SCIENCE CHINA
Information Sciences

• **Supplementary File** •

Fast Construction and Exploration of Performance-Cost Design Space for Belief Propagation Polar Decoders

Chao JI[1,2], You YOU[2], Weikang QIAN[3], Yongming HUANG[1,2] & Chuan ZHANG[1,2*]

[1] *Laboratory of Efficient Architectures for Digital-communication and Signal-processing (LEADS), National Mobile Communications Research Laboratory, Southeast University, Nanjing* 210096, *China;*
[2] *Purple Mountain Laboratories, Nanjing* 211111, *China;*
[3] *University of Michigan-Shanghai Jiao Tong University Joint Institute, Ministry of Education Key Laboratory of Artificial Intelligence, Shanghai Jiao Tong University, Shanghai* 200240, *China*

Appendix A Baseline architecture of BP polar decoders

Two architectures of BP polar decoders are proposed in our previous work [1], including the low-cost single-column and high-throughput double-column implementations. The paper takes the single-column design as the baseline architecture for the performance-cost evaluation of BP polar decoders. Figure A1 shows the baseline architecture with code length N and decoder parallelism M, which consists of MUX1, MUX2, MUX3, BCBs unit, SW_1 to SW_J, Hard Decision, Buffer, Pipeline Registers, Shared Memory, and Control Logic, where $J = \log_2 N/M + 2$. Note that the early termination unit of the baseline architecture is removed and all simulations with error-correction performance are performed at the same fixed iterations. Implementation details of the baseline architecture can be referred to [1].

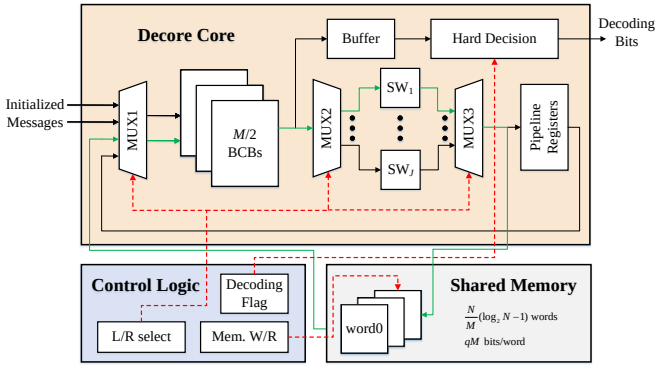


Figure A1 Baseline architecture of BP polar decoders.

Appendix B The mapping relationship between logic gates

In the realm of digital circuit design [2], gate circuits serve as fundamental hardware units that are utilized to establish various logic relationships and construct digital circuits, including combinational logic and sequential logic. Gate circuits include elemental units such as AND, OR, and NOT gates, along with combined units like NAND, NOR, and XOR gates, which can be represented by elemental units. As for the comparison of ASIC-based synthesis results in digital circuits, equivalent gates are usually used to evaluate the area. A common strategy is to select the 2-input NAND gate as the equivalent gate, and the entire consumption of equivalent gates can be computed by dividing the synthesized area by the area of a single 2-input NAND gate. Another method is to convert other gates into a combination of 2-input NAND gates. Different from the existing NAND-gate-based approaches, we divide digital circuits into two categories: combinational logic and sequential logic, then convert each part into a combination of 2-input NAND gates based on the mapping relationship between logic gates in this work.

Combinational logic circuits can be directly evaluated through the gate circuit, as depicted in Figure B1, which illustrates the simple mapping between elemental gates and the 2-input NAND gate. Sequential logic circuits, on the other hand, are typically implemented based on the flip-flop, and we select the holding-blocking D flip-flop as the basic unit to convert it into logic gates. Figure B1 also presents the logic-gate-based architecture of D flip-flop, where the conversion from the 3-input NAND gate to the 2-input NAND gate needs to be performed.

Appendix C Performance-cost metric evaluation

Appendix C.1 Fitting between *FSNR* and code parameters

For the simulation of error-correction performance, a general description is to use the FER-SNR curve. Influenced by the variation of K and N where code rate $R = K/N$, each design with given code parameters differs in the trend of FER-SNR curves. The

* Corresponding author (email: chzhang@seu.edu.cn)

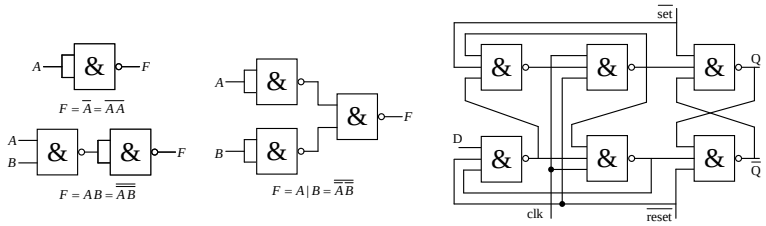


Figure B1 Mapping between elemental gates and the 2-input NAND gate.

goal of decoding algorithms is usually to achieve lower SNR while meeting specified error-correction performance constraints, and we evaluate different designs by comparing the corresponding SNR. In 5G mobile communications, FER $= 10^{-3}$ is a common requirement. Here, we set FER $= 10^{-3}$ as the unified constraint without loss of generality and use $FSNR$ to denote SNR under a given FER performance. We configure binary phase-shift keying (BPSK) modulation, additive white Gaussian noise (AWGN) channel, Gaussian approximation (GA) construction, the two-dimension offset min-sum (OMS) BP decoding with 7-bit quantization in [3], and 15 iterations to simulate the FER performance of polar codes. We simulate a series of designs where $1/4 \leqslant R \leqslant 3/4$ and $64 \leqslant N \leqslant 1024$, and Figure C1(a) compares FER performance of BP polar decoders under $R \in \{1/4, 1/2, 3/4\}$ and $N \in \{64, 128, 256\}$. Under the constraint of FER $= 10^{-3}$, we calculate the approximate $FSNR$ for those decoders shown in Figure C1(a). As indicated by the points of intersection from the red dashed lines, the results for $R \in \{1/4, 1/2, 3/4\}$ and $N = 64$ are $FSNR = 1.66$, 4.86, and 7.5, respectively. Also, it is easy to obtain the similar $FSNR$ under the cases of $N \in \{128, 256\}$.

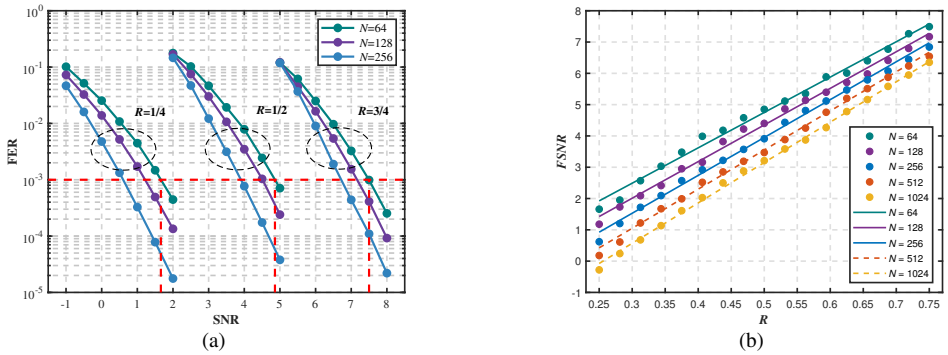


Figure C1 (a) FER performance of BP polar decoders under $R \in \{1/4, 1/2, 3/4\}$ and $N \in \{64, 128, 256\}$. (b) Fitting between R and $FSNR$ under $N \in \{64, 128, 256, 512, 1024\}$.

To determine the relationship between K, N and $FSNR$, decoding simulation under each N has to consider a wide range of R. By designating the step size of K, we simulate enough decoders under $N \in \{64, 128, 256, 512, 1024\}$ to achieve a satisfactory fitting, where the results of $N \in \{64, 128, 256\}$ are treated as the training set while that of $N \in \{512, 1024\}$ are considered to be the test set. As for the simulation of $N \in \{64, 128, 256, 512, 1024\}$, the corresponding step size of K is set as $\{2, 4, 8, 16, 32\}$ to get the same number of simulated decoders. Figure C1(b) gives the fitting curves of R and $FSNR$ for the training set. Here, the filled circles denote the actual simulation values of $FSNR$ while the solid and dashed lines are plotted to get the evaluated results of $FSNR$ based on the least-square fitting algorithm. From the view of Monte Carlo simulation, there is a nearly linear relationship between R and $FSNR$ by increasing the number of simulation samples under a fixed N, and the overall approximation error is within a controllable range. With 95% confidence bounds, the fitting expressions for $N \in \{64, 128, 256\}$ are approximated as $FSNR = 11.3R - 0.9$, $FSNR = 11.7R - 1.5$ and $FSNR = 12.1R - 2.1$, respectively. Moreover, the solid lines from the train set achieve a good fitting between R and SNR where the values of R-square are 0.9917, 0.9943, and 0.9961, respectively. It can be noted that the increments of slope and intercept between adjacent values of N are $\Delta_1 = 0.4$ and $\Delta_2 = -0.6$. Hence, we determine that the fitting expressions for the test set of $N \in \{512, 1024\}$ are $FSNR = 12.5R - 2.7$ and $FSNR = 12.9R - 3.3$ where the values of R-square are 0.9971 and 0.9981, respectively. Also, the dashed lines from the test set are a satisfactory approximation of the actual $FSNR$. The final relationship between R and $FSNR$ is given in (C1).

$$FSNR = (11.3 + (\log_2 N - 6)\Delta_1)R + (-0.9 + (\log_2 N - 6)\Delta_2) = (11.3 + 0.4(\log_2 N - 6))K/N - 0.9 - 0.6(\log_2 N - 6). \quad \text{(C1)}$$

Appendix C.2 Hardware representation based on the NAND gate

To determine the NAND-gate-based representation of the baseline architecture, we employ the 2-input NAND gate with the propagation delay of d_{NAND} as the evaluation unit. In the following discussion, for arbitrary circuit design, the consumption of NAND gates refers to the number of used 2-input NAND gates and the path delay is quantified by the total delay of passed 2-input NAND gates. Also, all NAND gates mentioned refer to the 2-input NAND gates.

In the analysis of combinational logic circuits, the truth table is used to derive the logic function between multiple input and output variables. Usually, the logic function expressions contain AND and OR operations of multivariable. For the AND case of multivariable, we assume that there are t logic variables such as $v_0, v_1, \ldots, v_{t-1}$. Since the AND operation of two logic variables can be represented by two NAND gates, the NAND-gate-based style of the former can be obtained by buliding the relationship between the AND operation of t logic variables and two logic variables. When t is even, t logic variables are decomposed into pairs of two elements to obtain $t/2$ pairs of elements, and then performing the AND operation on $t/2$ pairs of elements to compute

$t/2$ outputs. When t is odd, $t-1$ elements are decomposed into pairs of two elements to obtain $(t-1)/2$ pairs of elements, and then performing the AND operation on $(t-1)/2$ pairs of elements to compute $(t-1)/2$ outputs. As a result, the NAND gate consumption and the path delay of the AND operation of t logic variables are given by

$$\mathrm{NAND}_{\mathrm{AND}}(t)=\begin{cases}t-1+\mathrm{NAND}_{\mathrm{AND}}(\dfrac{t+1}{2}),t\ \text{is odd}\\ t+\mathrm{NAND}_{\mathrm{AND}}(\dfrac{t}{2}),t\ \text{is even}\end{cases}=2t-2,\mathrm{Delay}_{\mathrm{AND}}(t)=\begin{cases}2+\mathrm{Delay}_{\mathrm{AND}}(\dfrac{t+1}{2}),t\ \text{is odd}\\ 2+\mathrm{Delay}_{\mathrm{AND}}(\dfrac{t}{2}),t\ \text{is even}\end{cases},t\geqslant 2. \tag{C2}$$

Similar to the analysis of the AND case of multivariable, the NAND gate consumption and the path delay of the OR operation of t logic variables are given by

$$\mathrm{NAND}_{\mathrm{OR}}(t)=3t-3,\mathrm{Delay}_{\mathrm{OR}}(t)=1+\begin{cases}2+\mathrm{Delay}_{\mathrm{NAND}}(\dfrac{t+1}{2}),t\ \text{is odd}\\ 2+\mathrm{Delay}_{\mathrm{NAND}}(\dfrac{t}{2}),t\ \text{is even}\end{cases},t\geqslant 2. \tag{C3}$$

MUX1, MUX2, and MUX3 are the 4-input 2-output, 1-input J-output, and J-input 1-output multiplexers, respectively. As for these multiplexers, the data width of each input is qM where $q=7$ denotes the number of quantization bits. MUX1 contains one 2-bit controller which is used to select the suitable input signals, and a specific case of the 1-bit input is considered to simplify the analysis. Assuming that the inputs are D_0, D_1, D_2, and D_3, the outputs are Y_0 and Y_1, the control signals are S_0 and S_1. The logic relationship between D_0 to D_3 and Y_0 to Y_1 is given in (C4) according to the definition of MUX1.

$$Y_0Y_1=\begin{cases}D_0D_2,\ S_0S_1=00,\\ D_3D_2,\ S_0S_1=01,\\ D_1D_3,\ S_0S_1=10,\\ D_3D_0,\ S_0S_1=11.\end{cases} \tag{C4}$$

Then, the logic functions between D_0 to D_3 and Y_0 to Y_1 are given by

$$Y_0=\overline{S_0}\,\overline{S_1}D_0|S_0\overline{S_1}D_1|S_1D_3,\ Y_1=S_0S_1D_0|\overline{S_0}D_2|S_0\overline{S_1}D_3. \tag{C5}$$

Figure C2 gives the logic-gate-based architecture of MUX1. We obtain the NAND gate consumption of MUX1 by

$$\mathrm{NAND}_{\mathrm{MUX1}}=2+3\mathrm{NAND}_{\mathrm{AND}}(2)+2qM(3\mathrm{NAND}_{\mathrm{AND}}(2)+\mathrm{NAND}_{\mathrm{OR}}(3))=24qM+8, \tag{C6}$$

where the first and second terms represent the consumption of the NOT and AND operations related to S_0 and S_1, and the third term summarizes the results of logic functions associated with Y_0 and Y_1 by extending 1-bit to qM-bit as shown in the dashed box of Figure C2. Also, the data path delay of MUX1 is given by

$$\mathrm{Delay}_{\mathrm{MUX1}}=\mathrm{Delay}_{\mathrm{AND}}(2)+\mathrm{Delay}_{\mathrm{OR}}(3)=6. \tag{C7}$$

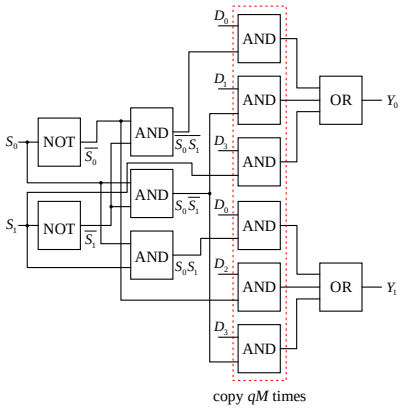


Figure C2 Logic-gate-based architecture of MUX1.

MUX2 and MUX3 achieve the opposite function and their controllers have the same number of bits l_J, i.e., $l_J=\lceil\log_2 J\rceil$. We use S_0 to S_{l_J-1} to represent the control signals of MUX2 and MUX3, with the full permutation of S_0 to S_{l_J-1} when $l_J=\log_2 J$, the NAND gate consumption of S_0 to S_{l_J-1} is given by

$$\mathrm{NAND}_{S_0^{l_J-1}}=\mathrm{NAND}_{S_0^{l_J-2}}+2^{l_J}\mathrm{NAND}_{\mathrm{AND}}(2)=\mathrm{NAND}_{S_0^{l_J-2}}+2^{l_J+1}=2^{l_J+2}-8,\ l_J\geqslant 1. \tag{C8}$$

Combining the case of J is not a power of 2, the NAND gate consumption of S_0 to S_{l_J-1} is given by

$$\mathrm{NAND}_{S_0^{l_J-1}}(J)=\mathrm{NAND}_{S_0^{l_J-2}}+J\mathrm{NAND}_{\mathrm{AND}}(2)=2^{l_J+1}-8+2J. \tag{C9}$$

Then, we obtain the NAND gate consumption and the data path delay of MUX2 by

$$\mathrm{NAND}_{\mathrm{MUX2}} = l_J + \mathrm{NAND}_{S_0^{l_J-1}}(J) + qMJ\mathrm{NAND}_{\mathrm{AND}}(2) = l_J + 2^{l_J+1} - 8 + 2J(1+qM), \mathrm{Delay}_{\mathrm{MUX2}} = \mathrm{Delay}_{\mathrm{AND}}(2) = 2. \quad \text{(C10)}$$

Also, we obatin the NAND gate consumption of MUX3 by

$$\mathrm{NAND}_{\mathrm{MUX3}} = l_J + \mathrm{NAND}_{S_0^{l_J-1}}(J) + qM(J\mathrm{NAND}_{\mathrm{AND}}(2) + \mathrm{NAND}_{\mathrm{OR}}(J)) = l_J + 2^{l_J+1} - 8 + 2J + qM(5J-3), \quad \text{(C11)}$$

and the data path delay of MUX3 by

$$\mathrm{Delay}_{\mathrm{MUX3}} = \mathrm{Delay}_{\mathrm{AND}}(2) + \mathrm{Delay}_{\mathrm{OR}}(J) = 3 + \mathrm{Delay}_{\mathrm{NAND}}(J). \quad \text{(C12)}$$

We demonstrate in [1] that SW_{J-1} is the routing module, SW_J is the reverse routing module, and SW_1 to SW_{J-2} are $(J-2)$ 2-input 2-output switch modules. SW_{J-1} and SW_J are actually implemented by wires without consuming any gate after being synthesized, while SW_1 to SW_{J-2} consists of delay registers which are evenly distributed at both sides and one 2-to-2 multiplexer. For the decoder with the parallelism M, the switch module contains $M/2$ groups of q-bit 2-to-2 multiplexers where these multiplexers employ the common control signal. Here, we again simplify the data width of the 2-to-2 multiplexer to 1-bit. Assuming that S_0 is the control signal, D_0, D_1 and Y_0, Y_1 are the inputs and outputs, we obtain the logic functions between D_0 to D_1 and Y_0 to Y_1 by

$$Y_0 = \overline{S_0}D_0|S_0D_1,\ Y_1 = S_0D_0|\overline{S_0}D_1. \quad \text{(C13)}$$

Then, the NAND gate consumption and the data path delay of the q-bit 2-to-2 multiplexer are given by

$$\mathrm{NAND}_{\mathrm{MUX}} = 1 + 2q(2\mathrm{NAND}_{\mathrm{AND}}(2) + \mathrm{NAND}_{\mathrm{OR}}(2)) = 1 + 14q, \mathrm{Delay}_{\mathrm{MUX}} = \mathrm{Delay}_{\mathrm{AND}}(2) + \mathrm{Delay}_{\mathrm{OR}}(2) = 4. \quad \text{(C14)}$$

Note that each switch module contains $M/2$ duplications of q-bit 2-to-2 multiplexers, so the NAND gate consumption and the data path delay of the multiplexer for SW_1 to SW_{J-2} are given by

$$\mathrm{NAND}_{\mathrm{SW}_1^{J-2}(\mathrm{MUX})} = (J-2)(1+14qM/2) = \log_2 N/M(1+7qM), \mathrm{Delay}_{\mathrm{SW}_1^{J-2}(\mathrm{MUX})} = \mathrm{Delay}_{\mathrm{MUX}}. \quad \text{(C15)}$$

Since each delay register can be implemented with the D flip-flop that consists of 18 NAND gates, the evaluation of delay registers for SW_1 to SW_{J-2} is given by

$$\mathrm{NAND}_{\mathrm{SW}_1^{J-2}(\mathrm{REG})} = 9qM \sum_{i=1}^{\log_2 N/M} \frac{N}{2^{i-1}M}. \quad \text{(C16)}$$

Then, we obtain the NAND gate consumption and the data path delay of SW_1 to SW_{J-2} by

$$\mathrm{NAND}_{\mathrm{SW}_1^{J-2}} = \mathrm{NAND}_{\mathrm{SW}_1^{J-2}(\mathrm{MUX})} + \mathrm{NAND}_{\mathrm{SW}_1^{J-2}(\mathrm{REG})}, \mathrm{Delay}_{\mathrm{SW}_1^{J-2}} = \mathrm{Delay}_{\mathrm{SW}_1^{J-2}(\mathrm{MUX})} = \mathrm{Delay}_{\mathrm{MUX}}. \quad \text{(C17)}$$

BCB is a 4-input 2-output module and consists of two duplications of $\mathrm{g}_1(x,y)$ and $\mathrm{g}_2(x,y)$ that perform message propagation in BP polar decoding. For the two-dimension OMS algorithm, $\mathrm{g}_1(x,y) = x + y$ and $\mathrm{g}_2(x,y)$ is given by

$$\mathrm{g}_2(x,y) = \begin{cases} \mathrm{sign}(x)\,\mathrm{sign}(y)\max(\min(|x|,|y|)-1,0), \text{left to right propagation} \\ \mathrm{sign}(x)\,\mathrm{sign}(y)\min(|x|,|y|), \text{right to left propagation.} \end{cases} \quad \text{(C18)}$$

Using the sign-magnitude scheme to quantize each message of BP polar decoders where the first bit and the latter $q-1$ bits denote the sign and magnitude of each message, the modified version of $\mathrm{g}_1(\cdot)$ is given by

$$\mathrm{g}_1(x,y) = \begin{cases} \mathrm{sign}(x)(|x|+|y|), \mathrm{sign}(x) \oplus \mathrm{sign}(y) = 0, \\ \mathrm{sign}(x)(|x|-|y|), \mathrm{sign}(x) \oplus \mathrm{sign}(y) = 1 \text{ and } |x| > |y|, \\ \mathrm{sign}(y)(|y|-|x|), \mathrm{sign}(x) \oplus \mathrm{sign}(y) = 1 \text{ and } |x| \leqslant |y|. \end{cases} \quad \text{(C19)}$$

We observe from (C18) and (C19) that the implementation of BCB involves the XOR operation, 6-bit comparator, and 6-bit adder-subtractor. By converting the XOR gate into the combination of elemental gates, we calculate the NAND gate consumption and the path delay of the XOR operation to be 9 and 5.

For the comparator, we consider three logic outputs including "greater than" (>), "equal to" (=), and "less than" (<). Figure C3(a) shows the logic gate architecture of the 1-bit comparator comp_1 and the x-bit ($x > 1$) comparator comp_x. We separate comp_x into comp_1 and comp_{x-1} to process the most significant bit and the lower inputs of $(x-1)$-bit. For comp_x, assume that the inputs are A_x and B_x, the logic outputs are $F_{A_x<B_x}$, $F_{A_x=B_x}$, and $F_{A_x>B_x}$, we can obtain by

$$F_{A_x<B_x} = F_{A_1<B_1}|F_{A_1=B_1}F_{A_{x-1}<B_{x-1}}, F_{A_x=B_x} = F_{A_1=B_1}F_{A_{x-1}=B_{x-1}}, F_{A_x>B_x} = F_{A_1>B_1}|F_{A_1=B_1}F_{A_{x-1}>B_{x-1}}. \quad \text{(C20)}$$

With $\mathrm{NAND}_{\mathrm{comp}_1} = 9$ and $\mathrm{Delay}_{\mathrm{comp}_1} = 5$, we obtain the NAND gate consumption and the data path delay of comp_x by

$$\mathrm{NAND}_{\mathrm{comp}_x} = \mathrm{NAND}_{\mathrm{comp}_1} + \mathrm{NAND}_{\mathrm{comp}_{x-1}} + 3\mathrm{NAND}_{\mathrm{AND}}(2) + 2\mathrm{NAND}_{\mathrm{OR}}(2) = 21x - 12, \mathrm{Delay}_{\mathrm{comp}_x} = \mathrm{Delay}_{\mathrm{comp}_{x-1}} + 4 = 4x + 1. \quad \text{(C21)}$$

Let $x = 6$, we calculate the NAND gate consumption and the data path delay of the 6-bit comparator to be 114 and 25.

For the adder-subtractor, the ripple carry adder is employed to implement this architecture where the addition and subtraction are switched by the carry input signal. Figure C3(b) gives an example of the 6-bit ripple carry adder with cin_0 as the selection

signal of addition-subtraction, where the critical path is marked by the green line. The 6-bit ripple carry adder consists of one 6-bit 2-to-1 multiplexer, 7 XOR gates, and 6 units of full-adder (FA), where the multiplexer is used to process the arithmetic overflow. We calculate the NAND gate consumption and the data path delay of the 6-bit adder-subtractor to be 256 and 44. Figure C3(c) gives the architectures of $g_1(\cdot)$ and $g_2(\cdot)$ with the comparator and adder, where the critical path is also marked by the green line. We calculate the NAND gate consumption and the data path delay of $g_1(\cdot)$ to be 475 and 74 while the NAND gate consumption and the data path delay of $g_2(\cdot)$ to be 622 and 82. Then, we obtain the NAND gate consumption and the data path delay of BCBs unit that contains $M/2$ BCBs by

$$\mathrm{NAND}_{\mathrm{BCBs}} = M/2(2\mathrm{NAND}_{g_1(\cdot)} + 2\mathrm{NAND}_{g_2(\cdot)}) = 1097M, \mathrm{Delay}_{\mathrm{BCBs}} = \mathrm{Delay}_{g_1(\cdot)} + \mathrm{Delay}_{g_2(\cdot)} = 156. \tag{C22}$$

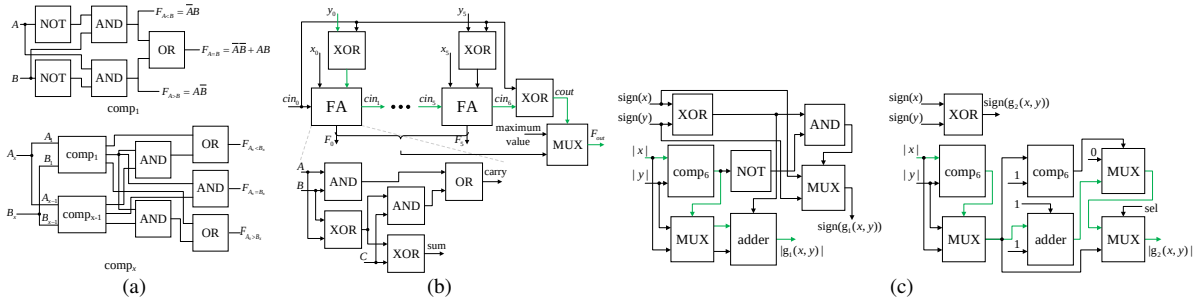


Figure C3 (a) comp_1 and comp_x. (b) 6-bit ripple carry adder. (c) $g_1(\cdot)$ and $g_2(\cdot)$.

Hard Decision is designed to decode the messages into source bits during the last iteration, and contains M units of $g_2(\cdot)$. Hence, the NAND gate consumption of Hard Decision is given by

$$\mathrm{NAND}_{\text{Hard Decision}} = M\mathrm{NAND}_{g_2(\cdot)} = 622M. \tag{C23}$$

Buffer is used to buffer the outputs of BCB into the decision module and consists of qM D flip-flops. Also, Pipeline Registers is employed to delay the outputs of MUX3 into the memory and consists of qN D flip-flops. As a result, the NAND gate consumptions of Buffer and Pipeline Registers are given by

$$\mathrm{NAND}_{\mathrm{Buffer}} = 18qM, \mathrm{NAND}_{\text{Pipeline Registers}} = 18qN. \tag{C24}$$

Shared Memory is the memory module used to store bidirectional messages and can be regarded as the combination of $qN(\log_2 N - 1)$ D flip-flops, one qM-bit 1-to-W multiplexer and one qM-bit W-to-1 multiplexer where $W = N/M(\log_2 N - 1)$. Referring to the derivations of MUX2 and MUX3, let $l_W = \lceil \log_2 W \rceil$, we obtain the NAND gate consumption and the data path delay of Shared Memory by

$$\mathrm{NAND}_{\text{Shared Memory}} = 18qN(\log_2 N - 1) + 2l_W + 2^{l_W+2} - 16 + 4W + qM(7W - 3), \mathrm{Delay}_{\text{Shared Memory}} = 5 + \mathrm{Delay}_{\mathrm{NAND}}(W). \tag{C25}$$

Control Logic is employed to provide the schedule of the baseline architecture. In this work, we use a finite state machine (FSM) to perform the control module and allocate one state for initialization, N/M states for decoding decision and $2N(\log_2 N - 1)/M$ states for iterative decoding. Hence, the number of states for Control Logic is given by

$$CL = 1 + N/M + 2N(\log_2 N - 1)/M. \tag{C26}$$

We use l_{CL} signals to implement the FSM where $l_{CL} = \lceil \log_2 CL \rceil$. Since FSM is the most key constitution of Control Logic, we employ the NAND gate consumption of l_{CL}-bit signals to approximate the result of the control module by

$$\mathrm{NAND}_{\text{Control Logic}} = l_{CL} + \mathrm{NAND}_{S_0^{l_{CL}-1}}(CL) = l_{CL} + 2^{l_{CL}+1} - 8 + 2CL. \tag{C27}$$

For all units listed in the baseline architecture, we derive the approximate NAND gate consumption and the data path delay related to each module. Then, we sum the results of all associated unts to obtain the NAND gate consumption $\mathrm{NAND}(N, M)$ and the critical path delay $\mathrm{Delay}(N, M)$ of BP polar decoders with code length N and decoding parallelism M in (C28).

$$\begin{aligned}\mathrm{NAND}(N, M) = {} & \mathrm{NAND}_{\mathrm{MUX1}} + \mathrm{NAND}_{\mathrm{MUX2}} + \mathrm{NAND}_{\mathrm{MUX3}} + \mathrm{NAND}_{\mathrm{BCBs}} + \mathrm{NAND}_{\text{Hard Decision}} + \mathrm{NAND}_{\mathrm{SW}_1^{J-2}} \\ & + \mathrm{NAND}_{\mathrm{Buffer}} + \mathrm{NAND}_{\text{Pipeline Registers}} + \mathrm{NAND}_{\text{Shared Memory}} + \mathrm{NAND}_{\text{Control Logic}}, \\ \mathrm{Delay}(N, M) = {} & \mathrm{Delay}_{\mathrm{MUX1}} + \mathrm{Delay}_{\mathrm{MUX2}} + \mathrm{Delay}_{\mathrm{MUX3}} + \mathrm{Delay}_{\mathrm{BCB}} + \mathrm{Delay}_{\mathrm{SW}_1^{J-2}} + \mathrm{Delay}_{\text{Shared Memory}}.\end{aligned} \tag{C28}$$

Appendix D Experiments of evaluation strategy validation

We declare in [1] that BP polar decoders with different values of N and M have almost the same critical path that is mainly dominated by BCB. Table D1 provides the estimated critical path delay of BP polar decoders under $N = 128$ and $2 \leqslant M \leqslant N$, where we still assume that all decoders share the same critical path, i.e., the delay of 178 NAND gates from $M = 128$. It can be seen that the maximum estimation error is an acceptable result of 10.1%, similar low-error results also can be calculated for other values of N. Hence, the delay approximation is a feasible strategy and is applied in the following discussion.

To evaluate the NAND gate consumption from our method and EDA tools [4], we use Synopsys Design Compiler to obtain the area from synthesized decoders under $N = 128$ and $2 \leqslant M \leqslant N$, then convert these results into the NAND gate consumption by

Table D1 Estimated critical path delay of BP polar decoders under $N = 128$ and $2 \leqslant M \leqslant N$.

	$M = 2$	$M = 4$	$M = 8$	$M = 16$	$M = 32$	$M = 64$	$M = 128$
Delay	198	196	194	192	188	186	178
Error	10.1%	9.18%	8.25%	7.29%	5.32%	4.3%	0%

dividing the area of the minimum 2-input NAND gate (about 1.12 μm^2) at SMIC 65 nm CMOS technology. Figure D1 compares the synthesized and estimated NAND gate consumptions of the baseline architecture, which are plotted by the green and blue dashed lines with the filled circle, respectively. We scale the estimation results by multiplying a certain parameter to get the same order of magnitude as the synthesis results, and the scaled results are plotted by the red dashed line. We omit the results of Buffer in Figure D1 since it has the same scaling parameter as Pipeline Registers, and provide the following analysis:

• MUX1, BCBs, and Hard Decision: In Figures D1(a), D1(d), and D1(e), it can be seen that the synthesized NAND gate consumptions of MUX1, BCBs, and Hard Decision keep a linear relationship with $\log_2 M$. The scaling parameters are approximately calculated to be 0.48, 0.26, and 0.23 while the average estimation errors are 2.77%, 0.44%, and 2.05%.

• MUX2 and MUX3: In Figures D1(b) and D1(c), the scaled estimation results achieve a good overlap with the synthesis results under sufferable errors, where the scaling parameters are approximately calculated to be 0.65 and 0.45. The synthesis results keep a nonlinear increase with the downward slope when the value of $\log_2 M$ changes, which is consistent with the trend of the estimation results. The average estimation errors for MUX2 and MUX3 are 4.94% and 9.26%.

• SW_1^{J-2}: In Figure D1(f), the scaling estimation results with the parameter of 0.31 almost overlap with the synthesis results. Moreover, the synthesis results keep a slow increase till reaching the maximum value of $M = 16$, and then go through a sharp decline as the value of M increases. The average estimation error for SW_1^{J-2} is 1.53%.

• Pipeline Registers: In Figure D1(g), both the synthesis and estimation results keep constant without being affected by M as the NAND gate consumption of Pipeline Registers depends on N. Also, the scaling parameter is approximately calculated to be 0.29. The average estimation error for Pipeline Registers is 0.58%.

• Shared Memory: In Figure D1(h), the scaled estimation results also achieve a good overlap with the synthesis results when setting the parameter of 0.38. Moreover, both the synthesis and estimation results go through a slow decline as the value of M increases. The average estimation error for Shared Memory is 1.04%.

• Control Logic: In Figure D1(i), the estimation results provide a relatively satisfying evaluation for the NAND gate consumption of Control Logic compared to the synthesis results, which means that the approximation with FSM is effective. The average estimation error for Control Logic is 24.28%.

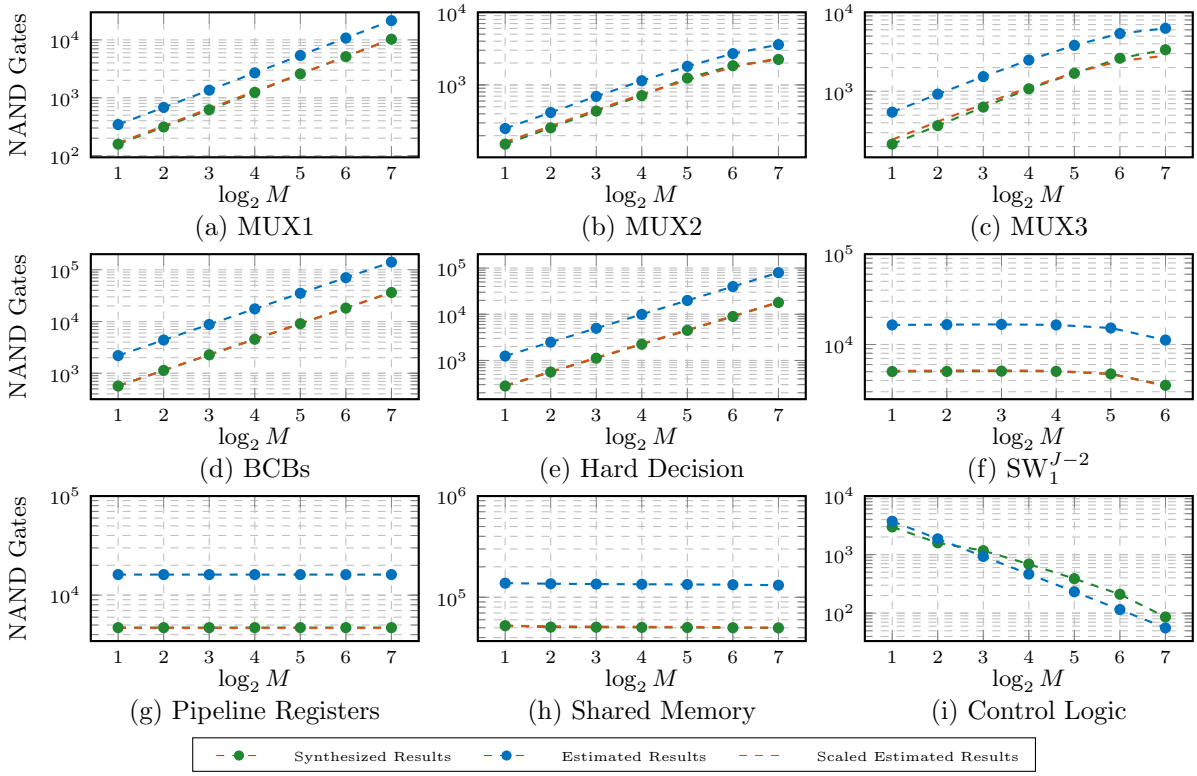


Figure D1 Comparison of the NAND gate consumption between the synthesis and estimation results of the baseline architecture.

In summary, the estimation errors may come from the optimization strategy adopted by EDA tools for different logic gates. Except for Control Logic with a relatively large estimation error, the remaining units all provide a satisfying approximation to the synthesized areas. Control Logic with a high approximation error has little impact on the estimation of the entire decoder since it occupies only a small portion of the total area. Notably, the scaling parameter of each unit under $N = 128$ is also applicable to other cases of $N \neq 128$, which demonstrates that our method provides a good area evaluation of BP polar decoders.

Appendix E Design space exploration of BP polar decoders

Appendix E.1 Multi-objective optimization model

According to the synthesis report of the critical path, we assume reasonably that the minimum 2-input NAND gate delay is $t_{\mathrm{NAND}} = 0.05\,\mathrm{ns}$ so that the critical paths of BP polar decoders with different code parameters have the same result of $178t_{\mathrm{NAND}} = 8.9\,\mathrm{ns}$. Note that the synthesized critical path is approximately 4.1 ns, we adopt the scaling parameter of $s = 0.46$ to reduce the error from the estimation. Set the maximum number of iterations *iter* as 15, we deploy the iteration-level decoding scheduling in [5] and calculate information throughput of the baseline architecture by

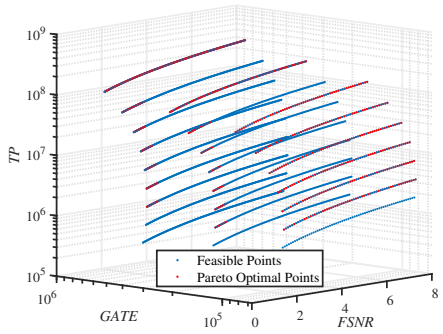
$$TP = \frac{K}{(2(\log_2 N - 1) \times iter + 1) \times N/M \times \mathrm{Delay}(N, M) \times t_{\mathrm{NAND}} \times s}. \tag{E1}$$

Let $GATE$ represent the NAND gate consumption, we obtain the multi-objective optimization model of BP polar decoders with the constraints on N, M, and K in (E2), where $f_1(\cdot)$ and $f_3(\cdot)$ are given by (C1) and (E1) while $f_2(\cdot)$ is determined by introducing the scaling parameters into (C28). Note that (E2) is a non-linear discrete integer multi-objective optimization problem, and an effective strategy is to enumerate all solutions of the model since the feasible region is constructed by finite points, then build the Pareto front consisting of the optimal solutions based on the mutual constraints between multiple objectives.

$$\begin{aligned} &\text{Goals}: && \min\ FSNR = f_1(N,K), \min\ GATE = f_2(N,M), \max\ TP = f_3(N,M,K); \\ &\text{Constraints}: && 2 \leqslant M \leqslant N,\ N/4 \leqslant K \leqslant 3N/4,\ N, M \text{ are the power of } 2,\ K \text{ is an integer}. \end{aligned} \tag{E2}$$

Appendix E.2 Pareto solutions of design space

Since the actual design space of BP polar decoders contains infinite solutions contributed by varying code parameters, we reduce the size of the design space to better describe the performance-cost exploration. We evaluate the decoders under $N = \{128, 256, 512\}$ and the constraints are consistent with (E2), then we obtain 3800 feasible solutions in the design space where each solution with different metric values is represented by unique design parameters of N, M, and K.



(a)

Solutions		Key metrics		
		TP [bps]	*GATE* [gates]	*FSNR* [dB]
Solution 1	Contraints	$\geqslant 10^7$	$\leqslant 10^5$	$\leqslant 1.45$
	Results	1.08×10^7	85419	1.43
	Matched?	Yes	Yes	Yes
Solution 2	Contraints	$\geqslant 10^7$	$\leqslant 10^5$	$\leqslant 1.45$
	Results	2.16×10^7	99278	1.43
	Matched?	Yes	Yes	Yes

(b)

Figure E1 (a) Three-dimension design space exploration of BP polar decoders. (b) Verification of constraint-satisfying solutions.

Figure E1(a) provides the design space of $\{TP, GATE, FSNR\}$, where blue and red solid circles also represent the feasible points and Pareto optimal points. Notably, the Pareto optimal set is shown to be the solution to the optimization problem in (E2) under $N = \{128, 256, 512\}$. Imposing additional requirements such as $TP \geqslant 10^7$ bps, $GATE \leqslant 10^5$ gates and $FSNR \leqslant 1.45$ dB at FER $= 10^{-3}$, we search the design space and then determine the decoders with $\{\text{Solution 1}: N = 128, M = 32, K = 32\}$ and $\{\text{Solution 2}: N = 128, M = 64, K = 32\}$ as the matched solutions while the remaining points cannot satisfy the joint restrictions. To validate the found decoders, we list the implementation results of TP, $GATE$ and $FSNR$ for the two designs in Figure E1(b). It can be observed that the three metrics all meet the predefined constraints, and the showcase verifies the efficacy of our performance-cost design space exploration.

Design space exploration itself will not guide how to implement BP polar decoders, but helps find feasible designs that satisfy the given constraints. The design space is built with the performance-cost metrics of BP polar decoders, exploring it essentially is to make trade-offs in multi-dimension metrics as we model this as a multi-objective optimization problem. The ultimate goal of the performance-cost design space exploration is to reach a specific balance state, in which all the metrics of the found designs meet the predefined requirements. Instead of targeting the feasible points in the entire design space, we merely apply auto-generation in [1] to the found design parameters of $\{N, M, K\}$ and deploy a specific synthesis into the resulting RTL implementations, shortening the entire evaluation cycle and further saving hardware development costs. Interested readers for the auto-generation framework can refer to our previous works in [1, 6].

References

1 Ji C, Shen Y F, Zhang Z Z, et al. Autogeneration of pipelined belief propagation polar decoders. IEEE T VLSI Syst, 2020, 28(7): 1703-1716.
2 Vahid F. Digital design with RTL design, VHDL, and Verilog. John Wiley & Sons, 2010.
3 Xu W H, Tan X S, Be'ery Y, et al. Deep learning-aided belief propagation decoder for polar codes. IEEE J Emerging Sel Topics Circuits Syst, 2020, 10(2): 189-203.
4 Lavagno L, Scheffer L, Martin G. EDA for IC implementation, circuit design, and process technology. CRC press, 2018.
5 Yuan B, Parhi K K. Early stopping criteria for energy-efficient low-latency belief-propagation polar code decoders. IEEE T Signal Process, 2014, 62(24): 6496-6506.
6 Zhong Z W, Gross W J, Zhang Z Z, et al. Polar compiler: Auto-generator of hardware architectures for polar encoders. IEEE T Circuits Syst I, 2020, 67(6): 2091-2102.