A novel QPP interleaver for parallel turbo decoder

Wei Liu[a)], **Shuming Chen, Hu Chen, Yaohua Wang, Sheng Liu, Kai Zhang, and Xi Ning**

Computer School, National University of Defense Technology, #109, Deya Road, Changsha, 410073, China

a) *microweiliu@gmail.com*

Abstract: Quadratic permutation polynomial (QPP) interleaver is more suitable for parallel turbo decoding due to it is contention-free. However, the parallel address generation of QPP is area-consuming when the parallel degree P is large, and the data shuffle between memory banks and processing elements (PE) introduces large interconnect cost. This paper first evaluates the area and power cost of three typical Parallel Address Generators (PAG) and four typical Data Shuffle Networks (DSN) from academic and industrial area, and then proposes a novel general QPP interleaver with a highly area-efficient PAG and an associated DSN. Our QPP interleaver can support general parallel turbo decoder design. Experimental results show that, for P=64, the area and power cost of the PAG are on average 9.2% and 9.8% of that of the evaluated respectively. Meanwhile, the DSN can also achieve a slight hardware cost reduction, compared with the evaluated works.

Keywords: QPP, interleaver, parallel turbo decoder, shuffle network

Classification: Integrated circuits

References

[1] Y. Sun and J. R. Cavallaro, "Efficient hardware implementation of a highly-parallel 3GPP LTE/LTE-advance turbo decoder," *INTEGRATION, the VLSI Journal*, vol. 44, pp. 305–315, Jan. 2011.
[2] T. Ilnseher, M. May, and N. Wehn, "A monolithic LTE interleaver generator for highly parallel SMAP decoders," *Wireless Telecommunications Symposium (WTS)*, pp. 1–4, April 2011.
[3] S. J. Wang, J. Ma, G. H. He, and Z. G. Mao, "Generalized interleaving network based on configurable QPP architecture for parallel turbo decoder," *IEEE Signal Process. Syst. (SiPS)*, pp. 204–209, Oct. 2011.
[4] C. Studer, C. Benkeser, S. Belfanti, and Q. T. Huang, "Design and Implementation of a Parallel Turbo-Decoder ASIC for 3GPP-LTE," *IEEE J. Solid-State Circuits*, vol. 46, pp. 8–17, Jan. 2011.
[5] K. E. Batcher, "Sorting networks and their applications," *ACM Proc. Spring Joint Computer Conference*, pp. 307–314, April 1968.
[6] C. C. Wong, Y. Y. Lee, and H. C. Chang, "A 188-size 2.1mm^2 reconfigurable turbo decoder chip with parallel architecture for 3GPP LTE system," *Symposium on VLSI Circuits*, pp. 288–289, June 2009.

DOI: 10.1587/elex.10.20120795
Received October 15, 2012
Accepted December 18, 2012
Published April 25, 2013

1 Introduction

Turbo decoding is widely used in communication systems, like LTE, LTE-A. It is computation-intensive and usually implemented through parallel hardware architecture (Fig. 1). Quadratic permutation polynomial (QPP) interleaver is more suitable for parallel turbo decoding due to it is contention-free. However, the parallel address generation of QPP interleaver is area-consuming when the parallel degree P is large, and the data shuffle between memory banks and processing elements (PE) seriously restricts the frequency of QPP interleavers, thus the throughput of parallel turbo decoding.

To reduce the complexity of Parallel Address Generators (PAG), Y. Sun [1] proposed an on-the-fly forward and backward recursive QPP Interleaving Address Generator (QPP-IAG). However, the QPP-IAG must be replicated multiple times. The memory Bank Index Address (BIA) and Bank Offset Address (BOA) are calculated directly in T. Ilnseher's [2] work, which adopts only two Forward Address Generators (FAG) but a large number of multipliers. Work by S.J. Wang [3] reduces the number of multipliers but suffers from the big fan-out problem. Besides, all works above do not support the calculation of the initial values for those recursive processes in the PAG, and the works in [2, 3] do not support the backward consecutive address generation.

To reduce the complexity of the (de-)interleaving Data Shuffle Networks (DSN), C. Studer [4] proposed a master-slave network based on Batcher's sorter [5]. However, it substantially increases power consumption. S.J. Wang [3] proposed a network which reduces the scale of C. Studer's [4] network to the slave network. Work by C.C. Wong [6] adopted a barrel shifter network with its control signals generated by $P/2$ adders that cause extra wire latency.

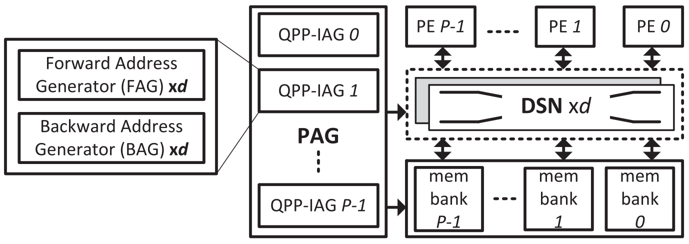

Fig. 1. general architecture of parallel turbo decoder

We proposed a novel PAG and an associated DSN. The PAG calculates the initial values, BOAs and BIAs in both forward and backward direction with step size d=1, 2, 4, ... Our PAG needs generating d BOAs and $2dP/4$ BIAs in parallel while the conventional PAGs require $2dP$ BIAs. The DSNs in [3, 4, 6] can not be used directly, thus our DSN is redesigned accordingly. And, one DSN can shuffle P data determined by our PAG's $P/4$ outputs.

2 Theories of the novel QPP interleaver

In order to derive the theories of our QPP interleaver, the original fundamentals are briefly described first. The QPP addresses $f(x)$ can be generated in the forward and backward recursive directions [1] using Eq. (1)∼Eq. (3). The works in [2, 3] calculate the BIA ($b_j(c)$) and BOA ($a_j(c)$) directly by Eq. (4).

$$f(x) = f_1 x + f_2 x^2 = f(c + jW) = b_j(c) \times W + a_j(c) \tag{1}$$

$$f(x + d) = (f(x) + g(x))\%K, g(x) = (g(x - d) + (2d^2 f_2)\%K)\%K \tag{2}$$

© IEICE 2013
DOI: 10.1587/elex.10.20120795
Received October 15, 2012
Accepted December 18, 2012
Published April 25, 2013

$$f(x-d) = (f(x) - g(x-d))\%K, g(x-d) = (g(x) - (2d^2 f_2)\%K)\%K \quad (3)$$

$$a_j(c) = a_0(c), b_j(c) = (b_0(c) + f_1 j + f_2 j^2 W + 2f_2 jc)\%P \quad (4)$$

where $0 \le x, f_1, f_2 < K$, $0 \le j < P$, $0 \le c < W = K/P$. Variables K, W, P denote the block length, the sub-block length and the parallelism respectively.

Eq. (4) shows that only the first BOA (i.e. $a_0(c)$) is needed to access all individual memory banks, we also adopt Eq. (4) to generate BOA. However, our BIA generation method is different. The works in [1, 2, 3] generate P BIAs to control one DSN while our method only generates $P/4$ BIAs. In the rest of this section, we first prove that our DSN can be controlled only by $P/4$ BIAs in Theorem 1, then we formulate our forward and backward BIA generation method with d=1, 2, 4, ... in Theorem 2.

Theorem 1: For the QPP interleaver, $|b_{j+iQ}(c) - b_j(c)| \in \{Q, 2Q, 3Q\}$, where $Q = P/4$, $0 \le j < Q$ and i=1,2,3. And our DSN can shuffle P data at each clock cycle only determined by $P/4$ BIAs, i.e. $b_{j=0\sim Q-1}(c)$.

Proof: Assume $0 \le j < Q$, i=1,2,3. According to Eq. (4), we can get

$$\begin{aligned} |b_{j+iQ}(c) - b_j(c)| &= (iQf_1 + f_2(2ijQW + i^2Q^2W + 2iQc))\%P \\ &= (iQf_1 + (f_2/2)P(ijW + i^2(PW/8) + ic))\%P \end{aligned} \quad (5)$$

Since in QPP, f_1 is an odd number, f_2 is an even number, K is multiple of 8, then $|b_{j+iQ}(c) - b_j(c)| = (iQf_1)\%P$. Let $f_1 = 2t + 1$, when i=1,2,3, then

$$\begin{aligned} |b_{j+Q}(c) - b_j(c)| &= (Q(2t+1))\%P = [2(t\%2) + 1]Q \in \{Q, 3Q\} \\ |b_{j+2Q}(c) - b_j(c)| &= (2Q(2t+1))\%P = 2Q \\ |b_{j+3Q}(c) - b_j(c)| &= (3Q(2t+1))\%P = [3 - 2(t\%2)]Q \in \{3Q, Q\} \end{aligned} \quad (6)$$

From the above all, $|b_{j+iQ}(c) - b_j(c)| \in \{Q, 2Q, 3Q\}$. Thus our PAG only need to output $b_{j=0\sim Q-1}(c)$ at each clock cycle, if the DSN can be controlled only by them. Then, we will prove the feasibility of our DSN.

At the c^{th} clock cycle, according to Eq. (6), we can get

$$b_{j+Q\times(i-1)}(c)\%Q = b_{j+Q\times i}(c)\%Q \quad (7)$$

$$(b_{j+Q\times(i-1)}(c)/Q)\%4 = (b_{j+Q\times i}(c)/Q + 1)\%4 \quad (8)$$

Let us denote the data read from P memory banks as $d_j(c)$, $d_{j+Q}(c)$, $d_{j+2Q}(c)$ and $d_{j+3Q}(c)$. Eq. (7) indicates that these data can be packed as $D_j(c) = \{d_j(c), d_{j+Q+2Q\cdot(t\%2)}(c), d_{j+2Q}(c), d_{j+3Q-2Q\cdot(t\%2)}(c)\}^*$ and shuffled simultaneously. $\{\}^*$ denotes the concatenation operator. Shuffling the $D_j(c)$ is determined by $b_j(c)\%Q$. Then, Eq. (8) shows that $D_j(c)$ need to be de-packed and dispatched uniquely to the associate PEs. Such process can be implemented through barrel shifters which are determined by $b_j(c)/Q$.

So according to Eq. (5)$\sim$Eq. (8), our DSN can shuffle P data only determined by $P/4$ BIAs, i.e. $b_{j=0\sim Q-1}(c)$. Next, we formulate the calculation of $b_{j=0\sim Q-1}(c)$ in both forward($'+'$) and backward direction($'-'$) in Theorem 2.

Theorem 2: The j^{th} BIA at c^{th} clock cycle can be generated recursively by $b_j(c \pm d) = (b_j(c) + \Psi_0(c) \pm 2df_2 j)\%P$, where $0 \le j < P/4, 0 \le c < W$.

Proof: Assume $0 \le c < W$, according to Eq. (1), we can get

$$\begin{aligned} b_j(c \pm d) &= \left\lfloor \frac{f(jW+c\pm d)}{W} \right\rfloor = \left\lfloor \frac{(f_1(jW+c\pm d) + f_2(jW+c\pm d)^2)\%K}{W} \right\rfloor \\ &= \left\lfloor \frac{((f_1(jW+c) + f_2(jW+c)^2) + (\pm f_1 d \pm 2cdf_2 + f_2 d^2) \pm 2jdf_2 W)\%K}{W} \right\rfloor \\ &= \left\lfloor (\frac{f(jW+c) - a_0(c)}{W} + \frac{a_0(c) + (\pm f_1 d \pm 2cdf_2 + f_2 d^2)}{W} \pm 2jdf_2) \right\rfloor \%P \\ &= (b_j(c) + \Psi_0(c) \pm 2jdf_2)\%P \end{aligned} \quad (9)$$

© IEICE 2013
DOI: 10.1587/elex.10.20120795
Received October 15, 2012
Accepted December 18, 2012
Published April 25, 2013

Next we derive $\Psi_0(c)$. According to Eq. (2), $(f_1d+2cdf_2+f_2d^2)\%K = g(c)$ and $g(c) = f(c+d) - f(c)$. According to Eq. (3), $(-f_1d-2cdf_2+f_2d^2)\%K = -g(c-d)$, and $-g(c-d) = f(c-d) - f(c)$, so

$$\begin{aligned}\Psi_0(c) &= \left\lfloor \frac{(a_0(c)+(\pm f_1d \pm 2cdf_2+f_2d^2))}{W} \right\rfloor \%P = \left\lfloor \frac{(a_0(c)+f(c\pm d)-f(c))\%K}{W} \right\rfloor \\ &= \left\lfloor \frac{((f(c\pm d)-a_0(c\pm d))+a_0(c\pm d)-(f(c)-a_0(c)))\%K}{W} \right\rfloor \\ &= \left(\frac{f(c\pm d)-a_0(c\pm d)}{W} - \frac{f(c)-a_0(c)}{W}\right)\%P = (b_0(c\pm d) - b_0(c))\%P\end{aligned} \tag{10}$$

We can get $\Psi_0(c)$ by combining Eq. (10)~Eq. (12). The detailed derivation of Eq. (11) can be referenced in [3]. Then we derive Eq. (12) analogously. $O1|2$ and $O1'|2'$ are the byproducts of FAG and BAG(Fig. 1) respectively.

$$b_0(c+d) - b_0(c) = O1 + I(c), I(c) = (I(c-d) + \lfloor 2d^2f_2/W \rfloor + O2)\%P \tag{11}$$

$$b_0(c-d) - b_0(c) = O1' - I(c-d), I(c-d) = (I(c) - \lfloor 2d^2f_2/W \rfloor + O2')\%P \tag{12}$$

Since Eq. (9) is a recursive process, $P/4$ forward initial values $b_{j=0\sim Q-1}(c=0)$ and backward initial values $b_{j=0\sim Q-1}(c=W)$ must be pre-computed. According to Eq. (4), these initial values can be got recursively using Eq. (13).

$$\begin{aligned} b_{j+1}(c=0) &= (b_j(c=0) + (f_1+f_2W) + 2f_2Wj)\%P \\ b_{j+1}(c=W) &= (b_j(c=W) + (f_1+f_2W) + 2f_2W(j+1))\%P \end{aligned} \tag{13}$$

The recursively computing pattern of Eq. (9) is similar to Eq. (13) in both forward and backward direction. The patten can be unified as $b = (b' + C1 \pm C2)\%P$. So, Eq. (9) and Eq. (13) can be implemented using one circuit by designing reuse. The circuit generates $P/4$ initial values and $P/4$ BIAs.

3 Implementation of the novel QPP interleaver

The proposed QPP interleaver is consisted of d PAGs and $2d$ DSNs. As shown in Fig. 2 (b), one PAG is consisted of a QPP-IAG, a Forward and a Backward Chained Address Generator (FCAG,BCAG). The QPP-IAG is responsible for generating the BOA to access memory banks in parallel(Eq. (4)), as well as generating $O1|2$ to FCAG(Eq. (11)) and $O1'|2'$ to BCAG(Eq. (12)).

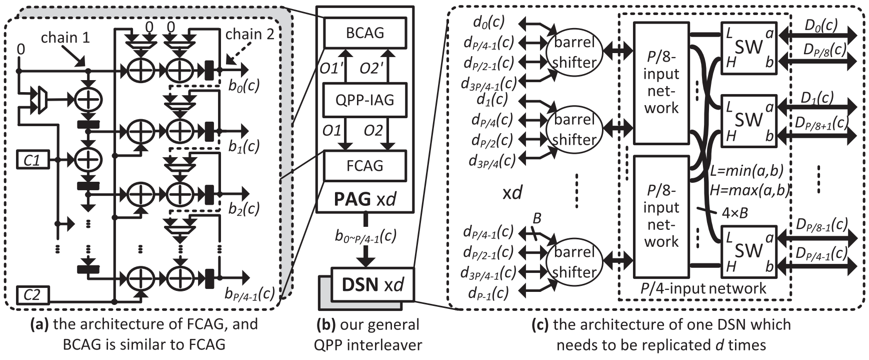

Fig. 2. the proposed QPP interleaver

In Fig. 2 (a), our FCAG generates $P/4$ forward initial values and $P/4$ BIAs in the forward direction to control a DSN. The BCAG is similar to the FCAG, while the BCAG adopts some modulo subtractors. If the FCAG and BCAG are active synchronously, our QPP interleaver needs $2d$ same DSNs which are controlled by the FCAG and BCAG group by group, as shown in Fig. 2 (b). Otherwise, only d DSNs are needed and they are controlled by time-sharing. The DSNs shuffle $2dP$ data between the PEs and memory banks in parallel. Since the scale of our PAG is reduced, our DSN (Fig. 2 (c)) needs to be redesigned accordingly.

© IEICE 2013
DOI: 10.1587/elex.10.20120795
Received October 15, 2012
Accepted December 18, 2012
Published April 25, 2013

As shown in Fig. 2 (a), The FCAG requires three operation steps to generate initial values in a pipeline manner and the BIAs in parallel. Step **I**). chain 1 (vertical direction with adders and registers) and chain 2 (dot line, vertical direction with MUXes, adders and registers) are both active in a pipelined manner to generate initial values. After $P/4$ cycles, the initial values are generated and stored in the registers. Step **II**). only chain 1 is active. The constants $(2jdf_2)\%P$ are calculated in a pipelined manner, which are used in Eq. (9). Step **III**). chain 1 and chain 2 are both idle. $P/4$ BIAs are generated recursively in parallel by switching all the MUXes properly. The registers in chain 2 are accumulated. All the adders in Fig. 2 (a) are modulo adders.

As shown in Fig. 2 (c), the DSN consists of $P/4$ barrel shifters and a $P/4$-input network. The $P/4$-input network can be constructed by two $P/8$-input networks and $P/8$ SWitchers (SW) according to the iterative rule [5]. The basic element of the $P/8$-input network is also the SWitcher which simply compare the value of input a and b. The highest two bits of BIAs $(b_j(c)/Q)$ are used to control the barrel shifter and the remnant bits $(b_j(c)\%Q)$ are used to control the $P/4$-input network. The width of the barrel shifter is B, while the width of the $P/4$-input network is $4B$ as it shuffles the packed data.

4 Experimental results

In order to evaluate the hardware complexity of our QPP interleaver, we firstly theoretically analyze the hardware components of them using a scaling factor with respect to P and d. Table I and Table II show the analyzed results.

Table I. hardware components of the PAGs ($\times 2d$)

	multiplier		FAG\BAG	log_2P bit	log_2P+1	2-1	P-1
	num.	width	(ref. [1])	register	bit add	MUX	MUX
[1]	P	13×6	$3P/2+2$	$2P$	P†	0	1
[2]	$P/4$	‡	2	$9P/4$+2	$9P/4$+2	0	1
[3]	2	8×6	1	$2P$+4	$3P$+6	0	1
proposed	0	0	1	$P/2$+1	$3P/4$+2	$P/4$+4	0

†: the bit width of P adders in [1] is 14. ‡: $(log_2P-2)\times(log_2P-2)$. The work in [2] also adopts $4d$ 8×6 multipliers and $2d$ 13×6 multipliers.

Table II. hardware components of the DSNs ($\times 2d$)

	log_2P+1 bit add	2-1 MUX	barrel shifter	P-1 MUX
crossbar	0	0	0	P
[3]	0	$Plog_2P$	0	0
[4]	$P/2log_2P$	$2Plog_2P$	0	0
[6]	$P/2$	$Plog_2P$	0	0
proposed	0	$P/4log_2(P/4)$	$P/4$	0

"crossbar"denotes the traditional fully-connected shuffle network.

The hardware cost is in proportion to d. As shown in Table I, our PAG needs no multipliers, and the number of registers is a quarter of that of [3] which is the least number among all the evaluated works. Our PAG utilizes the methodology of designing reuse, so it will introduce 2-1 MUXes inevitably. However, the MUXes introduces trivial hardware overhead. Table II shows that the proposed DSN needs no adders and the number of needed MUXes is a quarter of that of [3, 6]. The introduced barrel shifters in our DSN result in trivial hardware overhead either (as shown in Fig. 3 (c)).

© IEICE 2013
DOI: 10.1587/elex.10.20120795
Received October 15, 2012
Accepted December 18, 2012
Published April 25, 2013

We also implement the proposed QPP interleaver and the evaluated works with P=8, 16, 32 and 64. The implementations are synthesized under a 65 nm TSMC CMOS process with timing budget of 0.6 ns. Then normalization is done through the division of the area and power cost by each peak value. The normalized performance is shown in Fig. 3.

As shown in Fig. 3 (a)∼(b), our PAG consumes much smaller area and power cost. For P=64, Our PAG consumes 4%, 24.4% and 27.8% area cost (9.2 % on average) and 4.4%, 28.6% and 22.2% power cost (9.8 % on average) respectively, as compared to [1, 2, 3]. The hardware cost of Sun's [1] work increases sharply with the growth of P. Ilnseher's [2] and Wang's [3] work exhibit the similar trend, while the hardware cost of our PAG increases slowly. Because our PAG only needs generating $2dP/4$ BIAs while the other PAGs require $2dP$ BIAs. Compared to Wang's [3] work, the increasing trend of Ilnseher's [2] area cost is sharper while the power cost is slower. Because the former uses less multipliers, however, it adopts more recursive processes which causes higher transition frequency and higher dynamic power.

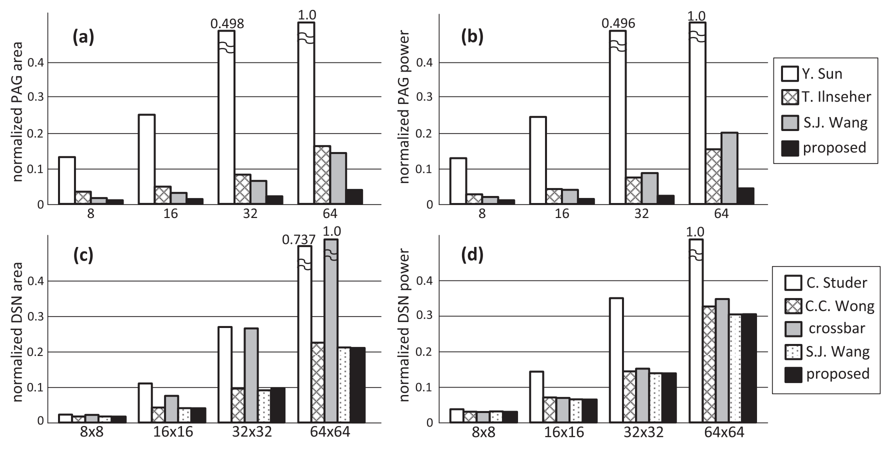

Fig. 3. The relative area cost and power consumption

As shown in Fig. 3 (c), for P=64, our DSN consumes 21.0% area cost to the crossbar and 28.5%, 93.3% and 99.2% respectively, compared to [3, 4, 6]. Since the crossbar has the capacity of broadcasting which is redundant for QPP interleaver, so, it consumes the highest area cost. Studer's [4] DSN includes a mass of adders, which cause extra area and power overhead. As shown in Fig. 3 (d), Studer's [4] DSN consumes higher power cost, while ours DSN consumes the least. These results are consistent with Table II.

5 Conclusion

This paper proposes a novel QPP interleaver which supports both forward and backward operation with step size d=1,2,4 .... The parallel address generator of our interleaver only generates a quarter of memory bank index addresses of that of the compared works. For the parallel degree P=64, the area and power cost of our parallel address generator are on average 9.2% and 9.8% of that of the compared respectively. And our data shuffle network is redesigned accordingly, which also contributes to hardware cost reduction.

Acknowledgments

This work is supported by NSF of China (60906014).

© IEICE 2013
DOI: 10.1587/elex.10.20120795
Received October 15, 2012
Accepted December 18, 2012
Published April 25, 2013