A scalable ASIP for BP Polar decoding with multiple code lengths

Wan Qiao[1] and Dake Liu[1, a]

[1]*The Institute of Application Specific Instruction-set Processor, Beijing Institute of Technology, 5 South Zhongguancun Street, Haidian District, 100081, Beijing, China (2120160687@bit.edu.cn)*

Abstract: In this paper, we propose a flexible scalable BP Polar decoding application-specific instruction set processor (PASIP) that supports multiple code lengths (64 to 4096) and any code rates. High throughputs and sufficient programmability are achieved by the single-instruction-multiple-data (SIMD) based architecture and specially designed Polar decoding acceleration instructions. The synthesis result using 65 nm CMOS technology shows that the total area of PASIP is 2.71 mm^2. PASIP provides the maximum throughput of 1563 Mbps (for N = 1024) at the work frequency of 400MHz. The comparison with state-of-art Polar decoders reveals PASIP's high area efficiency.

1 Introduction

The Polar code has been proved to be the first kind of error correction code that can achieve the Shannon capacity [1]. In 2016, 3GPP adopted the Polar code as the channel coding scheme for the control channel of the Enhanced Mobile Broadband (eMBB) scenario in future 5G standard [2]. The Polar code has drawn more and more attention these years.

Basically, Polar codes can be decoded by two kinds of algorithms: the successive cancellation (SC) algorithm and the belief propagation (BP) algorithm. SC decoders suffer from the high decoding latency and the low throughput caused by the serial processing nature. Compared with SC decoders, BP decoders provide higher throughputs because of the inherent high parallelism. However, BP decoders suffer from higher computation complexity and require larger memories for implementation. There have been several BP decoding proposals in literature for solving these problems. Yuan *et al.* proposed an early stopping criteria that can reduce about 30% of the average BP decoding complexity [3]. Park *et al.* [4] and Sha *et al.* [5] reduced about half of the memory requirement of BP decoding by double-column processing and combine-stage processing separately.

The BP Polar decoders mentioned above are all implemented as application specific integrated circuits (ASICs) that each supports only one fixed code length. However, eMBB control channel coding requires support for multiple code lengths [6]. ASICs lack flexibility for this application scenario. Pamuk proposed a FPGA implementation for Polar decoding with multiple code lengths [7]. But the throughput of this FPGA-based design is only about 50Mbps at the clock frequency of 160 MHz, which is too small for 5G communication systems. Application-specific instruction set processor (ASIP) is a promising solution to the predefined scope of application that requires high performance and sufficient flexibility at the same time [8], and thus is more suitable for Polar decoding in complex application scenarios.

In this paper, we propose a flexible scalable Polar decoding ASIP (PASIP) that supports multiple code lengths and any code rates. Single-instruction-multiple-data (SIMD) architecture is adopted in PASIP. The operands are divided by multiple parallel windows and processed in parallel to enhance the decoding throughput. A specially designed instruction set that contains Polar decoding acceleration instructions ensures the high throughput as well as the sufficient programmability of PASIP. PASIP is synthesized using 65 nm CMOS technology. And the synthesis results show that PASIP achieves much higher area efficiency than the sum of multiple single-code BP Polar decoders.

The rest of this paper is arranged as follows. Section 2 introduces some necessary background on the Polar BP decoding algorithm. Section 3 presents the architecture of PASIP in detail. Section 4 analyses the performance of PASIP. Section 5 provides the synthesis results and comparisons. Finally, Section 6 concludes the paper.

2 BP decoding algorithm

The BP Polar decoding procedure can be described using a factor graph [1]. Figure 1 (a) shows a unified factor graph with the code length N= 8 [7]. There are n computational stages (n = $\log_2 N$) that each consists of N/2 basic computational units (BCUs) in the factor graph. The structure of the BCU is shown in Figure 1 (b). There are n + 1 columns (Col.) that each consists of N nodes in the factor graph. The coordinates (i, j)

[a] Dake Liu: dake@bit.edu.cn

represent the *j*-th node in Col.*i*. Each node saves one intermediate result. All nodes should be initialized before the decoding begins. The nodes in Col.0 (the left most column) are set to infinity or 0 according to the locations of frozen bits and information bits separately. The nodes in Col.*n* are initialized by log likelihood ratios (LLR) of received data. Other nodes are set to 0.

As shown in Figure 1 (a), the BP Polar decoding procedure can be divided into three steps: the right-to-left propagation (LP), the left-to-right propagation (RP), and the final result hard decision. The nodes are updated column by column with right-to-left messages ($L_{i,j}$) calculated as Formula (1) and (2) during LP. Whereas RP updates the nodes using left-to-right messages ($R_{i,j}$) calculated by Formula (3) and (4). The final results are determined by the criterion shown in Formula (5). In Formula (1) to (4), $f(x, y) = 0.9375 \times \text{sign}(x)\,\text{sign}(y)\,\min(|x|, |y|)$.

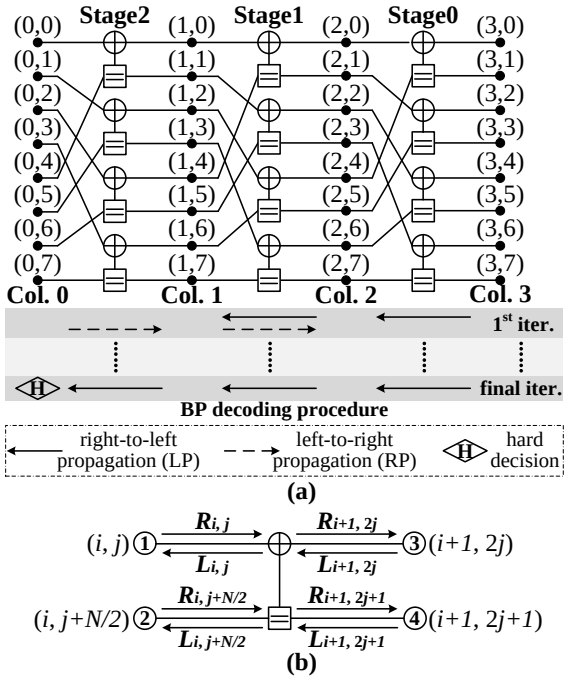

Figure 1. (a) The factor graph of Polar code (N=8), and the BP decoding procedure. (b) The structure of BCUs.

$$L_{i,j} = f\left(L_{i+1,2j}, L_{i+1,2j+1} + R_{i,j+N/2}\right) \quad (1)$$

$$L_{i,j+N/2} = f\left(R_{i,j}, L_{i+1,2j}\right) + L_{i+1,2j+1} \quad (2)$$

$$R_{i+1,2j} = f\left(R_{i,j}, L_{i+1,2j+1} + R_{i,j+N/2}\right) \quad (3)$$

$$R_{i+1,2j+1} = f\left(R_{i,j}, L_{i+1,2j}\right) + R_{i,j+N/2} \quad (4)$$

$$\hat{u}_j = \begin{cases} 1, & L_{0,j} + R_{0,j} < 0 \\ 0, & L_{0,j} + R_{0,j} \geq 0 \end{cases} \quad (5)$$

3 PASIP design

3.1 Top level architecture and pipeline structure

The top level architecture and the pipeline structure of PASIP are shown in Figure 2. PASIP adopts the SIMD architecture, and accelerates the decoding calculation through data-level parallelization. P = 16 homogeneous parallel windows are adopted in this paper.

PASIP consists of three submodules: the data path, the memory subsystem, and the control path. The decoding computation is executed in the data path that is composed of 16 homogeneous soft-in-soft-out (SISO) modules. The memory subsystem consists of a memory array, an address generation unit (AGU), and permutation networks. There are $8P$ 128-bit × 32 memory banks in the memory array (8 in each parallel window and 128 in total) for saving all $N(n+1)$ nodes in the factor graph. The AGU calculates the address signals, including memory access addresses, the write enable signal (*wen*), the sliding window number (*w*), and the sliding window address (*t*). And the permutation networks handle the data shuffling between memories and SISOs. Other blocks belong to the control path. The programmability of PASIP is mainly achieved by the control path.

The pipeline of PASIP contains 6 stages, including an instruction fetch (IF) stage, an instruction decoding (ID) stage, an address generation (AG) stage, an execution (EXE) stage, and a write-back (WB) stage. During BP decoding process, the intermediate results of one stage are part of inputs of the next stage. To avoid the potential read-after-write (RAW) data hazard, data forwarding technique is introduced, and the permuted SISO outputs are sent to the input-end of the EXE stage.

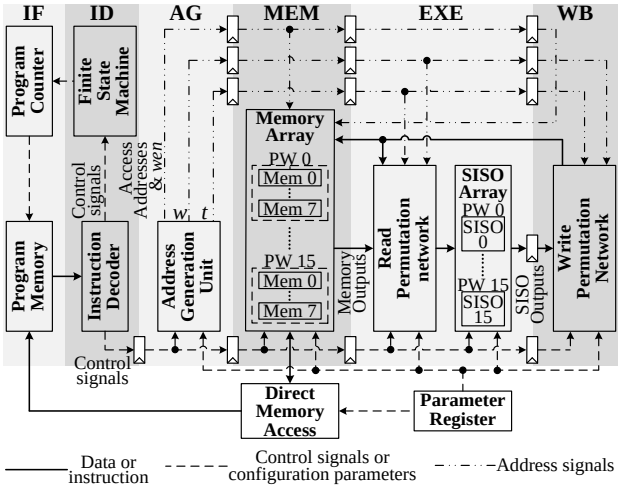

Figure 2. The top level architecture and the pipeline structure of PASIP.

3.2 BCU segmentation and data path design

As shown in Figure 1, BCUs in the same computational stage do not have data dependency. And thus, one stage of BCUs can be divided by multiple parallel windows and computed in parallel.

The BCUs are segmented using parallel windows (PWs) and sliding windows (SWs) according to the segmentation scheme shown in Figure 3. One position in the SW contains k = 32 BCUs. The number of positions in one SW is referred to as the *sliding window length* (*SWL*). One SISO only handles one position in the SW at a time, and one stage of computation takes up *SWL* clock cycles in total. In this way, the fixed hardware is able to support multiple code lengths.

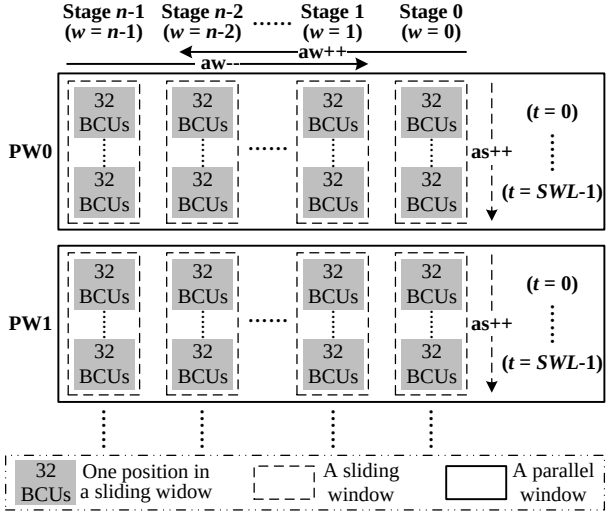

Figure 3. BCU segmentation scheme.

Figure 4 (a) shows the structure of the SISO module. PASIP adopts 8-bit fixed-point data type in 2's complement format. As one SISO module can handle 32 BCUs at a time, each SISO module requires 128 8-bit inputs every clock cycle. The inputs are divided into four groups according to the corresponding node positions in the BCUs (①, ②, ③, and ④ means four positions as shown in Figure 1), and the computation logic is designed according to Formula (1) to (5). As the Formula (1) and (2) have the same format as Formula (3) and (4), LP and RP can fully share the computation logic.

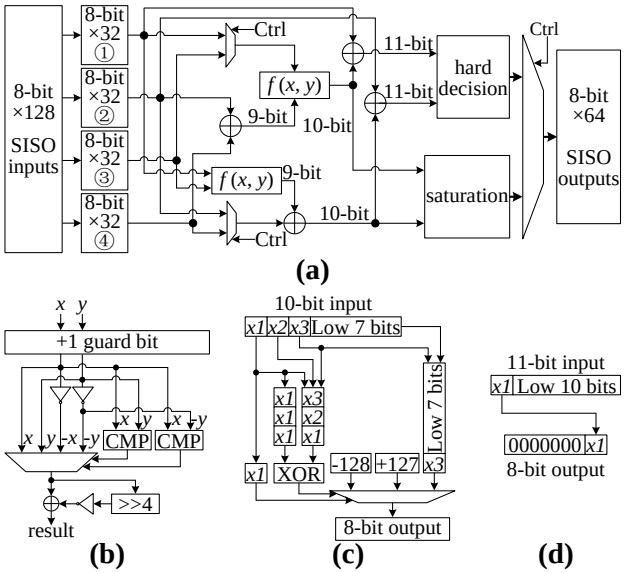

Figure 4. (a) SISO structure (b) $f(x, y)$ module (c) saturation module (d) hard decision module.

The function $f(x, y)$ is implemented as Figure 4 (b). The truth table of the selector is shown in Table 1. The multiplication with 0.9375 is replaced by $x + (- x >> 4)$ for easier hardware implementation.

Guard bits are inserted before inverse operations and add operations in order to eliminate data overflow. The saturation block shown in Figure 4 (c) is introduced to saturate the computation results before writing back. If the sign bits are not equal, the data overflows, and the saturation result should be +127 or -128 according to the most significant bit (MSB) of the 10-bit input. Otherwise, the saturation result is the low 8 bits of the 10-bit input.

The final result hard decision is implemented as shown in Figure 4 (d). The MSB of the sum of $R_{0,j}$ and $L_{0,j}$ can be taken as the final result directly according to Formula (5).

Table 1. The truth table of the selector in $f(x, y)$ module.

Conditions	Output
$x \geq y$, $x \geq -y$	y
$x \geq y$, $x < -y$	$-x$
$x < y$, $x \geq -y$	x
$x < y$, $x < -y$	$-y$

3.3 Memory subsystem design

Figure 5 gives a data access example when the sliding window number $w = 0$. From Figure 5, we can see that the left column and the right column have different access patterns. During the BP decoding procedure, a column of data may be accessed as the right column or the left column of BCUs in different computational stages. This leads to potential memory confliction problems.

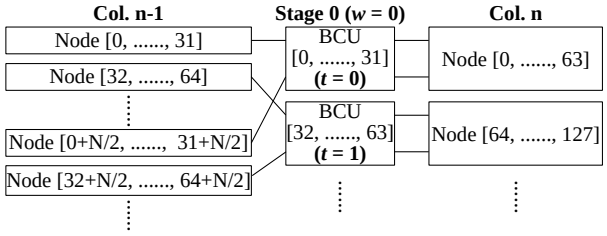

Figure 5. Two different memory access patterns (accessed as the left column or the right column).

To avoid memory confliction, the memory management should satisfy several requirements. Firstly, two neighbouring columns can be accessed at the same time. Secondly, nodes [a, ……, a+31] and nodes [a+N/2, ……, a+N/2+31] in one column can be accessed at the same time (a = 32m, 0 ≤ m ≤ N/64). Thirdly, nodes [b, ……, b+63] in one column can be accessed at the same time (b = 64m, 0 ≤ m ≤ N/64).

According to the requirements mentioned above, we propose a confliction-free memory management scheme that is applicable to any code length N. We allocate the data nodes at fixed positions in the memories, and shuffle the nodes using permutation networks between memories and SISOs.

N nodes in one column are separated by 16 PWs, and PW p only saves nodes [Np/16, ……, N(p+1)/16-1] in each column (0 ≤ p ≤ 15). All PWs share the same data allocation pattern. Figure 6 gives a data allocation example in PW 0 when N = 4096. Each column takes up only 4 memory banks. The columns linked by a line are accessed together in the SW marked on the line. The data allocation pattern of each column is also the same. Figure 6 shows the data arrangement inside the Col.12 as an example. The first half of nodes is arranged simply in order. Other nodes are arranged in a shifted order. The nodes filled with the same colour are accessed in the same clock cycle when used as the left column. The nodes with the same memory address are accessed in the same clock cycle when used as the right column.

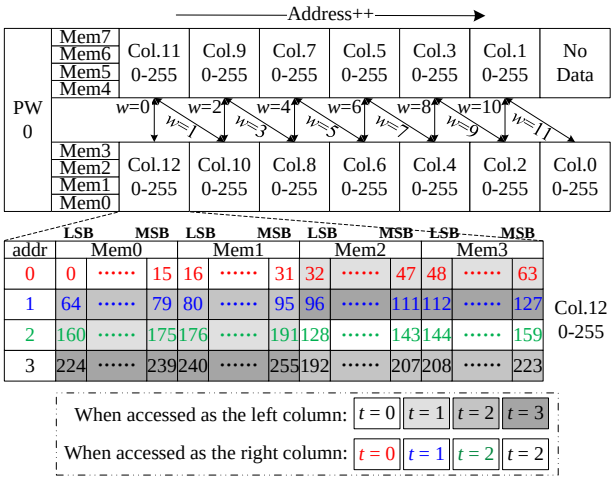

Figure 6. Data allocation example for PW 0 when N = 4096 (w and t stand for the corresponding sliding window number and the sliding window address at which the data are accessed).

As the data nodes are allocated at fixed positions in the memories, the read address and the write address of one node are actually the same. From Figure 6, we can see that the memory access addresses are related to the corresponding sliding window number w and the sliding window address t. Table 2 summarizes the address calculation methods for memory banks in one PW under different conditions, wherein *ad1*, *ad2*, *ad3*, and *ad4* are calculated as Formula (6) to (9). As all PWs share the same data allocation pattern, four access addresses are enough to access 128 memory banks at a time.

Table 2. Access address calculation for one parallel window under different conditions.

Bank number	0-1	2-3	4-5	6-7
LSB of w = 0				
LSB of t = 0	*ad*1+t	*ad*1+t	*ad*1+*ad*3	*ad*1+*ad*4
LSB of t = 1	*ad*1+t	*ad*1+t	*ad*1+*ad*4	*ad*1+*ad*3
LSB of w = 1				
LSB of t = 0	*ad*2+*ad*3	*ad*2+*ad*4	*ad*1+t	*ad*1+t
LSB of t = 1	*ad*2+*ad*4	*ad*2+*ad*3	*ad*1+t	*ad*1+t

$$ad1 = (w >> 1) \times SWL \tag{6}$$

$$ad2 = ad1 + SWL \tag{7}$$

$$ad3 = t >> 1 \tag{8}$$

$$ad4 = ad3 + (SWL >> 1) \tag{9}$$

The permutation networks are controlled by a 3-bit signal, and there are 8 different permutation patterns. The value and the meaning of each bit in the control signal are shown in Table 3. Figure 7 gives a permutation example when w = 0, t = 0, and N = 4096. When write back intermediate results, only half of the banks are updated under the control of the write enable signal "*wen*". The left column is updated during LP, and RP updates the right column.

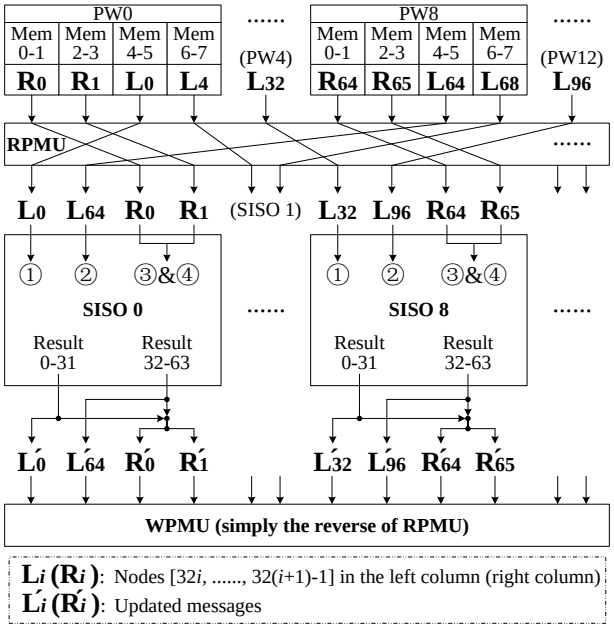

Figure 7. Data shuffling operation example when $w = 0$, $t = 0$, and $N = 4096$.

Table 3. 3-bit permutation control signal.

Bit	Value	Meaning
0	$t \geq (SWL>>1)$?	0: right column in order; 1: right column in the shifted order.
1	LSB of t	0: left column in order; 1: left column in the shifted order.
2	LSB of w	0: right column in bank 0-3; 1: right column in bank 4-7.

3.4 Control path and programmability design

The programmability of PASIP is realized by the control path hardware and a specially designed instruction set.

The program memory saves instructions, and instructions can be decoded by the instruction decoder into control signals to control other hardware modules. The instruction set consists of many kinds of instructions, such as system instructions (like RESET, NOP), move instructions (in charge of data exchange among registers, memories, and outer storages), and most importantly, the Polar decoding acceleration instructions.

From the decoding process shown in Figure 1, we extract three Polar decoding acceleration instructions. The instruction names and the corresponding functions are listed in Table 4. These three instructions introduce two control signals, "aw" and "as", to control the sliding window number w and the sliding window address t separately. There are four operators for "aw" and "as": "++" means increasing by 1, "--" means decreasing by 1, "clr" means setting to 0, and "set" means setting to the maximum value. With these instructions and control signals, PASIP is able to provide a fully programmable decoding procedure as shown in Figure 3.

Table 4. Polar decoding acceleration instructions.

Instruction name	Function
PLEFT	One step of right-to-left propagation
PRIGHT	One step of left-to-right propagation
PFIN	The final step of left-to-right propagation & Hard decision for final results

The control path also contains a parameter register file. The decoding settings and the code configurations, including the total stage number of the factor graph n, the code length N, the parallel window length, the SWL and the hardware loop settings, are saved in the parameter registers. The support for decoding with multiple code lengths is realized by adjusting these parameter registers using move instructions.

Figure 8 shows a slice of assembly program for Polar BP decoding when $SWL = 0$, where "rep = xx" makes the instruction self-repeat for "xx" times.

```
……
//hwfor (i = 0; i <max_ite-2; i ++)
//{
      //right-to-left propagation
      PLEFT, clras, clraw
      PLEFT, clras, ++aw, rep=n-2
      //left-to-right propagation
      PRIGHT, clras, setaw
      PRIGHT, clras, --aw, c0--, rep=n-2
//}
// last iteration
PLEFT, clras, clraw
PLEFT, clras, ++aw, rep=n-2
PFIN, clras, setaw
……
```

Figure 8. A slice of assembly program ($SWL = 0$).

4 Performance analysis

Table 5. The throughputs of Polar decoding with all supported code lengths (when $P = 16$, $k = 32$ and 128-bit×32 memories are used).

Supported code lengths (bit)	TP (Mbps)
64	175.34
128	292.57
256	501.96
512	878.97
1024	1563.36
2048	1407.56
4096	1280

One normal iteration of BP decoding computation consists of n-1 times of LP and n-1 times of RP, whereas the last iteration of decoding consists of n times of LP ($n = \log_2 N$). SWL clock cycles are needed

for each time of LP or RP, and thus the decoding procedure costs $[(\log_2 N\text{-}1) \times 2 \times (max\_ite\text{-}1)+\log_2 N] \times SWL$ clock cycles in total, where *max_ite* means the maximum iteration number. And the throughput of PASIP can be calculated as $f \times N \div total\ clock\ cycles$, where f is the work frequency. In this paper, we adopt *max_ite* = 15. The synthesis result shows that the maximum work frequency of PASIP is 400MHz. And the throughputs (*TP*s) for all supported code lengths are listed in Table 5.

We can see that the performance of PASIP shows two phases. From Subsection 3.2, we know that PASIP can handle at most 512 BCUs every clock cycle. When $N \le 1024$, one time of propagation is finished within one clock cycle so that $SWL = 1$. The characteristic of this phase can be summarized as Formula (10). In this phase, *TP* increases as N increases. The maximum *TP* is reached when $N = 1024$. When $N > 1024$, SWL can be calculated as $N/1024$, which is always larger than 1. The decoding throughput in this phase can be calculated as Formula (11), and thus *TP* decreases as N increases.

$$TP = (f \times N) / \left[(\log_2 N - 1) \times 2 \times 14 + \log_2 N \right] \quad (10)$$

$$TP = (f \times 1024) / \left[(\log_2 N - 1) \times 2 \times 14 + \log_2 N \right] \quad (11)$$

The performance and the supported code length range mentioned above are only the results when PASIP adopts $P = 16$, $k = 32$, and 128-bit × 32 memories. PASIP is scalable. The supported minimum code length is $2k$. The supported maximum code length is determined by the depth of the memories. The supported code length range can be extended by decreasing k or simply increasing the depth of memories. The maximum throughput of PASIP is determined by Pk. PASIP can be capable of satisfying higher throughput requirements in future high speed communication standards by increasing P or k.

5 Synthesis results and comparison

PASIP is synthesized by Synopsys Design Compiler. The 65nm CMOS technology is used in this paper.

The synthesis results show that the total area is 2.71 mm^2 when PASIP adopts $P = 16$ and $k = 32$, wherein the memory subsystem takes up 74.67%, SISO modules take up 19.43%, and the control path only takes up 5.90%.

Table 6 compares PASIP with state-of-art Polar decoders, wherein all areas, frequencies, and throughputs are normalized to 65nm process.

Compared with the multi-code SC Polar decoder proposed by Coppolino *et al.* [9], PASIP has a smaller supported code length range, and occupies a larger total area mainly because of the larger memory area consumption. However, PASIP provides much higher throughputs and better area efficiency than this multi-code SC Polar decoder due to the high parallelism.

PASIP is also compared with other BP Polar decoders. As far as we know, there are no other CMOS implementations of multi-code BP Polar decoders in literature until June 2018. Therefore, PASIP is compared with the single-code BP Polar decoders [5, 10] instead.

PASIP occupies larger total area and has lower area efficiency than the single-code decoders supporting $N = 1024$. This is mainly because PASIP introduces a larger memory area for supporting larger code lengths (2048 and 4096). On the other hand, PASIP achieves much higher flexibility than the single-code BP decoders. PASIP supports multiple code lengths (from 64 to 4096) with only little hardware overhead compared with the single-code decoder [5] that only supports $N = 4096$.

Table 6. Results comparison.

Decoder	This work (when P = 16 & k = 32)	[9]	[10]	[5]		
Algorithm	BP	SC	BP	BP		
Process (nm)	65	65	40	45		
***N*(bits)**	64 to 4096	2 to 4096	1024	1024	4096	1024+4096[6]
Total Area (mm^2)	2.71	2.01	1.859[1]	1.56[1]	2.57[1]	4.13
Memory Area (mm^2)	1.24	0.57	-	0.25[1]	1.27[1]	1.52
Frequency (MHz)	400	1060	308[2]	136[2]	136[2]	136
Throughput (Mbps)	1563 (N = 1024) 1280 (N = 4096)	352 (N = 1024) 348 (N = 4096)	2334[3,4]	1165[3]	932[3]	1165 (N = 1024) 932 (N = 4096)
NAE[5] (Mbps/mm^2)	577 (N = 1024) 472 (N = 4096)	175 (N = 1024) 173 (N = 4096)	1256	747	363	282 (N = 1024) 226 (N = 4096)

1: Normalized area, scale factor = $(65/\text{Process})^2$. 2: Normalized frequency, scale factor = Process/65. 3: Normalized throughputs, scale factor = Process/65. 4: Throughputs of BP decoders are normalized to throughput @ 15 iterations. 5: Normalized area efficiency, calculated by normalized throughput/normalized total area. 6: Sum of multiple single-code BP decoders, normalized to 65 nm process.

PASIP achieves much smaller total area than the sum of multiple single-code decoders, and thus achieves better area efficiency.

6 Conclusion

In this paper, we propose a scalable Polar decoding ASIP in 65nm CMOS technology, namely PASIP. The SIMD architecture and the specially designed Polar decoding acceleration instructions ensure the high performance as well as the programmability for PASIP. PASIP supports multiple code lengths, from 64 to 4096, with a total area of 2.71 mm^2. PASIP provides a maximum throughput of 1563 Mbps at the maximum frequency of 400 MHz when $N = 1024$. PASIP achieves higher flexibility and higher area efficiency than the sum of multiple single-code BP Polar decoders. And thus, PASIP is more practical for future base stations and terminals.

Acknowledgement

This work was supported by National High Technical Research and Development Program of China (863 program) 2014AA01A705.

References

1. E. Arikan, IEEE Trans. on Info. Theory, **55**, 3051 (2009).
2. 3GPP, Final Report of 3GPP TSG RAN WG1 #87 v1.0.0 (2016).
3. B. Yuan and K. K. Parhi, IEEE Trans. on Signal Processing, **62**, 6496 (2014).
4. Y. S. Park, Y. Tao and S. Sun, *Symposium on VLSI Circuits of Technical Papers*, 1 (2014).
5. J. Sha, X. Liu and Z. Wang, China Commun., **12**, 34 (2015).
6. 3GPP 36.212-200, Technical Specification Group Radio Access Network; NR; Multiplexing and channel coding (Release 15, 2018).
7. A. Pamuk, *Interational Symposium on Wireless Communication Systems*, 437 (2011).
8. D. Liu, *Embedded DSP Processor Design: Application Specific Instruction Set Processors* (Morgan Kaufmann, 2008).
9. G. Coppolino, C. Condo, G. Masera, W. J. Gross, IEEE Trans. Circuits Syst. I: Regular Papers, Early Access (2018).
10. Y. Chen, C. Cheng, T. Tsai, W. Sun, Y. Ueng, C. Yang, *2017 Symposium on VLSI Circuits*, C330 (2017).