2010 13th Euromicro Conference on Digital System Design: Architectures, Methods and Tools

Area-Efficient Multi-Moduli Squarers for RNS

D. Bakalis
Electronics Laboratory, Dept. of Physics
University of Patras
Patras, Greece
bakalis@physics.upatras.gr

H. T. Vergos
Dept. of Computer Engineering and Informatics
University of Patras
Patras, Greece
vergos@ceid.upatras.gr

***Abstract*—Multi-moduli architectures are very useful for reconfigurable digital processors and fault-tolerant systems that are based on the Residue Number System (RNS). In this paper we propose two architectures for multi-moduli squaring that support the most common moduli cases in RNS channels, that is, 2^n-1, 2^n and 2^n+1. The proposed architectures are based on the modified Booth encoding of the input operand for deriving the required partial products and on Dadda adder trees for their addition. Experimental results show that the proposed squarers offer significant savings in area compared to previous proposals while a small improvement in delay is achieved in most cases as well.**

Keywords- Modulo arithmetic; residue number system; modulo 2^n±1; modulo squarers

I. INTRODUCTION

The Residue Number System (RNS) is a number system commonly adopted for speeding up computations in digital signal processing, cryptography and telecommunication applications and for fault-tolerant computing [1] [2]. A non-positional RNS is defined by a set of L moduli, suppose $\{m_1, \ldots, m_L\}$ that are pair-wise relatively prime. Let $\langle A \rangle_M$ denote the modulo M residue of an integer A, that is, the least non-negative remainder of the division of A by M. A has a unique representation in the RNS, given by the set $\{a_1, \ldots, a_L\}$ of residues, where $a_i = \langle A \rangle_{m_i}$.

The three-moduli set $\{2^n\text{-}1,\ 2^n,\ 2^n\text{+}1\}$ is the most commonly used in RNS, given apart from the fast implementations of the arithmetic operations, fast converters between the residue and the binary representation. In this moduli set, the 2^n+1 channel has to deal with operands one bit wider than the other two, leading to a performance bottleneck. The diminished-one representation was introduced to face this problem [3]. In the diminished-one representation each number is represented decreased by one compared to its normal representation and all arithmetic operations are inhibited for a zero operand. The diminished-one representation has the advantage that it requires only n bits, allowing to better equalize the delay of the three channels.

Reconfigurable computing for RNS-based systems has recently gained a significant interest. Multi-moduli architectures [4], that is, architectures for arithmetic circuits that support more than one modulo cases are very useful building blocks since they can be exploited for hardware reuse and offer substantial area savings. Multi-moduli architectures can be applied in reconfigurable RNS-based digital signal processors for providing flexibility and for easing the customization of the desired dynamic range and in fault-tolerant RNS-based systems for reducing the hardware costs of the redundant RNS channels. To this end, several reconfigurable multi-moduli architectures for the $\{2^n\text{-}1,\ 2^n,\ 2^n\text{+}1\}$ moduli set have been presented recently [5]-[8].

In this paper we present efficient multi-moduli squarers for the modulo 2^n-1, 2^n and 2^n+1 cases, assuming both the normal and the diminished-one representation for the latter case. The proposed architectures are based on the modified Booth algorithm for generating the partial products and on Dadda adder trees for their addition.

The remaining of the paper is organized as follows. Section II presents the partial product matrices that are required in the modulo squaring operation, for the various modulo cases. Efficient multi-moduli squarer architectures are introduced in Section III while section IV presents experimental results. Section V concludes the paper.

II. BOOTH-ENCODED MODULO SQUARERS

Every modulo squarer requires the partial products generation, their reduction in two summands and a final addition for deriving the result. In this section we concentrate on the partial products generation.

Let $A = a_{n-1}\ldots a_0$ be the n-bit input operand of a squarer. For simplicity, we consider in the following that n is even and use the n=8 case as an example. According to the radix-4 modified Booth encoding, we can rewrite A as $A = \sum_{i=0}^{n/2-1} 2^{2i} A_i$, where $A_i = (-2a_{2i+1} + a_{2i} + a_{2i-1}) \in [-2,+2]$ and a_{-1}=0. Thus, the square of A, A^2, can be computed using the Booth-encoded digits A_i as follows [9]:

$$A^2 = \left(\sum_{i=0}^{n/2-1} A_i \right)^2 = \sum_{i=0}^{n/2-1} 2^{4i} C_i + \sum_{i=0}^{n/2-2} 2^{4i+3} P_i \qquad (1)$$

where $C_i = A_i \times A_i$ and $P_i = \sum_{k=i+1}^{n/2-1} 2^{2(k-i-1)} A_i \times A_k$. The C_i terms, $0 \le i < n/2$, are unsigned numbers that can be represented by three bits since they assume only the values 0, 1 or 4. However, since the middle bit is always equal to 0, there is no need to be included in the partial product matrix.

978-0-7695-4171-6/10 $26.00 © 2010 IEEE
DOI 10.1109/DSD.2010.25

On the other hand, the P_i terms, are signed two's complement numbers that can easily be derived using simple circuits, such as those presented in [9]. Each P_i term, $0 \le i < n/2-1$, requires $(n-1-2i)$ bits. Let $C_{i,j}$ and $P_{i,j}$ denote the j-th bit of C_i and P_i, respectively. Let the most significant bit of each P_i term also be denoted as $P_{i,MSB}$.

Let us now focus on the modulo squaring operation. We consider the several moduli cases separately in the following.

(a) Modulo 2^n-1

Relation (1) can be used in the modulo 2^n-1 case as long as we take into account that $a_{-1}=a_{n-1}$ in the least significant Booth-encoded digit [10]. For attaining an n-bit wide matrix, we need to reposition all partial product bits with weights greater than 2^{n-1} to the columns with weights less than or equal to 2^{n-1}. For a bit z, it holds that $\left\langle z2^{n+i}\right\rangle_{2^n-1} = \left\langle z2^{i}\right\rangle_{2^n-1}$, $i \ge 0$. Hence, all $C_{i,j}$ and $P_{i,j}$ bits (except the $P_{i,MSB}$ bits) with weights greater than 2^{n-1} can be moved to columns with weights less than or equal to 2^{n-1}. The $P_{i,MSB}$ bits however need to be treated differently since they represent the sign of a two's complement P_i term and correspond to negative values. It can be easily shown that

$$\left\langle \sum_{i=0}^{n/2-2} -2^{(n+2i+1)} P_{i,MSB} \right\rangle_{2^n-1} = 11\overline{P}_{(n/2-2),MSB}1\cdots\overline{P}_{1,MSB}1\overline{P}_{0,MSB}1 .$$

Hence, we invert every $P_{i,MSB}$ bit, move it in a column with weight less than 2^{n-1} and also include some constant correction bits in the remaining columns. Fig. 1(a) presents the derived partial product matrix for the modulo 2^8-1 case. It consists of 31 partial product bits, 8 of which have constant values, and its maximum height among all columns is equal to 5.

(b) Modulo 2^n

Since it holds that $\left\langle z2^{n+i}\right\rangle_{2^n} = 0$, $i \ge 0$, for attaining an n-bit wide matrix, we can simply ignore all $C_{i,j}$ and $P_{i,j}$ partial product bits with weights greater than 2^{n-1}. No correction of any kind is required. Fig. 1(b) presents the corresponding partial product matrix for the modulo 2^8 case. 10 bits are required and the maximum height is equal to 2.

(c) Diminished-one Modulo 2^n+1

In the diminished-one modulo 2^n+1 case, the least significant Booth-encoded digit is derived using $a_{-1} = \overline{a}_{n-1}$ [11]. For a bit z, it holds that $\left\langle z2^{n+i}\right\rangle_{2^n+1} = \left\langle (-1)z2^{i}\right\rangle_{2^n+1} = \left\langle -2^i + \overline{z}2^i\right\rangle_{2^n+1}$, $0 \le i < n$, and $\left\langle z2^{2n}\right\rangle_{2^n+1} = \left\langle z2^{0}\right\rangle_{2^n+1}$. Hence all $C_{i,j}$ and $P_{i,j}$ bits (except the $P_{i,MSB}$) with weights greater than 2^{n-1} can be inverted and re-positioned in columns with weights less than or equal to 2^{n-1} as long as an additive constant correction term is taken into account for each such move. For the $P_{i,MSB}$ bits we have that:

$$\left\langle \sum_{i=0}^{n/2-2} -2^{(n+2i+1)} P_{i,MSB} \right\rangle_{2^n+1} = \left\langle \sum_{i=0}^{n/2-2} 2^{(2i+1)} P_{i,MSB} \right\rangle_{2^n+1} .$$

Hence, each $P_{i,MSB}$ can be simply moved to the corresponding column with weight less than or equal to 2^{n-1} and no further action is required.

The additive constant correction terms due to the partial product bit repositioning, along with the corresponding constant correction terms required for reducing the partial product bits in two summands and for deriving the final result with a diminished-one modulo 2^n+1 adder, can be merged into a single n-bit constant correction term. Its value can be easily shown to be $88\ldots8_{16}$ in the case of even values of n that are multiples of 4 and be equal to $22\ldots2_{16}$ in the case of even values of n that are not multiples of 4. Fig. 1(c) presents the derived partial product matrix for the diminished-one modulo 2^8+1 case. It also consists of 31 partial product bits, 8 of which have constant values, and the maximum column height is also equal to 5.

Modulo 2^n-1

2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
	$C_{1,2}$		$C_{1,0}$		$C_{0,2}$		$C_{0,0}$
	$C_{3,2}$		$C_{3,0}$		$C_{2,2}$		$C_{2,0}$
$P_{0,4}$	$P_{0,3}$	$P_{0,2}$	$P_{0,1}$	$P_{0,0}$		$\overline{P}_{0,6}$	$P_{0,5}$
$P_{1,0}$				$\overline{P}_{1,4}$	$P_{1,3}$	$P_{1,2}$	$P_{1,1}$
		$\overline{P}_{2,2}$	$P_{2,1}$	$P_{2,0}$			
1	1	0	1	0	1	0	1

(a)

Modulo 2^n

2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
	$C_{1,2}$		$C_{1,0}$		$C_{0,2}$		$C_{0,0}$
$P_{0,4}$	$P_{0,3}$	$P_{0,2}$	$P_{0,1}$	$P_{0,0}$			
$P_{1,0}$							

(b)

Diminished-one Modulo 2^n+1

2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
	$C_{1,2}$		$C_{1,0}$		$C_{0,2}$		$C_{0,0}$
	$\overline{C}_{3,2}$		$\overline{C}_{3,0}$		$\overline{C}_{2,2}$		$\overline{C}_{2,0}$
$P_{0,4}$	$P_{0,3}$	$P_{0,2}$	$P_{0,1}$	$P_{0,0}$		$P_{0,6}$	$\overline{P}_{0,5}$
$P_{1,0}$				$P_{1,4}$	$\overline{P}_{1,3}$	$\overline{P}_{1,2}$	$\overline{P}_{1,1}$
		$P_{2,2}$	$\overline{P}_{2,1}$	$\overline{P}_{2,0}$			
1	0	0	0	1	0	0	0

(c)

Normal Modulo 2^n+1

2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
	$C_{1,2}$		$C_{1,0}$		$C_{0,2}$		$C_{0,0}$
	$\overline{C}_{3,2}$		$\overline{C}_{3,0}$		$\overline{C}_{2,2}$		$\overline{C}_{2,0}$
$P_{0,4}$	$P_{0,3}$	$P_{0,2}$	$P_{0,1}$	$P_{0,0}$		$P_{0,6}$	$\overline{P}_{0,5}$
$P_{1,0}$				$P_{1,4}$	$\overline{P}_{1,3}$	$\overline{P}_{1,2}$	$\overline{P}_{1,1}$
		$P_{2,2}$	$\overline{P}_{2,1}$	$\overline{P}_{2,0}$			
$\overline{a}_6$	$\overline{a}_5$	$\overline{a}_4$	$\overline{a}_3$	$\overline{a}_2$	$\overline{a}_1$	$\overline{a}_0$	$a_7 \vee a_8$
1	0	0	0	1	0	1	0

(d)

Figure 1. Partial product matrices of Booth-encoded 8-bit modulo squarers

(d) Normal Modulo 2^n+1

We now consider the case of modulo 2^n+1 squarers for an (n+1)-bit operand $A=a_n a_{n-1}\ldots a_0 \in [0, 2^n]$ in the normal representation. At first we consider the $A<2^n$ case, that is, a_n=0. If the n least significant bits of A are driven to the modulo 2^n+1 squarer derived for the diminished-one case then this will compute the n least significant bits of $\left\langle (A+1)^2-1\right\rangle_{2^n+1}$. Hence, we can use an extra partial product equal to $\left\langle -2A\right\rangle_{2^n+1}$ in the diminished-one squarer and derive the n least significant bits of $\left\langle A^2\right\rangle_{2^n+1}$. $\left\langle -2A\right\rangle_{2^n+1}$ can be expressed as the n-bit vector $\overline{a}_{n-2}\overline{a}_{n-3}\cdots\overline{a}_0 a_{n-1}$, provided that an additional correction term equal to 3 is also taken into account. The most significant bit can be derived if a slightly modified diminished-one adder [12] is used as the final adder.

If $A=2^n$, then a_n=1 and all the rest bits are 0. Since in this case $\left\langle A^2\right\rangle_{2^n+1}=\left\langle 2^{2n}\right\rangle_{2^n+1}=1$, we can just position a_n at the column with weight 2^0 and logically OR it with a_{n-1} of the partial product added for $\left\langle -2A\right\rangle_{2^n+1}$. All required correction terms can be merged into a single n-bit vector, whose value is constant and equal to the value of the corresponding constant correction term in the diminished-one modulo 2^n+1 squarers case increased by 2. Fig. 1(d) presents the derived partial product matrix for the normal modulo 2^8+1 case ($\vee$ denotes the logical OR operation). 39 partial product bits are required in total, 8 of which have constant values, while the maximum column height is equal to 6.

III. CONFIGURABLE MULTI-MODULI SQUARERS

Based on Section II, two multi-moduli squarer architectures can be derived: (a) a 3-moduli squarer architecture for operands in modulo 2^n-1, 2^n or 2^n+1 assuming the diminished-one representation, and (b) a 4-moduli squarer architecture for operands in modulo 2^n-1, 2^n or 2^n+1 assuming both the diminished-one and the normal representation.

A. 3-Moduli Squarer Architecture

Let $A = a_{n-1}\ldots a_0$ denote the n-bit input operand. An n-bit Booth-encoded squarer is based on n $C_{i,j}$ bits and (n^2/4-1) $P_{i,j}$ bits. A 3-moduli squarer requires, besides the $C_{i,j}$ and $P_{i,j}$ bits, n more bits with constant values for the required corrections in modulo 2^n-1 and diminished-one modulo 2^n+1. Half of the $C_{i,j}$ bits and almost half ($\lfloor n/4 \rfloor (n-2\lfloor n/4 \rfloor -1)$) of the $P_{i,j}$ bits are modulo independent and hence can be used as is for all modulo cases. The remaining bits are modulo-dependent and therefore multiplexers have to be used for them. However, since some of the multiplexers inputs have constant or inverted/non-inverted values, simplifications can be made.

After the partial product bits are generated, they are reduced in two n-bit summands using a Dadda adder tree of n-bit wide Carry Save Adders (CSAs). In modulo 2^n-1, a carry, suppose z, at the most significant bit position of a CSA can be moved to the least significant bit position since $\left\langle z2^n\right\rangle_{2^n-1}=\left\langle z2^0\right\rangle_{2^n-1}$. Hence, End-Around-Carry (EAC) CSAs can be used. In modulo 2^n, a carry z can be ignored since $\left\langle z2^n\right\rangle_{2^n}=0$. In modulo 2^n+1, a carry z can be inverted and moved to the least significant bit position since $\left\langle z2^n\right\rangle_{2^n+1}=\left\langle -1+\bar{z}2^0\right\rangle_{2^n+1}$. Hence, inverted EAC CSAs can be used as long as a constant correction term equal to -1 is considered for every inverted EAC CSA. We note that these constant correction terms are already included in the total correction term that was presented in the previous section. Hence, in the proposed architecture, each carry at the most significant bit position of a CSA is driven to a multiplexer that produces the correct value at the least significant bit position according to the selected modulo value.

The two n-bit outputs of the adder tree are finally added using a two-input modulo adder. The final adder of the 3-moduli squarer architecture can be designed according to [8], that is, using a parallel-prefix structure to add the two input operands, a multiplexer to derive the appropriate value of the reentrant carry, an extra prefix level to take this carry into account and a level of XOR gates to produce the result.

Fig. 2 presents the proposed 3-moduli squarer architecture for the n=8 case. $a_{7\ldots0}$ denote the bits of the input operand A while f_1 and f_0 select the modulo value. $t_{7\ldots0}$ denote the constant correction bits for the three modulo cases. These bits can be derived by simple logic gates. $pp_{22\ldots0}$ denote the partial product bits that are derived based on the $C_{i,j}$ and $P_{i,j}$ bits of the Booth-encoded squarer. $pp_{22\ldots0}$ and $t_{7\ldots0}$ bits are driven to a CSA tree that reduces them in two 8-bit vectors which are then driven to the final adder to produce the result $r_{7\ldots0}$ of the required modulo squaring. The result is equal to A^2 taken modulo 2^n-1, 2^n or 2^n+1 (in the diminished-one representation), according to f_1 and f_0.

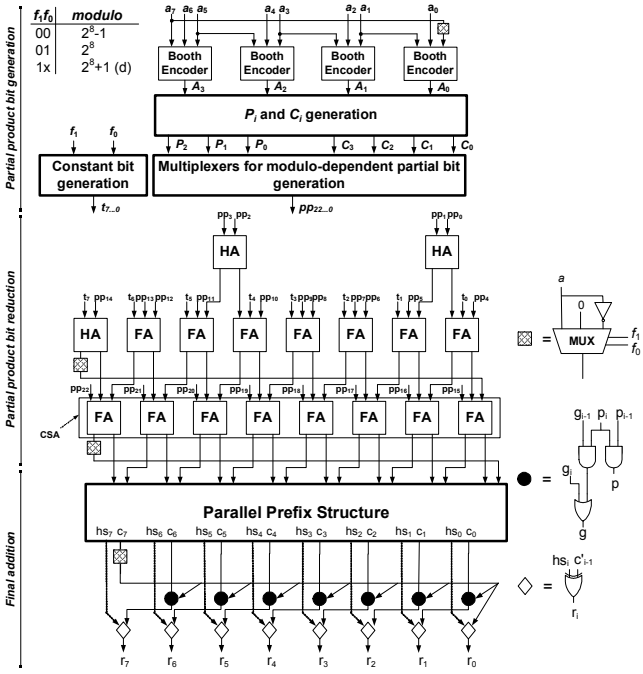

Figure 2. Proposed 3-moduli squarer for n=8

TABLE I. QUALITATIVE EVALUATION

	[8] 3-Mod Squarer		Proposed 3-Mod Squarer		Proposed 4-Mod Squarer	
n	***#bits***	***#MUXs***	***#bits***	***#MUXs***	***#bits***	***#MUXs***
8	44	20	31	14	39	14
12	90	42	59	27	71	27
16	152	72	95	44	111	44
20	230	110	139	65	159	65
32	560	272	319	152	351	152

B. 4-Moduli Squarer Architecture

A more generic multi-modulo squarer architecture can be derived if we include both normal and diminished-one representations in the modulo 2^n+1 case. Let $A = a_n \ldots a_0$ denote the input operand. The partial product matrix of the 4-moduli squarer architecture can be derived exactly as the corresponding of the 3-moduli one except that we have to add an extra n-bit partial product equal to $\langle -2A \rangle_{2^n+1}$ in the normal modulo 2^n+1 case and equal to 0 in the other 3 moduli cases. Hence, the Dadda adder tree that is used for adding the partial products must have one more CSA and the constant correction term in the diminished-one modulo 2^n+1 case must be decreased by one compared to its corresponding value in the case of the 3-moduli squarer architecture. The most significant bit of the result can be derived by the final adder as described in [12].

IV. EVALUATION AND COMPARISONS

The proposed architectures are compared against that of [8]. Table I presents, for several values of n, the total number of partial product bits and the number of multiplexers that are required in each case. Both proposed architectures require less bits and less multiplexers compared to that of [8]. Hence, the proposed squarers are expected to outperform those of [8] in area terms, even though the Booth-encoded partial product generation logic is more complex than the non-encoded one of [8].

In order to validate the conclusions drawn before, we described in HDL multi-moduli squarers for several values of n. In all cases, we considered that the final adder has a Kogge-Stone parallel prefix carry computation unit. After validating the correct operation of the HDL descriptions via simulation, we synthesized them in a standard cell 90nm CMOS technology, using a standard delay optimization script, and derived estimates for area and delay. The attained results, given in Table II, indicate that both the proposed squarers are significantly more area efficient than those of [8] while in most cases they also offer a faster implementation. Area reductions up to 31% are reported in the case of the 3-moduli squarer architecture and up to 26% in the case of the 4-moduli squarer architecture. The only exception is the case of n=8 where the proposed architectures are a little slower and in the case of the 4-moduli squarer architecture a little larger than the squarers of [8] since the complexity of the Booth-encoded partial product bit generation outperforms the reduction in the number of partial product bits and the reduction in the number of multiplexers that are required.

V. CONCLUSIONS

Two architectures for multi-moduli squaring have been presented. The first one supports operands in modulo 2^n-1, 2^n or 2^n+1 assuming the diminished-one representation while the second one supports operands in modulo 2^n-1, 2^n or 2^n+1 assuming both the diminished-one and the normal representation. The partial product bit generation has been based on the modified Booth encoding. Experimental data verified that the proposed architectures result in more efficient circuits than those of [8].

REFERENCES

[1] P. V. Ananda Mohan, Residue Number Systems: Algorithms and Architectures, Netherlands: Kluwer Academic Publishers, 2002.

[2] A. Omondi and B. Premkumar, Residue Number Systems: Theory and Implementation, London: Imperial College Press, 2007.

[3] L. Leibowitz, "A simplified binary arithmetic for the fermat number transform," IEEE Trans. Acoust., Speech, Signal Processing, vol. 24, no. 5, Oct. 1976, pp. 356-359.

[4] V. Paliouras and T. Stouraitis, "Multifunction architectures for RNS processors," IEEE Trans. on Circuits and Systems - II, vol. 46, no. 8, Aug. 1999, pp. 1041-1054.

[5] G. Jaberipur and B. Parhami, "Unified approach to the design of modulo-(2^n±1) adders based on signed-LSB representation of residues," Proc. IEEE Int. Symposium on Computer Arithmetic, 2009, pp. 57-64.

[6] C. –H. Chang, S. Menon, B. Cao and T. Srikanthan, "A configurable dual-moduli multi-operand modulo adder," Proc. IEEE Int. Symp. Circuits and Systems, 2005, pp. 1630-1633.

[7] S. Menon and C. –H. Chang, "A reconfigurable multi-modulus modulo multiplier," Proc. IEEE Asia Pacific Conf. on Circuits and Systems, 2006, pp. 1168-1171.

[8] R. Muralidharan and C. –H. Chang, "Fixed and variable multi-modulus squarer architectures for triple moduli base of RNS," Proc. IEEE Int. Symp. Circuits and Systems, 2009, pp. 441-444.

[9] A. Strollo and D. Caro, "Booth folding encoding for high performance squarer circuits," IEEE Trans. on Circuits and Systems – II, vol. 50, no. 5, May 2003, pp. 250-254.

[10] C. Efstathiou, H. T. Vergos and D. Nikolos, "Modified Booth modulo 2^n-1 multipliers," IEEE Trans. on Computers, vol. 53, no. 3, Mar. 2004, pp. 370-374.

[11] Y. Ma, "A simplified architecture for modulo (2^n+1) multiplication," IEEE Trans. on Computers, vol. 47, no. 3, Mar. 1998, pp. 333-337.

[12] H. T. Vergos, D. Bakalis and C. Efstathiou, "Fast modulo 2^n+1 multi-operand adders and residue generators," Integration, the VLSI Journal, vol. 43, no. 1, Jan. 2010, pp. 42-48.

TABLE II. CMOS VLSI EXPERIMENTAL RESULTS

	Squarer [8]		Proposed 3-Moduli Squarer				Proposed 4-Moduli Squarer			
n	***Area (um^2)***	***Delay (ns)***	***Area (um^2)***	***Delay (ns)***	***Area reduction***	***Delay reduction***	***Area (um^2)***	***Delay (ns)***	***Area reduction***	***Delay reduction***
8	6530	0.88	5601	0.90	14.2%	-1.6%	6914	0.93	-5.9%	-5.4%
12	14224	1.08	11870	1.01	16.5%	6.8%	13367	1.05	6.0%	2.3%
16	24471	1.19	19229	1.07	21.4%	9.6%	21386	1.16	12.6%	1.9%
20	38787	1.27	28210	1.18	27.3%	6.7%	30959	1.25	20.2%	1.2%
32	95346	1.42	65754	1.34	31.0%	5.4%	70475	1.35	26.1%	4.9%