# FPGA Implementation and Architecture Design of Polar Encoder for 5G-NR

Harsha Rudramuniyappa†, Samudrala Soujanya†, Rochak Jain‡, Naveed Anjum‡,
Prem Singh† and Ekant Sharma‡
† Department of ECE, IIIT Bangalore, India
‡ Department of ECE, IIT Roorkee, India

***Abstract*—This work focuses on the FPGA implementation of the 3GPP complaint polar encoder for 5G-new radio (NR). We use Xilinx's polar intellectual property (IP) aka polar IP for the polar encoding functionality. We design a polar configuration IP by generating the bit allocation (BA) table, configuring register values and passing the right control parameters. Next, a parameter handler is developed which is a memory mapped module and provides an advanced extensible interface (AXI)-lite interface to the polar IP. To test the functionality of the polar IP, a Verilog test bench is developed. It provides configuration parameters, payload and BA table in compliance with 3GPP standards to the polar configuration IP. We verify the output of our implementation with the Matlab 5G-NR built-in library on the Xilinx evaluation board. Our results show the efficiency of the implementation in terms of latency and resource utilization on the evaluation board.**

## I. Introduction

The third generation partnership project (3GPP) created standards for the new radio (NR) air interface for the fifth generation (5G) mobile communications technology [1]. 5G-NR lays the foundation for future wireless technologies to deliver more faster, reliable and secure communication that makes many things possible that long term evolution (LTE) could-not. As a result of new scenarios, use cases, requirements and frequencies, the deployment of 5G, especially the physical layer is significantly challenging and different from its 4G counterpart. In order to enable several use cases like enhanced mobile broadband (eMBB), ultra-reliable low-latency communications (URLLC) and massive machine type communications (mMTC), 5G is intended to meet a variety of requirements from a wide range of business sectors. This demands 5G to use a very broad frequency range from below 1 GHz to above 50 GHz [2]. Numerous new challenges have popped up as a result of new scenarios, use cases, requirements and frequencies. In particular, the implementation of the physical layer is very challenging because it requires high computational power. To meet these computational requirements, FPGA is an ideal choice for the physical layer implementation in 5G.

Many different types of data need to be transferred into a cellular network. User data needs to be transferred, but so does control information to manage the radio communication link, as well as data to provide synchronisation, access and reliability. The reliability of a radio communication link depends on the design of control channels, namely the physical downlink/uplink control channel (PDCCH/PUCCH). PDCCH carries the downlink control information (DCI), the decoding of which at the user device gives scheduling information [2]. Additionally, it is employed to provide information about termination of UL transmissions, slot formats and power-control orders. The design of 5G-NR PDCCH involves developing various physical layer functionalities in compliance with the 3GPP standards. In practice, a wireless channel is noisy which introduces errors in the transmitted signal. To combat this effect, error control coding is used in all practical wireless systems. For example, 5G-NR uses low density parity check (LDPC) and polar error control coding for data and control channels, respectively. Polar encoding is one of the critical functional blocks in the PDCCH chain which detects and corrects the error over an unreliable channel. The incorporation of polar codes in 5G-NR improves the data rate significantly with a low block error rate (BLER), in contrast to the convolution codes used in 4G LTE.

In the literature, an overview of 5G-NR with a brief introduction to various physical layer functionalities of the PDCCH chain is given in [2]. In this paper, concepts of control resource set (CORESET), search space (SS) set and PDCCH monitoring mechanism are explained in detail. In [3], the authors have described the physical layer aspects of NR PDCCH structure for the design of the SS set to improve blind decoding and the effectiveness of channel estimation. An enhancement to the already existing PDCCH hash function is proposed in [4], in which the author claimed to have achieved better inter-user blocking capability than the existing hash function. Link-level simulations are carried out using precoder cycling with a single antenna port adopting an aggregation level of 16 in [5]. The work in [6], describes the complete receiver functionalities of the 5G-NR PDCCH chain. Here rate-dematching, de-scrambling, demodulation and channel estimation are discussed in detail. In [7], the author has proposed new encoder architecture for 16-bit polar codes and implemented it on Xilinx XC5VLX110T and XC6SLX25T kit. By running the 4-parallel folded 16-bit polar code-based encoder architecture on the Xilinx vertex-5 13.2 and spartan-6 13.2, the performance of the polar code-based encoder with respect to logic utilisation has been explored and compared in the paper. However, note that this implementation is not 5G-NR compliant.

We can see from the literature survey, no work has been done on FPGA implementation of the polar coding in compliance with 3GPP standards for 5G-NR. Therefore, this work focuses on the FPGA implementation of polar

979-8-3503-0219-6/23/$31.00 ©2023 IEEE

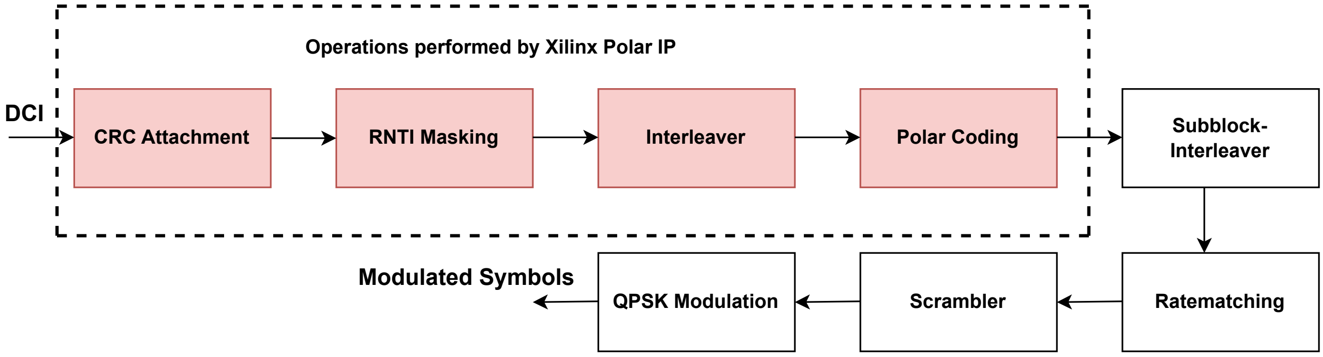


Fig. 1: PDCCH Transmitter Chain

coding for the 5G-NR PDCCH chain using Xilinx's polar IP. Note that the 3GPP standards provide the logic, but they do not tell how to implement that logic on hardware, which is a tedious task. Our contributions are given below

- Predominantly, our work is focused on configuring the Xilinx polar IP which is accomplished by generating the bit allocation (BA )table, configuring register values and passing the right control parameters. This is a challenging task, failing to which the polar IP behaves improperly. For this, we design a polar configuration IP using Vivado high-level synthesis (HLS) compiler that facilitates the program to target the MPSoC ZCU 111 evaluation kit.
- A parameter handler is developed which is a memory mapped module and provides an AXI-lite interface to the polar IP.
- To test the functionality of polar IP, a Verilog test bench is developed through which payload, BA table and configuration parameters in compliance with 3GPP standards are provided to the polar configuration IP.
- In Vivado integrated development environment (IDE), we verify the output of our implementation with the Matlab 5G-NR built-in library on the Xilinx evaluation board. Our results show the efficiency of the implementation in terms of latency and resource utilization on the evaluation board.

## II. 5G-NR PDCCH Chain Processing

Downlink control information (DCI) coming from the upper layer of the 5G-NR protocol stack undergoes series of different physical layer functionalities to establish a reliable link between transmitter and receiver. Fig. 1 depicts the functional blocks of the PDCCH chain for 5G-NR. This work focuses on FPGA implementation of polar coding. For this, we use Xilinx IP, which in addition to polar coding, also performs CRC attachment, RNTI masking and interleaving. These blocks are highlighted in Fig. 1. We next explain the operations of these blocks.

### *A. Xilinx Polar IP Additional Functionalities*

*1) **Cyclic Redundancy Check:*** Cyclic redundancy check (CRC) functionality is used to detect the errors at the receiver. According to TS 38.212 section 7.3.2 [1], a 24-bit CRC is calculated using a generator polynomial. The CRC is calculated by dividing the input bits by the generator polynomial. The remainder of the division operation is treated to be the check sum and is appended to the input. Let the output of CRC block be represented by a vector $\boldsymbol{b}$ whose last 24 bits are the CRC appended bits. The bits received at the receiver are divided by the same generator polynomial used at the transmitter. If the remainder of the modulo 2 division is zero, the transmission is successful else there are errors in the received bits.

*2) **RNTI Masking:*** Radio network temporary identifier (RNTI) is a 16-bit identifier and is used to differentiate single or a group of UEs. For example, system information (SI)-RNTI , paging (P)-RNTI are common RNTIs and are specific to group of UE's, whereas cell (C)-RNTI is a UE specific RNTI. One can refer TS 38.321 table 7.1.2 [8] for details about different RNTIs and their usage. The last 16-bits of the CRC output vector $\boldsymbol{b}$ are scrambled with a 16-bit cell specific RNTI mask so that the UE is able to distinguish different DCIs of same length and is able to decode the DCI for its unicast data. The output of RNTI masking is represented by a vector $\boldsymbol{c}$ such that,

$$c_k = \begin{cases} b_k \text{ for } k = 0, 1, 2 ... A+7 \\ b_k \oplus x_{k-A-8} \text{ for } k = A+8, A+9 \dots A+23, \end{cases} \tag{1}$$

where $x_k$ is the $k^{th}$ bit of the vector $\boldsymbol{x}$ representing the 16-bit RNTI mask.

*3) **Interleaving:*** The function of the interleaving block is to distribute CRC bits among information bits. For this, it uses the mother interleaving pattern as given in Table 5.3.1.1 of TS 38.212 [1]. The range of input bits that the interleaving block can accept is (12 bit DCI + 24 bit CRC = 36) $\leq K \leq$ (164 = 140 bit DCI + 24 bit CRC). The interleaver is activated through a flag $I_{IL}$ and used only for downlink. The steps for the interleaving are as follows

- Number of interleaved bits is upper bounded by $K_{IL}^{max} = 164$.
- Parameter $'h'$ is calculated as $h = K_{max} - K$. For example, let $K = 36$ and we know that $K_{max} = 164$. So, $h = 128$.
- Elements of $\Pi_{IL}^{max}$ in Table 5.3.1.1 of TS 38.212 are compared with $h$. In case they are larger, they are stored in $\Pi$.
- $h$ is subtracted from entries of $\Pi$ such that it contains integers smaller than $K$ in permuted order.
- $\Pi$= [129 132 134 138 139 140 128 130 133 135 141 131 136 142 137 143 144 145 146 147 148 149 150 151 152

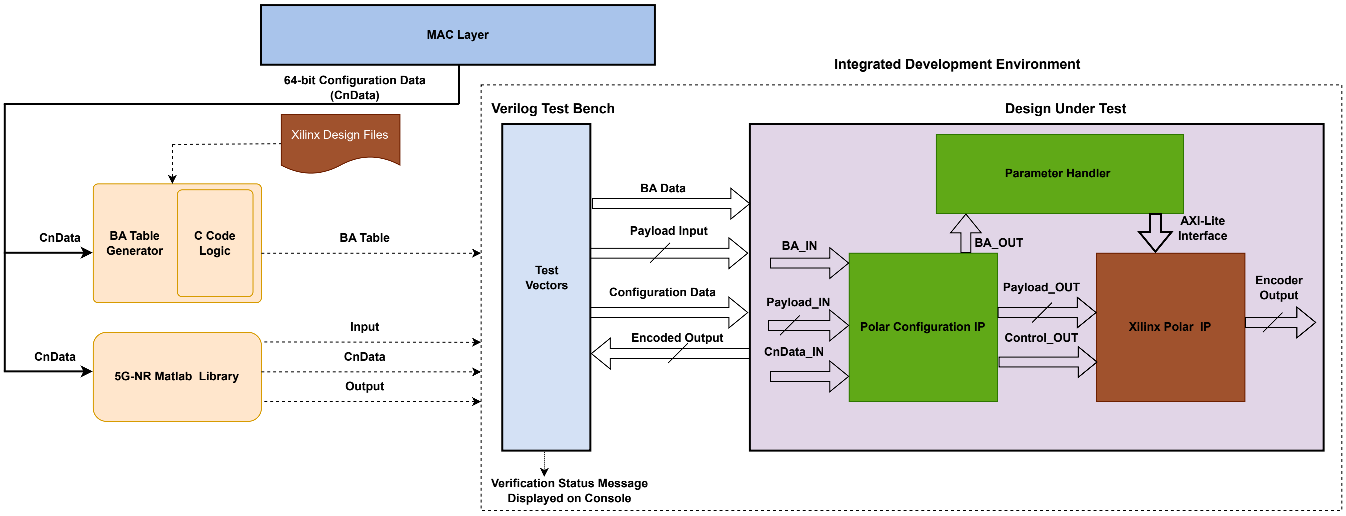


Fig. 2: Architecture for Polar IP Implementation.

153 154 155 156 157 158 159 160 161 162 163 ].

- Interleaving pattern is $\Pi-h$=[ 1 4 6 10 11 12 0 2 5 7 13 3 8 14 9 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35].

According to the interleaving pattern derived, the output of the interleaving block is denoted by vector $\boldsymbol{u}$ and interleaved bits are given by $u_k = c_{\Pi(k)}$.

### B. Polar Coding

1) ***Channel Polarization:*** The fundamental idea of polar codes stimulated with the exploration of basis generator matrix $\mathbf{G}_2 = \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix}$ which is generally known as polarization kernel [9]. The basis generator matrix $\mathbf{G}_2$ is used in an iterative manner for the construction of polar codes. The encoding of 2-bit input vector $\boldsymbol{x} = [x_0, x_1]^T$ is driven by the generator matrix $\mathbf{G}_2$ to give the encoded output $\boldsymbol{y} = [y_0, y_1]^T$ as $\boldsymbol{y} = \boldsymbol{x}\mathbf{G}_2$, where $y_0 = x_0 \oplus x_1, y_1 = x_1$. The concept of channel polarization holds good for the binary erasure channel, where the transmitted bits are received correctly with probability $1-e$ or lost with probability say, $e$. Let us analyze by modeling channel $C$ to binary erasure channel such that it is decomposed to $C^-$ and $C^+$ as

$$C^- : x_0 \rightarrow (y_0, y_1), C^+ : x_1 \rightarrow (y_0, y_1) \qquad (2)$$

The notation followed for (2), is that the $C^-$ channel is utilized to decode the input bit $x_0$ which is a function of $(y_0, y_1)$ and $C^+$ channel is utilized to decode the input bit $x_1$ which is a function of $(y_0, y_1)$ and also decoded bit $x_0$. As there are 2 channels, 4 possibilities exist as

- Both channels are decoded successfully with probability $(1-e)^2$.
- First channel is erased and second channel is decoded successfully with probability $e*(1-e)$.
- First channel is decoded successfully and second channel is erased with probability $(1-e)*e$.
- Both the channels are erased with probability $e^2$.

The bits are decoded sequentially and as $x_0 = y_0 \oplus y_1$, both the encoded bits have to be received correctly to extract the bit $x_0$. Hence $x_0$ is successfully decoded with probability $(1-e)^2$ and decoding fails with probability of $e_1$=$e*(1-e)$ +$(1-e)*e$+$e^2$ =$(2-e)*e$>$e$. As $x_1 = y_1$, through similar analysis we can arrive at the conclusion that $x_1$ is decoded successfully with probability $(1-e)^2$+$e*(1-e)$+$(1-e)*e$ =1-$e^2$ and fails with probability $e_2 = e^2$<$e$. Hence by the analysis we can conjecture that bit $x_0$ is transmitted over unreliable or a bad channel with probability of erasure $e_1 > e$ and bit $x_1$ is transmitted over a reliable or a good channel with probability of erasure $e_2 < e$.

2) ***5G-NR 3GPP Specifications for Polar Coding:*** The polar encoder of length $N$ is chosen such that $N = 2^n \geq K$, where $N$ represents the number of bits at the output of the polar encoder and $K$ is the number of bits input to the polar encoder. The minimum and maximum length of polar encoder is $N_{min} = 32$ and $N_{max} = 512$ corresponding to $n_{min} = 5$ and $n_{max} = 9$ respectively. The functionality of polar encoder as defined in TS 38.212, section 5.3.1.2 is given by $\boldsymbol{d} = \boldsymbol{u} * \mathbf{G}_N$ where $\boldsymbol{d}$ and $\boldsymbol{u}$ are output and input vectors of polar encoder. $\mathbf{G}_N = \mathbf{G}_2^{\otimes n}$ is the transformation matrix. The minimum acceptable code rate is 1/8. The polar encoder creates $N$ virtual channels of different reliability patterns, which are described in 38.212, Table 5.3.1.2-1 [1]. The objective is to determine the $K$ best reliable channels that form the indices of information set $I$ to transmit the information bits and $(N-K)$ unreliable channels that form the indices of frozen set $F$ which must be avoided due to poor channel quality.

## III. FPGA Implementation of 5G Polar Coding

Polar coding operation for the 5G-NR PDCCH channel is accomplished using Xilinx IP. However, it is important to note that to make this IP functional, one has to design additional IPs as shown in Fig. 2. One can say that FPGA implementation of polar coding is a significantly challenging task. In particular, as shown in Fig. 2, we design polar configuration IP which takes the configuration parameters coming from MAC layer constituting payload length $A$, length of the polar encoder $N$ and $E$, the number of bits to be transmitted on resource elements (REs). BA table is also fed as one among many inputs to polar configuration

TABLE I: REG2 Configuration

| Bit position | Functionality | Bit selection | Description |
|---|---|---|---|
| 31:6 | Reserved | All 0's | Reserved bits are set to 0 as off now, later can be utilized for any functionality. |
| 5 | Initialization of CRC | 1 | CRC bits are initialized to all 1's and appended to message bits prior to CRC calculation. |
| 4 | Interleaving (ITLV) | 1 | Interleaving of CRC bits among information bits is enabled for DCI. |
| 3:2 | CRC selection | 00 | 24-bit CRC is selected for DCI according to 3GPP 5G NR standards. |
| 1:0 | Code Augmentation | 10 | As 24-bit CRC is appended to information bits, CRC Augment is selected. |

TABLE II: Bit Types of the Codeword

| Bit type | Frozen | CRC | Information | Parity |
|---|---|---|---|---|
| Bits | 00 | 01 | 10 | 11 |

IP which then processes the inputs and yields the output in a manner the polar IP expects. The design of this IP also involves the calculation of appropriate register values to be fed to the polar IP. We use AXI-lite interface provided by a parameter handler, which is a memory mapped module for configuring code parameter register of the polar IP. The input, output and control ports of our IPs are designed with an AXI-stream interface. In addition to the above, one has to have a sheer understanding of Xilinx PG280 document [10] for integration of these modules with the polar IP. We now plunge into the implementation part in detail which is organized as below.

a) **Bit Allocation (BA) Table Generation:** The bit allocation table holds the bit-type of each bit position in $N$ bit polar encoder output. BA table comprises of $N/16$ contiguous entries, each of 32-bit. Every 32-bit entry describes the bit-type of 16-bit positions of polar encoder output. Two bits in a 32-bit entry describe the bit type of one-bit position of polar encoder output as per TABLE II. The BA table is generated as follows:

- Download the BA table design folder from the Xilinx website which contains C design files for downlink configuration, header files and matlab design files to generate test vectors for verification purpose.
- Modify the testbench file for downlink by writing a C code which takes the input parameters like the number of information bits $A$, the number of bits to be transmitted on resource elements (REs), $E$ and the number of CRC bits $L$. These input parameters are linked to other header files and functions of the downloaded folder.
- In the Vivado HLS command line, give the command pertaining to downlink which produces the required BA table to be fed to the polar configuration IP through $BA\_IN$ port as shown in Fig. 2.

b) **Configuring Code Parameter Registers:** The code parameter registers of 32-bit are utilized to program $N$, $K$, $L$, $CRC\_AUGMENT$, $CRC\_INIT$ for each codeword using AXI-lite interface. The polar code settings for DCI is as shown in the TABLE III.

TABLE III: DCI Mode Settings

| Use mode | ITLV | CRC_SEL | CRC_INIT | Augment |
|---|---|---|---|---|
| DCI | 1 | CRC24C | 1 | CRC |

The description of each entry of TABLE III is given in TABLE I. The 32-bit parameter registers are memory mapped in such a way that 20 bits from LSB represent the address and 12 bits from MSB hold the value. The values of each register should be calculated carefully in accordance with the document [10]. Recall that in addition to the polar coding, the polar IP also performs CRC attachment, scrambling of CRC bits with RNTI value and interleaving as explained in II-A. This is achieved by loading appropriate values in the register REG2 at the memory location 0x2008, as shown in TABLE I. The binary representation of the 6 bits from the bit selection column in TABLE I is 110010, which is 50 in decimal and 32 in hexadecimal. Therefore, the memory-mapped register REG2 holds **0x03202008**. In a similar manner, we have computed other register values which are classified as core parameters and polar code parameters in accordance with PG280 document [10]. Recall that to program the polar IP, we design the polar configuration IP and the parameter handler . The polar configuration IP feeds the BA table and the configured register values to the parameter handler through $BA\_OUT$ port as shown in Fig. 2. The parameter handler provides an AXI-lite interface and is used to configure memory-mapped registers of Xilinx polar IP. Algorithm 1, summarises the steps of the designed polar configuration IP for FPGA implementation of polar coding for 5G-NR PDCCH channel.

The block diagram of the design in the Vivado IDE for FPGA implementation is shown in Fig. 3. Here, the ports payload_in_0, BA_IN_0 and config_V_V_0 represent payload data, BA table and configuration data fed through Verilog testbench to polar configuration IP. Rest of the IPs fall into the category of Xilinx inbuilt IPs and their usages are explained here. ZynqUltra scale is used to provide synchronous clock to all the IPs in the design. Processor system reset IP is used as power on reset generation to all our designed IPs. The Virtual Input/Output (VIO) is a customizable IP used to drive internal FPGA signals in real time. As seen in the Fig. 3, VIO is connected to ap_start port of polar configuration IP through which we can give or alter input to test on hardware. The system Integrated Logic Analyzer (ILA) core IP is used to monitor the flow of signals in the implemented design. It offers probe ports according to user requirement for analyzing the waveform on FPGA.

## IV. FPGA Implementation Results

The parameters considered for implementation are strictly in agreement with the 5G-NR 3GPP standards. We

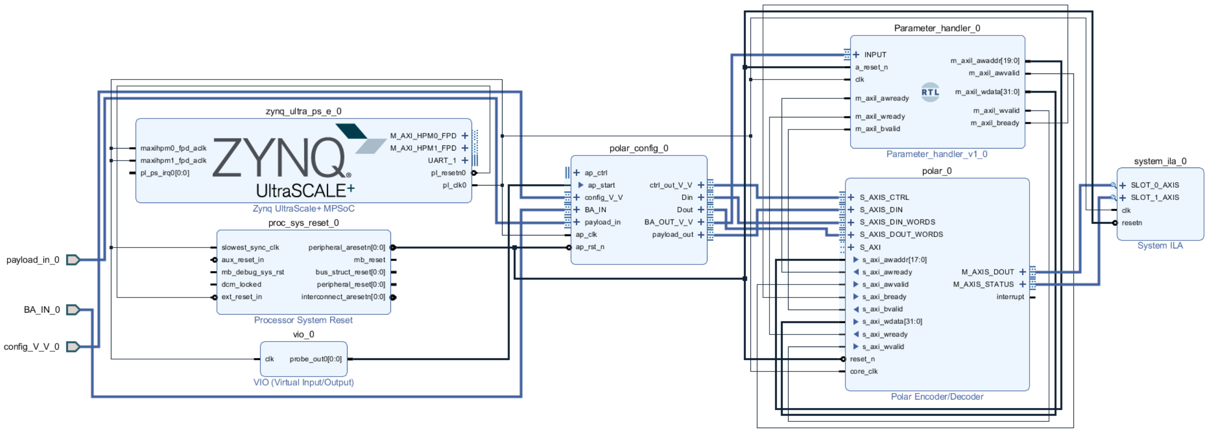


Fig. 3: FPGA Implementation of Xilinx's Polar IP in Vivado IDE.

```
Algorithm 1: Polar Configuration
Declare the input and output ports with
  appropriate size and data type.
State 1: Read the configuration information.
State 2: Write the pre-configured register values
  and configuration information on $BA\_OUT$ port.
$value \leftarrow$ 0x10000 (Starting Address of BA table).
for $c1 = 1$ to $N/16$ do
    State 3: Read the BA table entries from BA_IN
      port and write it on the $BA\_OUT$ port along
      with their addresses.
    $value \leftarrow value + 4$ (BA table address is
      incremented to next address).
end
State 4: Configure the 32-bit control interface for
  Xilinx polar IP.
State 5: Write the number of bytes per cycle at Din
  port to be read on the input of Xilinx polar IP.
State 6: Read the payload and write it on the
  payload output port.
State 7: Write the number of bytes per cycle at
  Dout port to be written on the output of Xilinx
  polar IP.
Revert to State 1 and reset the values.
```

have considered payload size (DCI length) $A = 78$, polar encoder output length $N = 128$ and $E = 108$, the number of bits to be transmitted on resource elements (REs). We use the Xilinx MPSoC ZCU 111 evaluation board for the FPGA implementation of the polar coding along with our IPs. The resources available and utilized on this hardware are mentioned in TABLE V. The design is implemented with a 100 Mhz clock. Xilinx's Vivado IDE is used for the implementation.

*1) **Simulation Waveform on Vivado IDE**:* Fig. 4 depicts the simulated waveform using Verilog test bench in Vivado IDE. The 128-bit input payload fed to the Xilinx polar IP is reflected on $tdata\_in$ port as 32-bit hexa decimal number. The 64-bit configuration data is reflected on $config\_data$ port. The BA table generated is seen on $ba\_data$ port. The 128-bit encoded output is seen on $tdata\_out$ port.

Matlab 5G-NR toolbox is used to verify the output generated by the polar IP. Test vectors comprising input, output and configuration data are generated using 5G-NR in-built function in Matlab for the polar encoder. These test vectors are fed to the polar configuration IP using the Verilog test bench and then output from the polar IP is read which is compared with the output test vector to generate $error\_count$. As we can see in Fig. 4 $error\_count$ is 0, implying that the polar encoder output generated from our design matches with Matlab generated output. Also, the message 'Test is successful' is displayed on the console. As we use AXI-stream interfaces, one key observation is that the valid output always corresponds to the assertion of $tvalid\_out$ and $tready\_out$ signal as seen in the waveform Fig. 4. This ensures a proper handshake between different blocks when connected in an integrated development environment.

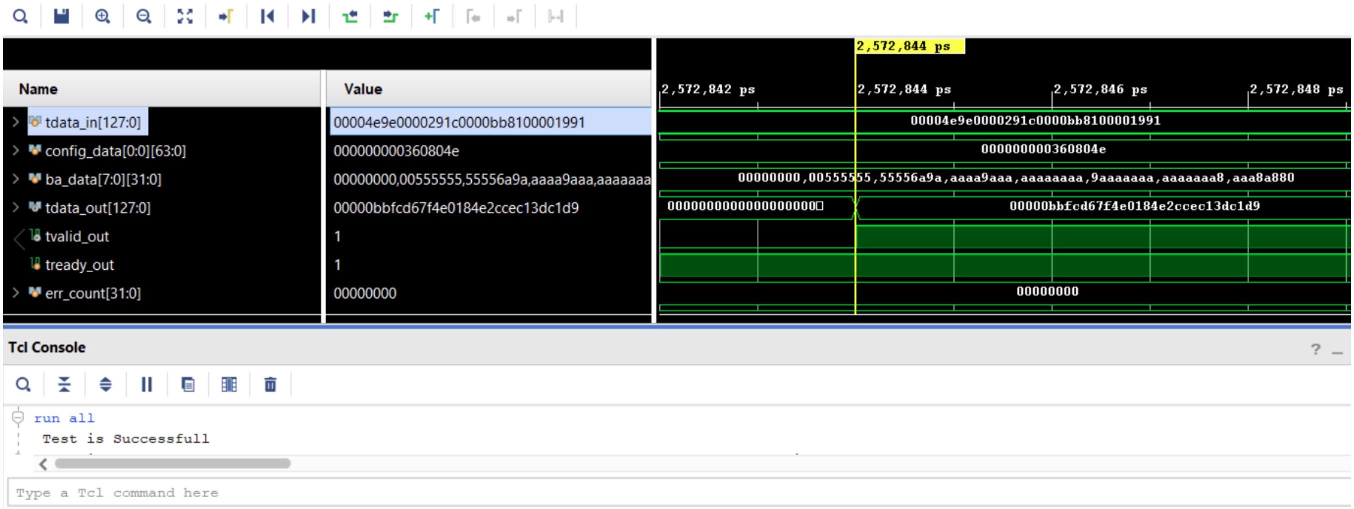
One can observe in Fig. 4 that the output of the polar IP is generated at 2.57$\mu s$, where the maximum delay incurred by our polar configuration IP is only 0.36$\mu s$.

*2) **Simulation Waveform on FPGA**:* Fig. 5 depicts the waveform as seen on Xilinx MPSoC ZCU 111 evaluation board. The output interface of polar IP is probed to slot0 of system ILA which facilitates the output to be seen on the hardware. The valid output corresponding to the assertion of $TVALID$ and $TREADY$ signals which can be observed on slot0:polar_0_M_AXIX_DOUT:TDATA signal. The interface being ACTIVE and that it is the last stream of data as we have tested for $N$=128 can also be observed in Fig. 5.

*3) **Latency and Resource Utilization Report of Polar Configuration IP**:* The latency report of the polar configuration IP is shown in TABLE IV. We see that it has a maximum latency of 0.36$\mu s$ which is significantly lower than the duration of an OFDM symbol in 5G-NR with any numerology. To achieve this number, the IP is optimized by following the state-wise operation of the block, such that the sequential transition is in accordance with the core functionality of the polar encoder.

TABLE IV: Latency Report

| Clock(MHz) | Latency(Min) | Latency(Max) |
|---|---|---|
| 100 | 10ns | 0.36$\mu s$ |



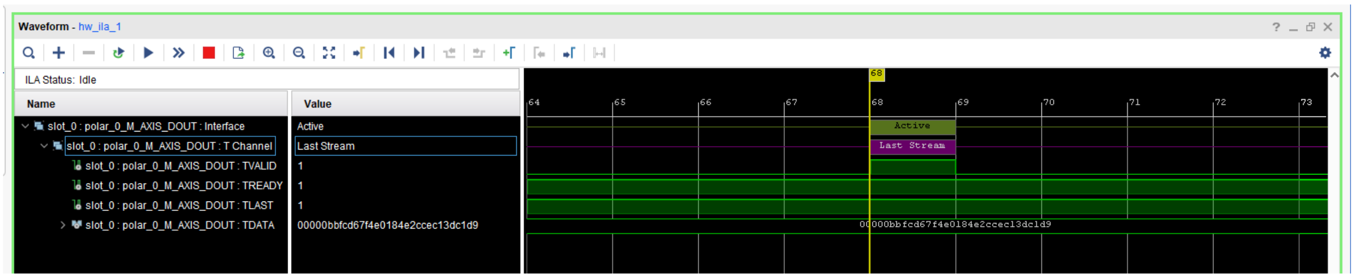
| Name | Value |
|---|---|
| tdata_in[127:0] | 00004e9e0000291c0000bb8100001991 |
| config_data[0:0][63:0] | 000000000360804e |
| ba_data[7:0][31:0] | 00000000,00555555,55556a9a,aaaa9aaa,aaaaaaa |
| tdata_out[127:0] | 00000bbfcd67f4e0184e2ccec13dc1d9 |
| tvalid_out | 1 |
| tready_out | 1 |
| err_count[31:0] | 00000000 |

Tcl Console

```
run all
 Test is Successfull
```

2,572,844 ps
2,572,842 ps
2,572,844 ps
2,572,846 ps
2,572,848 ps
00004e9e0000291c0000bb8100001991
000000000360804e
00000000,00555555,55556a9a,aaaa9aaa,aaaaaaaa,9aaaaaaa,aaaaaaa8,aaa8a880
00000000000000000000
00000bbfcd67f4e0184e2ccec13dc1d9
00000000
Type a Tcl command here

Fig. 4: Simulation Waveform



Waveform - hw_ila_1

ILA Status: Idle

| Name | Value |
|---|---|
| slot_0 : polar_0_M_AXIS_DOUT : Interface | Active |
| slot_0 : polar_0_M_AXIS_DOUT : T Channel | Last Stream |
| slot_0 : polar_0_M_AXIS_DOUT : TVALID | 1 |
| slot_0 : polar_0_M_AXIS_DOUT : TREADY | 1 |
| slot_0 : polar_0_M_AXIS_DOUT : TLAST | 1 |
| slot_0 : polar_0_M_AXIS_DOUT : TDATA | 00000bbfcd67f4e0184e2ccec13dc1d9 |

Fig. 5: Waveform as seen on Hardware

The resource utilization report of the polar configuration IP is shown in TABLE V. We see that the utilization percentage of the IP is negligible. We also see that the designed algorithm does not use BRAM_18K and complex DSP processors. This happens due to the efficient and low complex implementation of the IP.

TABLE V: Utilization Report

| Resources | Available | Utilization | Utilization% |
|---|---|---|---|
| BRAM_18K | 2160 | 0 | 0 |
| DSP48E | 4272 | 0 | 0 |
| Flip flop | 850560 | 449 | 0.053 |
| LUT | 425280 | 584 | 0.14 |

## V. Conclusions

We implemented the 3GPP complaint polar encoder for 5G-new radio (NR) on FPGA. For this, we designed polar configuration IP, a parameter handler to provide an AXI-lite interface and a Verilog testbench to test the functionality of the polar IP. We showed that the output of our implementation is matching with the Matlab 5G-NR built-in library. Our results showed the efficiency of the implementation in terms of latency and resource utilization on the evaluation board.

## References

[1] *Multiplexing and channel coding*, 3rd Generation Partnership Project (3GPP) TS 38.212, 2018, release 15 V.15.2.0.

[2] K. Takeda, H. Xu, T. Kim, K. Schober, and X. Lin, “Understanding the heart of the 5g air interface: An overview of physical downlink control channel for 5g new radio,” *IEEE Communications Standards Magazine*, vol. 4, no. 3, pp. 22–29, 2020.

[3] F. Hamidi-Sepehr, Y. Kwak, and D. Chatterjee, “5g nr pdcch: Design and performance,” in *2018 IEEE 5G World Forum (5GWF)*, 2018, pp. 250–255.

[4] V. Braun, K. Schober, and E. Tiirola, “5g nr physical downlink control channel: Design, performance and enhancements,” in *2019 IEEE Wireless Communications and Networking Conference (WCNC)*, 2019, pp. 1–6.

[5] S. Varatharaajan, M. Grossmann, and G. Del Galdo, “5g new radio physical downlink control channel reliability enhancements for multiple transmission-reception-point communications,” *IEEE Access*, vol. 10, pp. 97 394–97 407, 2022.

[6] L. Tianyi and Z. Huijie, “Research on pdcch channel in 5g nr system,” in *2021 International Conference on Networking and Network Applications (NaNA)*, 2021, pp. 113–118.

[7] A. Arpure and S. Gugulothu, “Fpga implementation of polar code based encoder architecture,” in *2016 International Conference on Communication and Signal Processing (ICCSP)*, 2016, pp. 0691–0695.

[8] *Medium Access Control (MAC) protocol specification*, 3rd Generation Partnership Project (3GPP) TS 38.321, 2018, release 15 V.15.3.0.

[9] V. Bioglio, C. Condo, and I. Land, “Design of polar codes in 5g new radio,” *IEEE Communications Surveys & Tutorials*, vol. 23, no. 1, pp. 29–40, 2021.

[10] *Polar Encoder/Decoder PG280*, Xilinx LogiCORE IP Product Guide, June 2018, v1.0.