for Polar Encoder and Decoder-https://www.mouser.in/ProductDetail/Xilinx/EF-DI-POLAR-ENC-DEC-SITE?qs=Zz7%252BYVVL6bFAkP7PqmYCYQ%3D%3D.

[2] C. Berrou, A. Glavieux, and P. Thitimajshima, "Near shannon limit error-correcting coding and decoding: Turbo-codes. 1," in *Proceedings of ICC '93 - IEEE International Conference on Communications*, vol. 2, pp. 1064–1070 vol.2, 1993.

[3] CodesDavid, C. J., and MacKayCavendish, "Near shannon limit performance of low density parity check codes," 1996.

[4] X. Lin, J. Li, R. Baldemair, J.-F. T. Cheng, S. Parkvall, D. C. Larsson, H. Koorapaty, M. Frenne, S. Falahati, A. Grovlen, and K. Werner, "5g new radio: Unveiling the essentials of the next generation wireless access technology," *IEEE Communications Standards Magazine*, vol. 3, no. 3, pp. 30–37, 2019.

[5] Z. Shen, A. Papasakellariou, J. I. Montojo, D. Gerstenberger, and F. Xu, "Overview of 3gpp lte-advanced carrier aggregation for 4g wireless communications," *IEEE Communications Magazine*, vol. 50, 2012.

[6] E. Arikan, "Channel polarization: A method for constructing capacity-achieving codes for symmetric binary-input memoryless channels," *IEEE Transactions on Information Theory*, vol. 55, no. 7, pp. 3051–3073, 2009.

[7] I. Tal and A. Vardy, "List Decoding of Polar Codes," *IEEE Transactions on Information Theory*, vol. 61, pp. 2213–2226, May 2015.

[8] K. Niu and K. Chen, "CRC-Aided Decoding of Polar Codes," *IEEE Communications Letters*, vol. 16, pp. 1668–1671, Oct. 2012.

[9] C. Zhang, B. Yuan, and K. K. Parhi, "Reduced-latency SC polar decoder architectures," in *2012 IEEE International Conference on Communications (ICC)*, (Ottawa, ON, Canada), pp. 3471–3475, IEEE, June 2012.

[10] C. Zhang and K. K. Parhi, "Low-Latency Sequential and Overlapped Architectures for Successive Cancellation Polar Decoder," *IEEE Transactions on Signal Processing*, vol. 61, pp. 2429–2441, May 2013.

[11] C. Leroux, A. J. Raymond, G. Sarkis, and W. J. Gross, "A Semi-Parallel Successive-Cancellation Decoder for Polar Codes," *IEEE Transactions on Signal Processing*, vol. 61, pp. 289–299, Jan. 2013.

[12] A. J. Raymond and W. J. Gross, "Scalable successive-cancellation hardware decoder for polar codes," in *2013 IEEE Global Conference on Signal and Information Processing*, (Austin, TX, USA), pp. 1282–1285, IEEE, Dec. 2013.

[13] P. Giard, A. Balatsoukas-Stimming, T. C. Muller, A. Burg, C. Thibeault, and W. J. Gross, "A multi-Gbps unrolled hardware list decoder for a systematic polar code," in *2016 50th Asilomar Conference on Signals, Systems and Computers*, (Pacific Grove, CA, USA), pp. 1194–1198, IEEE, Nov. 2016.

[14] A. Pamuk, "An FPGA implementation architecture for decoding of polar codes," in *2011 8th International Symposium on Wireless Communication Systems*, (Aachen, Germany), pp. 437–441, IEEE, Nov. 2011.

[15] G. Berhault, C. Leroux, C. Jego, and D. Dallet, "Hardware implementation of a soft cancellation decoder for polar codes," in *2015 Conference on Design and Architectures for Signal and Image Processing (DASIP)*, (Krakow, Poland), pp. 1–8, IEEE, Sept. 2015.

[16] A. Balatsoukas-Stimming, A. J. Raymond, W. J. Gross, and A. Burg, "Hardware Architecture for List Successive Cancellation Decoding of Polar Codes," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 61, pp. 609–613, Aug. 2014.

[17] J. Tong, H. Zhang, L. Huang, X. Liu, and J. Wang, "An Asymmetric Adaptive SCL Decoder Hardware for Ultra-Low-Error-Rate Polar Codes," in *2019 16th International Symposium on Wireless Communication Systems (ISWCS)*, (Oulu, Finland), pp. 532–536, IEEE, Aug. 2019.