FPGA Based Design of Elliptic Curve Cryptography Coprocessor

WANG You-Bo
School of Communication,
Shandong Normal University,
Jinan, China, 250014
wybxy7076@163.com

DONG Xiang-Jun
School of Information
Science and Technology,
Shandong Institute of Light
Industry, Jinan, China, 250353
dxj@sdili.edu.cn

TIAN Zhi-Guang
College of Mechanical and
Electrical Engineering,
Nanjing University of Aeronautics
and Astronautics, China,
tiany9812@163.com

Abstract

Elliptic curve cryptography plays a crucial role in networking and communication security. FPGA based architecture of elliptic curve cryptography coprocessor is proposed in this paper. The coprocessor is consisted of elements operation over binary finite fields, point adding and doubling on elliptic curve and scalar multiplication, and these modules are described in Verilog HDL. In this coprocessor a new type of FPGA-based modular multiplier architecture is proposed for trade-off multiplication between bit serial and bit parallel. Experiment results show that coprocessor designed in this paper can achieve high performance. With the coprocessor embedded in PCI adaptor is realized for encryption and decryption.

1. Introduction

Public key cryptography has gained much attention from both companies and end user for information security. Public key with the enormous growth of the computer and communication security became the type of cryptography that controls electronic mail, e-commerce and Internet [5]. Elliptic curve cryptography (ECC) delivers the highest strength per bit of any known public-key system because of the hard problem upon which it is based. ECC has received increased commercial acceptance as evidenced by its inclusion in standard by accredited standard organizations such as ANSI, IEEE, ISO, NIST [3]. Each application has different demands on the utilized cryptosystem. Since there has been more than 20 years theoretical research on ECC, our study in this paper is mostly focused on coprocessor implementation based on FPGA device.

The remainder of this paper is organized as follow. Section 2 introduces mathematical background and computational hierarchy of the ECC. In section 3 elements operations over finite field are described. Section 4 provides point arithmetic over elliptic curve group in detail. Section 5 proposes scalar multiplication scheme and architecture of coprocessor. Section 6 provides hardware experiment result and conclusions.

2. Hierarchy of ECC Arithmetic

Arithmetic of ECC is primarily based on several elements operations over finite field, which have unique advantage in terms of hardware implementation. Elements operations over finite fields, such as elements modular addition, subtraction, modular squaring and modular multiplication, are quite necessary. However, the computing performance is greatly affected by the element representation basis, two of which, polynomial basis and normal basis, are widely used in binary finite field [7].

The mathematic foundation of ECC is based on Weierstrass equation. Nowadays the research work on the equation is focused on affine and projective coordinates form. The equation in affine coordinates is

$$y^2+xy = x^3+ax^2+b \quad (b\neq 0) \tag{1}$$

Several projective coordinate systems for elliptic curve equation have been proposed in order to avoid the time-consuming inversion operation [2]. Projective coordinate system proposed by Lopez and Dahab is suitable for hardware implementation, and it is called L-D projective coordinates in this paper. In the L-D projective coordinates, *point* $(X{:}Y{:}Z)(Z\neq 0)$ is corresponding to *point* $(X/Z, Y/Z^2)$ in the affine coordinates, and the elliptic curve equation is simplified as below.

$$Y^2+XYZ = X^3Z+aX^2Z^2+bZ^4 \tag{2}$$

Research work shows that point arithmetic in L-D projective coordinates is the most efficient one. All of the following algorithms in this paper about the point operation are in L-D projective coordinates.

This work was partially supported by Scientific Research Development Project of Shandong Provincial Education Department, China (J06N06).

Third International Conference on Natural Computation (ICNC 2007)

IEEE COMPUTER SOCIETY

Scalar multiplication, based on the point arithmetic on elliptic curve, is another crucial arithmetic. The elliptic curve scalar multiplication is computed by repeated point addition such as

$$\underbrace{P+P+\ldots+P}_{k}=k\cdot P=Q \tag{3}$$

Where k is a positive integer and P and Q are points on elliptic curve. Scalar multiplication is the foundation of ECC, and it directly contributes to discrete logarithm problem and several cryptographic schemes.

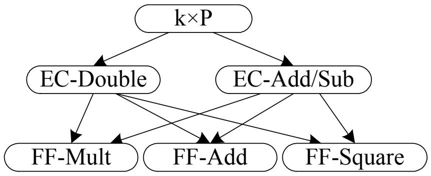

Figure 1. Computing hierarchy of ECC arithmetic

The hierarchy of arithmetic for elliptic curve scalar multiplication is depicted in Figure 1, where *FF-Mult* represents finite field multiplication, *FF-Add* represents finite field adding, *FF-Square* represents finite field squaring, *EC-Add* represents addition on elliptic curve, and so on [4].

3. Arithmetic over Binary Finite Field

The most important arithmetic over binary finite field is modular multiplication; hence it is necessary to design an efficient multiplier, which depends on basis choice for given finite field. Considering resource consumption, both shifted canonical basis and type II optimal normal basis are utilized in our design to achieve the efficient multiplier.

A straightforward choice for a basis is the ordered set $\{1,\beta,\beta^2,\cdots,\beta^{m-1}\}$ where $\beta\in GF(2^m)$, which is called the canonical basis[6].

If the set of elements $M=\{\beta,\beta^2,\beta^4,\ldots,\beta^{2^{m-1}}\}$ forms a basis for some $\beta\in GF(2^m)$, then the basis M is called normal basis and the element β is called normal element. Furthermore, a type II optimal normal basis for the field $GF(2^m)$ is constructed using the normal element $\beta=\xi+\xi^{-1}$, where ξ is a primitive $(2m+1)$th root of unity in binary finite field, i.e., $\xi^{2m+1}=1$ and $\xi^i\neq 1$ for any $1\leq i<2m+1$. Consequently, type II optimal normal basis can also be expressed as

$$M=\{\xi+\xi^{-1},\xi^2+\xi^{-2},\xi^{2^2}+\xi^{-2^2},\cdots,\xi^{2^{(m-1)}}+\xi^{-2^{(m-1)}}\} \tag{4}$$

In fact, there is another basis $N=\{\beta, \beta^2,\cdots, \beta^{m-1}, \beta^m\}$, where $\beta_i=\xi^i+\xi^{-i}$, so the basis N can be expressed as

$$N=\{\xi+\xi^{-1},\xi^2+\xi^{-2},\xi^3+\xi^{-3},\cdots,\xi^m+\xi^{-m}\} \tag{5}$$

There exists a relationship between the representation basis M and N. Basis N (also called shifted canonical basis) can be obtained by a simple permutation of the basis elements in M. As a result of the following presented facts, the basis element of the form $\xi^{2^i}+\xi^{-2^i}$ $(i\in[1,m])$ can be written uniquely as $\xi^j+\xi^{-j}$ $(j\in[1,m])$.

The representation of field element A with type II optimal normal basis is given as

$$A=\sum_{i=0}^{m-1}a_i^{'}\beta^{2^i} \tag{6}$$

The bases M and N are equivalent. The representation of field element A with canonical basis is given as

$$A=\sum_{i=1}^{m}a_i\beta_i \tag{7}$$

The permutation between the coefficients $a_j = a'_i$ can be expressed as

$$j=\begin{cases}k & k\in[1,\ m]\\(2m+1)-k & k\in[m+1,\ 2m]\end{cases} \tag{8}$$

Where $k=2^{i-1}$ (mod $2m+1$) for $i = 1, 2, \cdots,m$. For example, the relationship between the two basis is shown below over finite field $GF(2^5)$.

$\beta=\ (\xi+\xi^{-1})\ \ =(\xi+\xi^{-1})\ =\beta_1$
$\beta^2=(\xi^2+\xi^{-2})=(\xi^2+\xi^{-2})=\beta_2$
$\beta^4=(\xi^4+\xi^{-4})=(\xi^4+\xi^{-4})=\beta_4$
$\beta^8=(\xi^8+\xi^{-8})\ =(\xi^{-3}+\xi^3)=\beta_3$
$\beta^{16}=(\xi^{16}+\xi^{-16})=(\xi^5+\xi^{-5})=\beta_5$

Where $\{a, a^2, a^4, a^8, a^{16}\}$ denotes type II optimal normal basis and $\{a_1, a_2, a_3, a_4, a_5\}$ denotes canonical basis.

Given the elements A and B over finite field represented with shifted canonical basis, i.e. $A=\sum_{i=1}^{m}a_i\beta_i$, $B=\sum_{i=1}^{m}b_i\beta_i$ and the multiplying product C is represented with

$$C=\sum_{i=1}^{m}a_i\beta_i\times\sum_{i=1}^{m}b_i\beta_i=\sum_{i=1}^{m}c_i\beta_i \tag{9}$$

As an example, for elements A and B over $GF(2^5)$, the product of $C = A\times B$ is provided as below.

$c_1=a_1(b_2+0)+a_2(b_1+b_3)+a_3(b_2+b_4)+a_4(b_3+b_5)+a_5(b_4+b_5)$
$c_2=a_1(b_3+b_1)+a_2(b_4+0)+a_3(b_1+b_5)+a_4(b_2+b_5)+a_5(b_4+b_3)$
$c_3=a_1(b_4+b_2)+a_2(b_5+b_1)+a_3(b_5+0)+a_4(b_1+b_4)+a_5(b_2+b_3)$
$c_4=a_1(b_5+b_3)+a_2(b_5+b_2)+a_3(b_1+b_4)+a_4(b_3+0)+a_5(b_1+b_2)$
$c_5=a_1(b_4+b_5)+a_2(b_3+b_4)+a_3(b_2+b_3)+a_4(b_1+b_2)+a_5(b_1+0)$

Sunur and Koc have proposed an efficient full parallel multiplier with the above formula [6], but the efficient full serial modular multiplier and trade-off multiplication design have not been proposed yet. We design an efficient full serial modular multiplier in FPGA in this paper, and a trade-off multiplier design scheme is proposed for the first time. The multiplier uses both shifted canonical basis and type II optimal normal basis. The architecture of the trade-off version multiplier is depicted in Figure 2, in which symbol ⊗ denotes logic *AND* operation and ⊕ denotes logic *XOR* operation, and the bit a_0 and b_0 in the figure are zero. The multiplier consumes only half of the total clock cycles compared to full serial multiplier, that is, $\lceil m/2\rceil$ clock cycles.

Third International Conference on Natural Computation (ICNC 2007)
0-7695-2875-9/07 $25.00 © 2007 IEEE

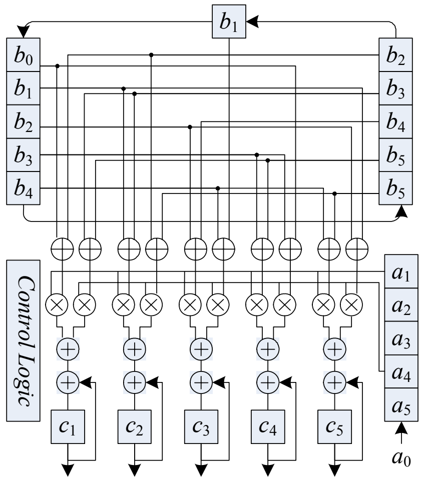

Figure 2. Trade-off design of modular multiplier

Addition of two elements over finite field is a different operation. Fortunately, the adding operation in binary finite field can be finished easily with exclusive XOR logic circuit. Element squaring is also a necessary arithmetic in the field arithmetic, to implement squaring arithmetic in FPGA just use register rotating.

4. Point Operations on Elliptic Curve

Both point addition and subtraction operations are foundation of scalar multiplication. Point addition operation includes two different points adding and two same point doubling. In this section, we will discuss the implementation of point operations inside FPGA in detail based coprocessor in L-D projective coordinates.

Since the modular inversion operation over field is a time consuming operation, points on elliptic curve should be represented in projective coordinates to avoid it. In the L-D projective coordinates, elliptic curve equation is the same as formula (2). The points addition formula that do not involve modular inversion operation can be derived by converting the point to affine projective as $x=X/Z$, $y=Y/Z^2$ at first, then adding the affine points with the formula, and finally clearing the denominators. With the method mentioned above, the distinct points adding and the same point doubling formula can be derived.

As for two distinct points adding, suppose the first point $P(X_1,Y_1,Z_1)$ and the second point $Q(X_2,Y_2,Z_2)$ are on elliptic curve, the adding result of the two different point is $R(X_3,Y_3,Z_3)$ as following.

$$(X_1,Y_1,Z_1)+(X_2,Y_2,Z_2)=(X_3,Y_3,Z_3) \qquad (10)$$

The two distinct points adding formula is shown as in detail [1], and it requires 13 field multiplications.

1. $A_0 = Y_2 \cdot Z_1^2$
2. $A_1 = Y_1 \cdot Z_2^2$
3. $B_0 = X_2 \cdot Z_1$
4. $B_1 = X_1 \cdot Z_2$
5. $C = A_0 + A_1$
6. $D = B_0 + B_1$
7. $E = Z_1 \cdot Z_2$
8. $F = D \cdot E$
9. $Z_3 = F^2$
10. $G = D^2 \cdot (F + aE^2)$
11. $H = C \cdot F$
12. $X_3 = C^2 + H + G$
13. $I = D^2 \cdot B_0 \cdot E + X_3$
14. $J = D^2 \cdot A_0 + X_3$
15. $Y_3 = H \cdot I + Z_3 \cdot J$

It is necessary to note that when using the double and add method for scalar multiplication $k \cdot P$ the point P is never modified in all distinct point adding. In mixed coordinates (one is projective point and the other is affine point) the point P maintaining in affine coordinates will simplify the computing. The point adding in mixed coordinates

$$(X_0, Y_0, Z_0) + (X_1, Y_1, 1) = (X_2, Y_2, Z_2)$$

is computed by

1. $A = Y_1 \cdot Z_0^2 + Y_0$
2. $B = X_1 \cdot Z_0 + X_0$
3. $C = Z_0 \cdot B$
4. $D = B^2 \cdot (C + aZ_0)$
5. $Z_2 = C^2$
6. $E = A \cdot C$
7. $X_2 = A^2 + D + E$
8. $F = X_2 + X_1 \cdot Z_2$
9. $G = X_2 + Y_1 \cdot Z_2^2$
10. $Y_2 = E \cdot F + Z_2 \cdot G$

The point doubling operation is to add a point on the elliptic curve with itself. Suppose point $P(X_1,Y_1,Z_1)$ and the projective coordinate form of doubling formula is

$$2P(X_1,Y_1,Z_1) = Q(X_2,Y_2,Z_2) \qquad (11)$$

Implementation of the operation requires 4 field multiplications. The point doubling is computed by

$$\begin{aligned} Z_2 &= Z_1^2 \cdot X_1^2 \\ X_2 &= X_1^4 + b \cdot Z_1^4 \\ Y_2 &= bZ_1^4 + X_2 \cdot (aZ_2 + Y_1^2 + bZ_1^4) \end{aligned} \qquad (12)$$

The following algorithm implements a full point addition and subtraction in the projective coordinates.

```
Input: The field elements $a$ and $b$ defining a elliptic curve $E$ over $GF(2^m)$;
  projective coordinates $(X_1, Y_1, Z_1)$ and
  $(X_2, Y_2, Z_2)$ for points $P_1$ and $P_2$ on $E$.
Output: Projective coordinates $(X_2, Y_2, Z_2)$
  for the point $P_3 = P_1+P_2$

1.If $Z_1=0$, then output
  $(X_3, Y_3, Z_3)=(X_2, Y_2, Z_2)$ and stop.
2.If $Z_2=0$, then output
  $(X_3, Y_3, Z_3)=(X_1, Y_1, Z_1)$ and stop.
3.Set $(X_3, Y_3, Z_3)$=Add[$(X_1, Y_1, Z_1)$,
$(X_2, Y_2, Z_2)$].
4.If $(X_3, Y_3, Z_3)=(0,0,0)$, then
  set $(X_3, Y_3, Z_3)$=Double[$(X_1, Y_1, Z_1)$].
5. Output($X_3$, $Y_3$, $Z_3$).
```

The elliptic subtraction is implemented as follows.

```
Subtract[$(X_1, Y_1, Z_1)$, $(X_2, Y_2, Z_2)$]
    =FullAdd[$(X_1, Y_1, Z_1)$, $(X_2, U, Z_2)$]
```

Where $U = X_2Z_2 + Y_2$.

With modular addition, modular squaring and modular multiplication arithmetic implemented in FPGA device, the point addition module inside coprocessor can be designed. The addition of two different points operation requires nine 233-bit wide temporary registers. These registers can be implemented in Block RAM inside FPGA

Third International Conference on Natural Computation (ICNC 2007)
0-7695-2875-9/07 $25.00 © 2007 IEEE

IEEE COMPUTER SOCIETY

device. A very small amount of clock cycles is required to read and write the temporary data to and fro Block Ram. The architecture of the coprocessor is shown in Figure 3, in which *Mod Mult* is instantiated modular multiplication module in Verilog description, *X3*, *Z3* and *Qout* are 233-bit wide register, *Add Logic Control* and *Double Logic Control* are operation process finite state machine module.

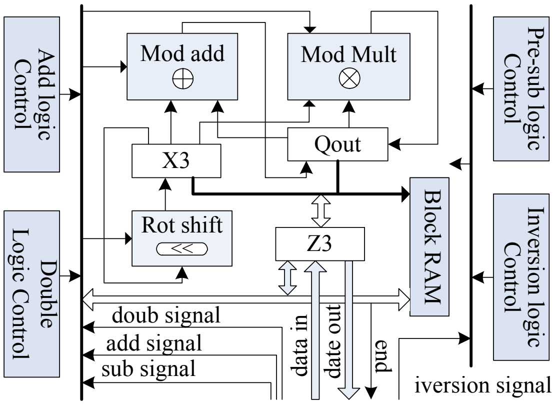

Figure 3. Architecture of point operation module

On the field of arithmetic operation over $GF(2^{233})$, the coprocessor can achieve two different point adding operation, point doubling operation and two point subtracting operation and so on. To improve computing performance the data bus inside coprocessor is 233-bit wide. All the computing operation is controlled by finite state machine.

5. Scalar multiplication

Many scalar multiplication algorithms for certain elliptic curve have been proposed, such as binary method, non-adjacent form (NAF) method, slide window NAF method, fixed comb method, Montgomery method, etc. Among these algorithms, the NAF method, without pre-computing, is more suitable for hardware implementation for its concision and efficiency. The following NAF algorithm is adopted in our design of coprocessor. The architecture of scalar multiplying module in FPGA device is depicted as Figure 4.

```
1.set $h_l h_{l-1}...h_1 h_0$ is binary representation of $3k$.
2.set $k_l k_{l-1}...k_1 k_0$ is binary representation of $k$.
3.set $S=Q$.
4.For $i$ from $l$-1 Downto 1 Do
    4.1 set $S = 2S$.
    4.2 if ($h_i$=1 && $k_i$=0)   $S=S+Q$;
    4.3 if ($h_i$=0 && $k_i$=1)   $S=S-Q$;
5.output $S$.
```

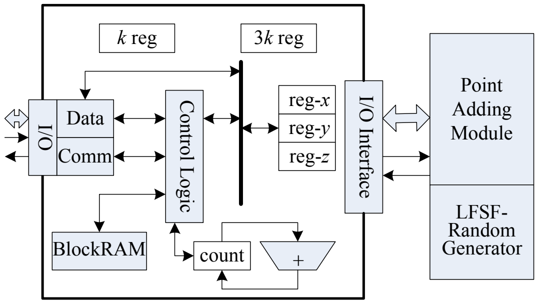

Figure 4. Architecture of scalar multiplying module

For example, two different elliptic curve points addition requires 9 field multiplications in mixed coordinate system, one is affine coordinate system and the other is L-D projective coordinate system. A point doubling require 4 field multiplications and each multiplication requires $\lceil 233/2 \rceil$ =117 clock cycles in $GF(2^{233})$. Suppose k is 233 bits and the hamming weight is approximately 70, so a general scalar multiplication requires about 183000 clock cycles. Since our ECC coprocessor can work at 80 MHz, that means 438 general scalar multiplications can be finished per second, and each scalar multiplication requires 2.28 ms.

Table 1. Comparison of published design

Design	Field	Speed (MHz)	$k{\cdot}P$ (ms)	Device
[8]	m=155	15	10.3	XC4020XL
[9]	m=155	36	6.8	320KGates
[10]	m=233	30	2.94	XC4085XL
[11]	m=160	21	3.8	XCV800-4
[12]	m=161	166	4.7	XC2V100
[13]	m=233	37	13.2	XCV2000E
our	m=233	80	2.28	XC3S1000

The performance comparison between different designing of processor is not always straightforward. There are so many aspects, such as field size, representing basis, point arithmetic, coordinate system selecting etc, influence the performances that it is hard to determine which one is more excellent. Yet we summarize several coprocessor published in recent years and list them in Table 1 with our own design in it. The data in the table mainly show time each scalar multiplication consuming.

6. Experiment and Conclusions

The FPGA based architecture of ECC arithmetic coprocessor introduced above is described with Verilog HDL. The Verilog description is correctly simulated in ModelSim and the synthesis of the description is finished in Xilinx's integrated software environment. The placing and routing process are finished in Xilinx's XC3S1000 device. The synthesis report shows that the coprocessor can work at over 100MHz. Experiment show that the chip works actually at 80-MHz successfully in the PCI card with the chip embedded in.

The FPGA based architecture of ECC arithmetic coprocessor is proposed in this paper according to the computational hierarchy of ECC. The architecture of our coprocessor is described with Verilog HDL, in which a new design of trade-off modular multiplier is proposed. The new design can achieve more convenience and flexibility into modular multiplier design over finite field. Moreover, the multiplier can achieve half or quarter of clock cycles consuming in the full bit-serial multiplier (*m* clock cycles in $GF(2^m)$). Our coprocessor can achieve a higher performance shown in above comparison. With the coprocessor embedded in, a PCI ECC adapter for data encryption and decryption is implemented.

References

[1] J.Lopez, R.Dahab. "Improved Algorithm for Elliptic Curve Arithmetic in GF(2^n) ". Selected Areas in Cryptography-SAC'98, LNCS 1556, 1999: pp. 201-212.

[2] IEEE Std P1363-2000. "IEEE Standard Specifications for Public-Key Cryptography". 2000.8.

[3] Michael Jung, "FPGA Based Implementation of an Elliptic Curve Coprocessor Utilizing Synthesizable VHDL code", Darmstadt University of Technology. Available at http://www.vlsi.informatik.tu-darmstadt.de/staff/mjung/publications/comprehensive.pdf

[4] Darrel Hankerson, Alfred J. Menezes, Scott Vanstone, "Guide to Elliptic Curve Cryptography". Springer-Verlag, New York. 2005.8.

[5] "ISO/IEC 15946-1 Information technology-Security Techniques–cryptographic techniques Based on Elliptic Curves", Committee Draft (CD), 1999.

[6] Sunar, B. Koc, C.K. "An Efficient Optimal Normal Basis Type II Multiplier". Computers, IEEE Transactions on, Jan 2001, pp.83-87.

[7] A. Reyhani-Masoleh and M. A. Hasan, "Efficient Digit-Serial Normal Basis Multipliers over GF(2^m)", in proceedings of IEEE International Symposium on Circuits and Systems (ISCAS 2002), May 2002, pp. 781-784.

[8] S. Sutikno, R. Effendi, A. Surya. "Design and Implementation of Arithmetic Processor GF(2^{155}) for Elliptic Curve Cryptosystems", in: Proceedings of the IEEE Asia-Pacific Conference on Circuits and Systems (APCCAS'98),1998, pp. 647~650.

[9] K.H.Leung, K.W.Ma, W.K. Wong, P.H.W.Leong, "FPGA implementation of a microcoded elliptic curve cryptographic processor", in: Proceedings of Field- Programmable Custom Computing Machines (FCCM'00), 2000, pp.68-76.

[10] Ernst, M, Henhapl, B.; Klupsch, S.; Huss, S. "FPGA based hardware acceleration for elliptic curve public key cryptosystems". Journal of Systems and Software Volume: 70, Issue: 3, March, 2004, pp. 299-313.

[11] Nele Mentens, Siddika Berna Ors, Bart Preneel. "An FPGA Implementation of an Elliptic Curve Processor over GF(2^m)". in Proceedings of the 2004 ACM Great lakes Symposium on VLSI, GLSVLSI 2004: VLSI in the Nanometer Era. pp. 454-457

[12] Morales-Sandoval Miguel, Feregrino-Uribe Claudia. "On the Hardware Design of an Elliptic Curve Cryptosystem". Proceedings of the Fifth Mexican International Conference in Computer Science, ENC 2004: pp.64~70.

[13] T. Kerins, E. M. Popovici and W. P. Marnane. "An FPGA Implementation of a Flexible, Secure Elliptic Curve Cryptography Processor", istinguished paper, International Workshop on Applied Reconfigurable Computing-ARC 2005, IADIS press, pp.22-30