International Journal of Computer Applications (0975 – 8887)
Volume 43– No.3, April 2012

Triple-Base Hybrid Joint Sparse Form and its Applications

Subhashis Maitra
Department of Electronics and Communication
Kalyani Government Engineering College
kalyani, Nadia, West Bengal, India

Amitabha Sinha
School of Information Technology
West Bengal University of Technology
Salt Lake, Kolkata, West Bengal, India

ABSTRACT

Multi-scalar multiplication and multi-exponentiation are the major problems in digital signal processing(DSP) and in public-key cryptography. In DSP, multiplication of the filter coefficients, which is used in different signal processing algorithms, is a time consuming operation. Also in elliptic curve cryptography(ECC), this multiplication process takes a lot of time. Though there are several algorithms to speed up this multiplication process, but they are not up to the satisfactory limit. The algorithms are based on the minimization of the multipliers or adders required, i.e. the minimization of the numbers of 'one' appear in the binary representation of the signal and the coefficients in DSP or of the number of general multiplications and points addition in ECC. Different existing algorithms in this context are Joint Sparse Form(JSF), w-NAF, Double Base Chain(DBC), Hybrid Binery-Ternary Joint Sparse Form(HBTJSF) etc. In this paper, a novel algorithm has been proposed which is the modified HBTJSF, known as Triple-Base Hybrid Joint Sparse Form(TBHJSF). The proposed method is based on the decomposition of an integer or fraction that mixes the power of the base 2,3 and 5. The experimental results show that it requires less numbers of multiplier and adder and hence show its novelty over other algorithms.

Keywords

DBC, DBNS, Digital Filter, DSP, ECC, HBTJSF, JSF, TBC, TBHJSF, TBNS, w-NAF.

1. INTRODUCTION

The complexities of multiplication and addition in the design of digital filter or multi-exponentiation operation and points addition in ECC are the major problem in current signal processing and different cryptographic algorithms. To speed-up these operation, Shamir's trick[1][2] can be used which eliminates the unnecessary separate computation of the two expressions. Shamir explained in[1], that two integers m and n can be expanded in the binary form at the same time and shown an extra savings of doublers and multipliers. He proposed that if 'a' represents the bit length of the largest exponent, this method requires 'a' squaring and 3a/4 multipliers on average. In ECC, where the operation [m]P + [n]Q is an important part to perform, the elements can be easily inverted using Shamir's algorithm. Here the scalars m and n can be represented as a 2×a matrix as,

$$\binom{k}{l} = \begin{pmatrix} k_{n-1} & k_{n-2} & \ldots\ldots\ldots\ldots\ldots & k_1 & k_0 \\ \mathrm{l}_{n-1} & \mathrm{l}_{n-2} & \ldots\ldots\ldots\ldots\ldots\ldots & \mathrm{l}_1 & \mathrm{l}_0 \end{pmatrix} \quad (1)$$

where k_i and $l_i \in \{ 1, 0, -1\}$ for all 'i', The numbers of adders required in Shamir's method is equal to the joint Hamming weight.

For example, two integers 145 and 207 can be represented in JNAF form using Shamir's method[1][2][3][4] as

$$\binom{425}{521} = \begin{pmatrix} 0\,1\,1 & 0\,1\,0\,1\,0 & 0\,1 \\ 1\,0\,0 & 0\,0\,0\,1\,0 & 0\,1 \end{pmatrix}_{\text{JNAF,}}$$

here the joint Hamming weight is 6. Hence only six adders are required to compute 425P + 521Q.

In case of digital signal processing (DSP), an FIR filter implementation of the linear convolution

$$Y(n) = \sum_{k=0}^{N-1} X(k).H(n-k) \quad (2)$$

Can be performed by multiplying $X(k)$ with $H(n-k)$.
To reduce the complexity of these multi-scalar multiplications, it is important to select filter coefficients with small number of nonzero binary digits.

Solinas, in[5][6], introduced Joint Sparse Form (JSF) where the average number of nonzero columns have further been reduced to a great extent. For example, the two given integers 425 and 521 in JSF form can be represented as

425 = (1 0 $\bar{1}$ 0 1 0 1 0 0 1) and
521 = (1 0 0 0 0 0 1 0 0 1)

Hence, here the computation of 425P + 521Q requires 9 doublers and 4 adders(in general, t-1 doublers and (t-1)/2 adders, where t denotes the bit-length).

In [7], Dimitrov and Cooklev introduced Hybrid Binery-Ternery Number System (HBTNS) to speed-up modular exponentiation and multi-scalar multiplication. Using this method, any integer 'n' can be represented in digits[] and base[] form as
Digits[] = [D_i, D_{i-1}, ------------- D_0] and
Base[] = [B_i, B_{i-1}, --------------- B_0] , where $D_i \in \{0, 1\}$ and $B_i \in \{2, 3\}$ (3)

The Algorithm proposed by J. Adikari et. all in [1] is given below.

Algorithm I: Hybrid binary-ternary joint sparse form [HBTJSF].

Input: Two positive integer k_1 and k_2
Output: Arrays hbt1[], hbt2[], base[]
1. i = 0
2. while $k_1 > 0$ or $k_2 > 0$ do
3. if $k_1 \equiv 0 \pmod 3$ and $k_2 \equiv 0 \pmod 3$ then
4. base[i] = 3;
5. hbt1[i] = hbt2[i] = 0;
6. $k_1 = k_1/3$; $k_2 = k_2/3$;
7. else if if $k_1 \equiv 0 \pmod 2$ and $k_2 \equiv 0 \pmod 2$ then
8. base[i] = 2;

9. hbt1[i] = hbt2[i] = 0;
10. $k_1 = k_1/2$; $k_2 = k_2/2$;
11. else
12. base[i] = 2;
13. hbt1[i] = k_1 mods 6; hbt2[i] = k_2 mods 6;
14. $k_1 = (k_1 –$ hbt1[i])/2; $k_2 = (k_2 –$ hbt2[i])/2;
15. end if
16. i = i + 1
17. end while
18. return hbt1[], hbt2[], base[]

With the help of this Algorithm I, 1134 in HBTNS form can be represented as[8][9]
Digits[] = [0, 0, 0, 0, 0, 1, 0, 0, 1] and
Base[] = [3, 3, 3, 3, 2, 2, 3, 2, 2],

whereas 1134 in binary form can be represented as 1134 = [1 0 0 0 1 1 0 1 1 1 0]. Hence it is clear that in HBTNS, there are only 9-digits among which 2 are non-zero whereas in binary form 1134 requires 11 bits among which 6 are non-zero. So HBTNS eliminates 4 non-zero digits. Elliptic curve scalar multiplication and coefficient multiplication in DSP based on mixing powers of the bases 2 and 3 require doublings and additions as well as triplings. In [1], Dimitrov et. al. proposed an efficient method of tripling for ordinary elliptic curves over large prime fields.

In [1], Adhikari, Dimitrov and Imbert proposed a new method, known as Hybrid Binary-Ternary Joint Sparse Form (HBTJSF), to further reduce the numbers of non-zero columns and hence to speed-up multiplication and exponentiation processes. In [1], Adhikari et. al. explained about how a pair of integers can be converted into HBTJSF. The conversion process starts by checking whether both the integers are divisible by 3. If they are divisible by 3, both digits are to be set to zero and the base is to be set to 3. If they are not divisible by 3, then it should be checked that whether they are divisible by 2. If they are divisible by 2, then digits are set to zero and the base is set to 2. Finally, if the pair is not divisible by 3 as well as by 2, the numbers should made divisible by 6 by subtracting k_i mods 6 ∈ { -2, -1, 0, 1, 2, 3} from k_i and then divide the results by 2. This concept will be clear from Example 1.

Example 1: Representation of 1556 and 1026 in HBTJSF.

Sol: 1556 and 1026 in HBTJSF can be represented as
$1556 = (\, 2\ 0\ \bar{1}\ 0\ 0\ \bar{2}\ 0\ \bar{2}\ 0\,)$
$1026 = (\, 1\ 0\ 1\ 0\ 0\ 1\ 0\ 3\ 0\,)$ and
Base[] = (2 3 2 2 3 2 3 2 2)

Since HBTJSF uses the digit set { -2, -1, 0, 1, 2, 3 } (mentioned in [1]), total 14 points are to be pre-computed to find kP + lQ as shown in Table 1[1].

Here, the points 2P, 2Q, 3P and 3Q need not to be computed since the pairs (2P, 2Q and 3P, 3Q) are divisible either by 2 or by 3. Again since P – 2Q is easily derived from 2Q – P, only one set of difference are to be computed and hence, among the 14 points, only 7 points are to be pre-computed.

2. PROPOSED ALGORITHM

Here a new algorithm known as Triple-Base Hybrid Joint Sparse Form [TBHJSF] will be discussed. This is basically a modification of HBTJSF. Here a new base(5) is to be added in Base[] array. The base 5 is chosen here in order to perform decimal shifting[since 5*2 =10 when multiplied with 1.7 gives 17 and hence if 17 can be represented in TBHJSF, 1.7 can easily be computed from the representation of 17].

In the proposed Algorithm, a pair of integers can be represented in TBHJSF by first checking whether the two integers are divisible by 5. If they are divisible by 5, the digits for both the integers will be set to 0, otherwise they should be check whether they are divisible by 3. If they are divisible by 3, the digits for both the integers will be set to 0, otherwise the integers should be check whether they are divisible by 2. If they are divisible by 2, the digits for both the integers will be set to 0, otherwise both the integers are made divisible by 30(2*3*5) by adding or subtracting x, where x ∈ { -14, -13, -----, 0, 1, 2, --------- 15} and then the sum are divided by 2. The quotients are then treated as the two integers and the previous steps are repeated.

Proposed Algorithm[Algorithm II]

Input: Two positive integers m_1 and m_2 ;
Output: Arrays Digit1[], Digit2[], Base[];

1. i = 0;
2. while $m_1 > 0$ or $m_2 > 0$, do
3. if $m_1 \equiv 0 (\text{mod } 5)$ and $m_2 \equiv 0 (\text{mod } 5)$ then
4. base[i] = 5;
5. Digit1[i] = Digit2[i] = 0;
6. $m_1 = m_1/5$, $m_2 = m_2/5$;
7. else if $m_1 \equiv 0 (\text{mod } 3)$ and $m_2 \equiv 0 (\text{mod } 3)$ then base[i] = 3;
8. Digit1[i] = Digit2[i] = 0;
9. $m_1 = m_1/3$, $m_2 = m_2/3$;
10. else if $m_1 \equiv 0 (\text{mod } 2)$ and $m_2 \equiv 0 (\text{mod } 2)$ then
11. base[i] = 2;
12. Digit1[i] = Digit2[i] = 0;
13. $m_1 = m_1/2$, $m_2 = m_2/2$;
14. else
15. base[i] = 2;
16. Digit1[i] = m_1 mods 30, Digit2[i] = m_2 mods 30 ;
17. $m_1 = (m_1$ - Digit1[i])/2, $m_2 = (m_2$ - Digit2[i])/2 ;
18. end if ;
19. i = i + 1;
20. end while;
21. return Digit1[],Digit2[], base[];

Example 2: Representation of 1556 and 1026 in TBHJSF.
Sol: 1556 and 1026 in TBHJSF can be represented as
$1556 = (\, 1\ 0\ 0\ \bar{4}\ 0\ 0\ \bar{2}\ 0\,)$
$1026 = (\, 1\ 0\ 0\ \overline{13}\ 0\ 0\ 3\ 0\,)$
Base[] = (2 3 5 2 3 5 2 2)
From the above example it is clear that the number of non-zero column has been reduced by one(though the number of reduction of non-zero column is more than one in general case) at the cost of the size of the pre-computation look-up-table as shown in Table A, Appendix-I. From the Table it is clear that among the 550 points, only 144 points are to be pre-computed. Whereas for DBNS, the numbers of pre-computation are 7 among 14 points. The pre-computation compression ratio for TBNS is (144/550) = 0.2618, whereas for DBNS, the same is (7/14) = 0.5 which is an advantage of TBNS.

3. THEORETICAL ANALYSIS

It is clear that any integer can be represented in the form 30i + 5j+ k with i∈{ 0,1,2,3,……}, j ∈ {0,1,2,3,4,5} and k ∈ { 0,1,2,3,4}. We have to find out how many non-zero column can be obtained using the proposed algorithm, i.e. how often the given integers are divisible by 30, 15, 10, 6, 5, 3 and 2. Let us consider two integers of the form (30i + 5j+ k) and (30i´ + 5j´+ k´). The total number of states available for different values of j and k are 900. We can denote these states as $S_{kj\ ,\ k'j'}$ and hence we have 750x750 transition matrix M, where M[30i + 5j+ k, 30i´ + 5j´+ k´] is equal to the probability $P(S2_{kj\ ,\ k'j'}| S1_{kj\ ,\ k'j'})$ to go from state $S1_{kj\ ,\ k'j'}$ to $S2_{kj\ ,\ k'j'}$. For example, $S_{00,00}$ represents the state when both the integers are divisible by 30, $S_{02,04}$ represents the states when both are divisible by 2 only, $S_{11,03}$ represents the state when both are divisible by 3 only and $S_{10,30}$ represents the state when both are divisible by 5 only. Now if two numbers are divisible by 30, then divisions by 5 performed in step 6 of the proposed algorithm lead to any of the state's $S_{00,00}$, $S_{00,11}$, $S_{00,22}$, $S_{00,33}$, $S_{00,44}$, $S_{11,00}$, $S_{11,11}$, $S_{11,22}$, $S_{11,33}$, $S_{11,44}$, $S_{22,00}$, $S_{22,11}$, $S_{22,22}$, $S_{22,33}$, $S_{22,44}$, $S_{33,00}$, $S_{33,11}$, $S_{33,22}$, $S_{33,33}$, $S_{33,44}$, $S_{44,00}$, $S_{44,11}$, $S_{44,22}$, $S_{44,33}$ and $S_{44,44}$ with probability 1/25. For example, two integers 150(j = k = 0) and 155(j = 1, k = 0) belonging to state $S_{00,10}$,when divided by 5 give 30(j = k = 0) and 31(j = 0, k = 1) which belong to state S_{0001}. Similarly other two integers of the same state($S_{00,10}$) are 150 and 185. These two integers when divided by 5 give 30(j = k = 0)and 37(j = 0, k = 2) which belong to state $S_{00,12}$. Thus it can be proved that two integers belonging to S_{0010} when divided by 5 give one of the 25 states like $S_{00,01}$, $S_{00,12}$, $S_{00,23}$, $S_{00,34}$, $S_{00,50}$, $S_{11,01}$, $S_{11,12}$, $S_{11,23}$, $S_{11,34}$, $S_{11,50}$, $S_{22,01}$, $S_{22,12}$, $S_{22,23}$, $S_{22,34}$, $S_{22,50}$, $S_{33,01}$, $S_{33,12}$, $S_{33,23}$, $S_{33,34}$, $S_{33,50}$, $S_{44,01}$, $S_{44,12}$, $S_{44,23}$, $S_{44,34}$ and $S_{44,50}$. That is the probability to go to state $S_{00,01}$ from $S_{00,10}$ when divided by 5, is 1/25. Similarly divisions by 3 in step 10 of Algorithm II lead to any of the state's $S_{10,00}$, $S_{10,20}$, $S_{10,40}$, $S_{30,00}$, $S_{30,20}$, $S_{30,40}$, $S_{50,00}$, $S_{50,20}$ and $S_{50,40}$ with probability 1/9 and divisions by 2 in step 14 lead to four different states with probability 1/4. For example, from $S_{00,20}$ we go to any of the four states $S_{00,10}$, $S_{00,40}$, $S_{30,10}$ and $S_{30,40}$ with probability 1/4. Now in the last case when the numbers are neither divisible by 5 or 3 nor by 2, the numbers are then made divisible by adding or subtracting (1, 2, 3, 4, ……15) from each of them and then performed divisions by 2 which lead to the states $S_{00,00}$, $S_{00,30}$, $S_{30,00}$ and $S_{30,30}$ with probability 1/4. The complete transition matrix is given in Table B, Appendix II.

Lemma 1: The probability to go from state S_{0020} to S_{0002} when divided by 5 is 1/25.

Proof: The pairs of integer belong to state S_{0020} are (150, 160), (150, 190), (150, 220), (150, 250), (150, 280), (180, 160), (180, 190), (180, 220), (180, 250), (180, 280), (210, 160), (210, 190), (210, 220), (210, 250), (210, 280), (240, 160), (240, 190), (240, 220), (240, 250), (240, 280), (270, 160), (270, 190), (270, 220), (270, 250), (270, 280) and others.

Now

(150, 160) dividing by 5 give 30(j = k = 0) and 32(j = 0, k = 2) which belong to state S_{0002}.
(150, 190) dividing by 5 give 30(j = k = 0) and 38(j = 1, k = 3) which belong to state S_{0013}.
(150, 220) dividing by 5 give 30(j = k = 0) and 44(j = 2, k = 4) which belong to state S_{0024}.
(150, 250) dividing by 5 give 30(j = k = 0) and 50(j = 4, k = 0) which belong to state S_{0040}.
(150, 280) dividing by 5 give 30(j = k = 0) and 56(j = 5, k = 1) which belong to state S_{0051}.

(180, 160) dividing by 5 give 36(j = k = 1) and 32(j = 0, k = 2) which belong to state S_{1102}.
(180, 190) dividing by 5 give 36(j = k = 1) and 38(j = 1, k = 3) which belong to state S_{1113}.
(180, 220) dividing by 5 give 36(j = k = 1) and 44(j = 2, k = 4) which belong to state S_{1124}.
(180, 250) dividing by 5 give 36(j = k = 1) and 50(j = 4, k = 0) which belong to state S_{1140}.
(180, 280) dividing by 5 give 36(j = k = 1) and 56(j = 5, k = 1) which belong to state S_{1151}.

(210, 160) dividing by 5 give 42(j = k = 2) and 32(j = 0,k = 2) which belong to state S_{2202}.
(210, 190) dividing by 5 give 42(j = k = 2) and 38(j = 1, k = 3) which belong to state S_{2213}.
(210, 220) dividing by 5 give 42(j = k = 2) and 44(j = 2, k = 4) which belong to state S_{2224}.
(210, 250) dividing by 5 give 42(j = k = 2) and 50(j = 4, k = 0) which belong to state S_{2240}.
(210, 280) dividing by 5 give 42(j = k = 2) and 56(j = 5, k = 1) which belong to state S_{2251}.

(240, 160) dividing by 5 give 48(j = k = 3) and 32(j = 0, k = 2) which belong to state S_{3302}.
(240, 190) dividing by 5 give 48(j = k = 3) and 38(j = 1, k = 3) which belong to state S_{3313}.
(240, 220) dividing by 5 give 48(j = k = 3) and 44(j = 2, k = 4) which belong to state S_{3324}.
(240, 250) dividing by 5 give 48(j = k = 3) and 50(j = 4, k = 0) which belong to state S_{3340}.
(240, 280) dividing by 5 give 48(j = k = 3) and 56(j = 5, k = 1) which belong to state S_{3351}.

(270, 160) dividing by 5 give 54(j = k = 4) and 32(j = 0, k = 2) which belong to state S_{4402}.
(270, 190) dividing by 5 give 54(j = k = 4) and 38(j = 1, k = 3) which belong to state S_{4413}.
(270, 220) dividing by 5 give 54(j = k = 4) and 44(j = 2, k = 4) which belong to state S_{4424}.
(270, 250) dividing by 5 give 54(j = k = 4) and 50(j = 4, k = 0) which belong to state S_{4440}.
(270, 280) dividing by 5 give 54(j = k = 4) and 56(j = 5, k = 1) which belong to state S_{4451}.

Hence the possible outcomes when pair of integers belonging to state S_{0020} divided by 5 are pairs of integers belonging to states S_{0002}, S_{0013}, S_{0024}, S_{0040}, S_{0051}, S_{1102}, S_{1113}, S_{1124}, S_{1140}, S_{1151}, S_{2202}, S_{2213}, S_{2224}, S_{2240}, S_{2251}, S_{3302}, S_{3313}, S_{3324}, S_{3340}, S_{3351}, S_{4402}, S_{4413}, S_{4424}, S_{4440} and S_{4451}. The other pairs of integers belonging to S_{0020} when divided by 5 go to any one of the above mentioned states. Hence the probability to go from state S_{0020} to S_{0002} when divided by 5 is 1/25.

Lemma 2: The probability to go from state S_{0002} to S_{0001} when divided by 2 is 1/4.

Proof: The pairs of integer belong to state S_{0002} are (60, 62), (60, 92), (90, 62), (90, 92) and others.

Now (60, 62) dividing by 2 give 30(j = k = 0) and 31(j = 0, k = 1) which belong to state S_{0001}.
(60, 92) dividing by 2 give 30(j = k = 0) and 46(j = 3, k = 1) which belong to state S_{0031}.
(90, 62) dividing by 2 give 45(j = 3, k = 0) and 31(j = 0, k = 1) which belong to state S_{3001}.
(90, 92) dividing by 2 give 45(j = 3, k = 0) and 46(j = 3, k = 1) which belong to state S_{3031}.

Hence the possible outcomes when pair of integers belonging to state S_{0002} divided by 2 are pairs of integers belonging to states S_{0001}, S_{0031}, S_{3001} and S_{3031}. The other pairs of integers belonging to S_{0002} when divided by 2 go to any one of the above mentioned states. Hence the probability to go from state S_{0002} to S_{0001} when divided by 2 is 1/4.

The stationary distribution π_∞ is obtained by taking the limiting value of $\pi_0 M^n$[M =Transition probability Matrix], where π_0 = initial probability = (1/900, 1/900, 1/900, ….1/900), i.e.

$$\pi_\infty = \lim_{n \to \infty} \pi_0 M^n \quad (4)$$

Hence we have, $\pi_\infty[i]$= the stationary distribution of the ith element in the row vector π after n steps is 7.07×10^{-2}, for i $\in\{0,15,450,465\}$ and otherwise $\pi_\infty[i] = 2.87 \times 10^{-3}$. With the help of these two values we can compute the following average probabilities.

$p_5 = \sum_{i=0}^{29}\sum_{j=0}^{5}\sum_{k=0}^{4} \pi_\infty\ (30i+5j+k)$, [if i,j,k ≡ 0 (mod 5)]

$=2.\ \pi_\infty(\mathrm{i})[\text{for i} = 0 \text{ and } 450]+10.\pi_\infty(\mathrm{i})$ [for other values of i]

$= 2 \times 7.07 \times 10^{-2} + 10 \times 2.87 \times 10^{-3} = 1701 \times 10^{-4} \approx 43/250$

$p_3 = \sum_{i=0}^{29}\sum_{j=0}^{5}\sum_{k=0}^{4} \pi_\infty\ (30i+5j+k)$, [if i,j,k ≡ 0 (mod 3)]

$= 4.\ \pi_\infty(\mathrm{i})[\text{for i} = 0, 15, 450 \text{ and } 465] + 20.\ \pi_\infty(\mathrm{i})$ [for other values of i]

$= 4 \times 7.07 \times 10^{-2} + 20 \times 2.87 \times 10^{-3} = 3402 \times 10^{-4} \approx 73/250$

$p_z = \sum_{i=0}^{29}\sum_{j=0}^{5}\sum_{k=0}^{4} \pi_\infty\ (30i+5j+k)$, [if i,j,k ≡ 0(mod 5) or i,j,k ≡ 0(mod 3)or i,j,k ≡ 0(mod 2)]

$= 4.\ \pi_\infty(\mathrm{i})\ [\text{for i} = 0, 15, 450 \text{ and } 465] + 159.\ \pi_\infty(\mathrm{i})$ [for other values of i]

$= 4 \times 7.07 \times 10^{-2} + 159 \times 2.87 \times 10^{-3} = 7391.3 \times 10^{-4} \approx 185/250$

Here p_5 and p_3 denote the probabilities to perform a division by 5 and 3 respectively and p_z represents the probability to perform a division so that a zero column will be generated. Hence the probability to perform a division by 2 is $p_2 = 1 - (p_5 + p_3) \approx 134/250$ and the probability to generate a nonzero column is $p_{nz} = 1 - p_z \approx 65/250$.
Using these probabilities we can compute the average base as

$$ß = \sqrt[4000]{2^{134}3^{73}5^{43}} \approx 2.6357 \quad (5)$$

Hence for a pair of n-bit integers, the number of columns using the proposed Algorithm can therefore be calculated as $(\log_ß 2)$xn ≈0.7152n. Hence using TBHJSF, the number of addition, required in the calculation to find out the response of a digital filter or to find out the point addition in ECC, is Number of addition = (probability of non-zero columns)x 0.5273n = (65/250)x0.7152n ≈ 0.186n.

Table 2 gives a comparative study of the proposed Algorithm with HBTJSF, JSF and interleaving w-NAF with respect to average base, average number of columns, average number of base-2 columns, base-3 columns, base-5 columns, non-zero columns and number of pre-computations required. Table 3 gives the theoretical results of TBHJSF in comparison with HBTJSF, JSF and w-NAF. This results is shown here in rounded to nearest hundredth form.

4. COMPARISON

The results obtained from the theoretical analysis using TBHJSF when compared with that obtained using HBTJS, JSF and interleaving w-NAF methods show the novelty of the proposed algorithm. Here also two types of curves like ordinary elliptic curve over large prime fields with Jacobian coordinates(with a = -3) and tripling oriented Doche-Icart-Kohel curves[8], have been considered like[9][10]. Table 3 shows the costs of the operations on the two types of curves and Table 4 shows the summarized results of operations for 256-bit pairs of integers using the data of Table II with respect to TBHJSF, HBTJSF, JSF and interleaving w-NAF. Here the costs of pre-computations is not included. If the results are compared then it can be inferred that TBHJSF is advantageous than other methods. Similar types of values can be obtained for other size also. Figure.1. shows graphically the requirements of adders, doublers, triplers and pentuplers and Figure.2. shows the requirements of multipliers used for ECC over prime fields in Weierstrass Jacobian Coordinates (a = -3)[11][12] with respect to the proposed method, HBTJSF, JSF, 4-NAF methods. The curves are drawn for up to 2 KB pairs of integers. From these figure, the advantages of proposed algorithm can be clearly understood. The differences will be significant for the pairs of integers of large size.

5. CONCLUSIONS

The proposed algorithm can also be used efficiently in DSP to design FIR filter. In DSP, the computational complexities for arithmetic operations like addition, multiplication etc. are the major problems that can be reduced to great extent using the proposed method. From the representation of any integer in Triple Base Number System(TBNS)[13] obtained using the proposed methods, representation in Single Digit TBNS can be derived at the cost of a Look-up-table that represents the prime numbers in SDTBNS[14] form. Hence the proposed algorithm can further reduce the number of hardware and at the same time enhance the speed.

6. REFERENCES

[1] J. Adikari, V. Dimitrov and L. Imbert, "Hybrid Binary-Ternary Joint Sparse Form and its Application in Elliptic Curve Cryptography", Draft, July 2, 2008, supported by the Natural Science and Engineering Research Council of Canada.

[2] D. Hankerson, A. Menezes and S. Vanstone, Guide to Elliptic Curve Cryptography, Springer, 2004.

[3] A. D. Booth, "A signed binary multiplication technique", Quarterly Journal of Applied Mechanics and Applied Mathematics, vol. 4, no.2, pp. 236-240,1951, reprinted in E.E. Swartzlander, Computer Arithmetic, vol. 1, IEEE Computer Society Press Tutorial, Los Alamitos, CA, 1990.

[4] T. ElGamal, "A public key cryptosystem and a signature scheme based on discrete logarithms", IEEE Transactions on Information Theory, vol. 31, no. 4, pp. 469-472, July 1985.

[5] J. A. Solinas, Low-weight binary representation for pairs of integers", Center for Applied Cryptographic Research, University of Waterloo, Waterloo, ON, Canada, Research Report CORR 2001- 41, 2001.

[6] Avanzi, R. M., Dimitrov, V. S., Doche, C., Sica, F.: Extending Scalar Multiplication using Double Bases, in: Lai, X., Chen, K.(Eds.), ASIACRYPT 2006, LNCS, vol. 4284, pp. 130 –144, Springer,Heidelberg(2006).

[7] V. Dimitrov and T. V. Cooklev, "Two algorithm for modular exponentiation based on nonstandard arithmetic", IEICE Transactions on Fundamentals of Electronics, Communications and Computer Science, vol. E78-A, no. 1, pp. 82 -87, Jan. 1995, special issue on cryptography and information security.

[8] V. Dimitrov, L. Imbert and P. K Mishra, "Efficient and secure elliptic curve point multiplications using double-base chains", in advances in Cryptography, ASIACRYPT'05, ser. Lecture Notes in Computer Science, vol. 3788, Springer, 2005, pp. 59-78.

[9] C. Doche and L. Imbert, "Extended double-base number system with applications to elliptic curve cryptography", in Progress in Cryptography, INDOCRYPT'06,ser. Lecture Notes in Computer Science, vol.4329, , Springer, 2006, pp. 335 – 348.

[10] V. S. Miller, "Uses of elliptic curves in cryptography", in Advances in Cryptology, CRYPTO'85, ser. Lecture Notes in Computer Science, vol.218., Springer,1986, pp. 417-428.

[11] D.J. Bernstein, P.Birkner, T.Lange and C.Peters, "Optimizing double-base elliptic-curve single-scalar multiplications", in Progress in Cryptography – INDOCRYPT2007, ser. Lecture Notes in Computer Science, vol.4859, Springer, 2007, pp. 167 - 182.

[12] C. Doche, T. Icart and D. R. Kohel, "Efficient scalar multiplication by isogenies decomposition", in Public Key Cryptography, PKC'06, ser. Lecture Notes in Computer Science, vol.3958,, Springer, 2006, pp. 191 - 206.

[13] Pavel Sinha, Amitabha Sinha, Krishanu Mukherjee and Kenneth Alan Newton, "Triple Base Number Digital and Numerical Processing System", Patent filed under E.S.P.Microdesign Inc., Pennsylvania, U.S.A., U.S.Pat. App. No. 11/488, 138.

[14] S. Maitra, A. Sinha, "A Single Digit Tripple Base Number System – A New Concept for Implementing High Performance Multiplier Unit for DSP Applications", Proceedings of the sixth International Conference on Information, Communication and Signal Processing (ICICS2007), December, 10-13,2007.

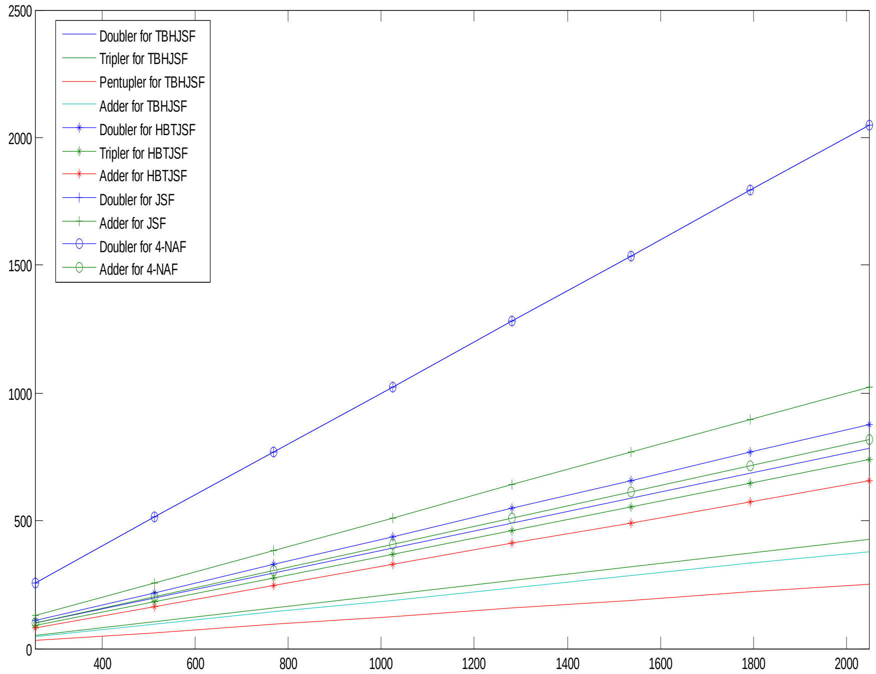

Figure 1. Requirements of hardware for different size of integers w.r.t. TBHJSF, HBTJSF, JSF and 4-NAF.

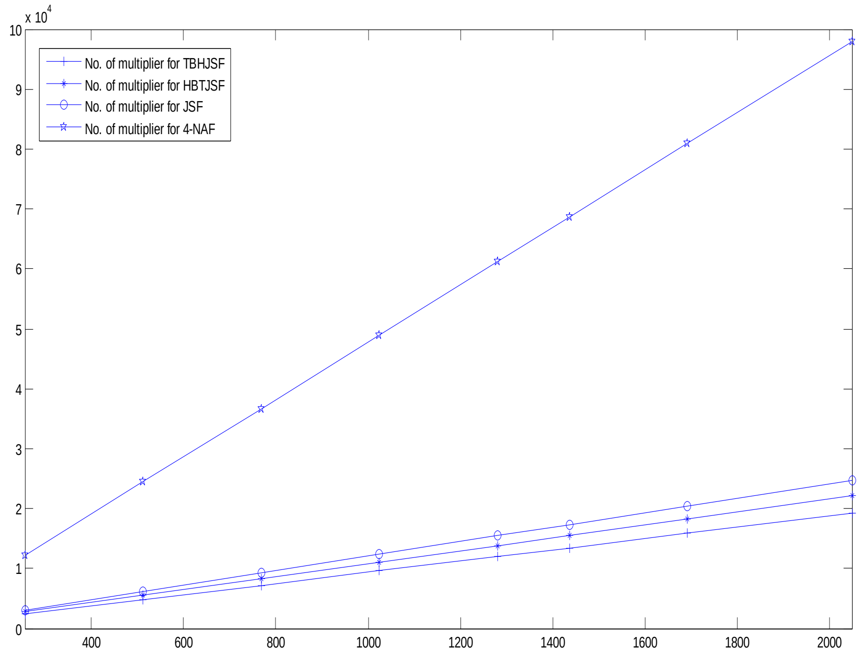

Figure.2. Number of multipliers required for ECC over prime fields in Weierstrass Jacobian coordinates using the proposed method and HBTJSF, JSF and 4-NAF methods.

Table 1

14 points pre-computation for HBTJSF Scalar multiplication as mentioned in [1]

	P	----	----
Q	P ± Q	2P ± Q	3P ± Q
---	P ± 2Q	----	3P ± Q
---	P ± 3Q	2P ± 3Q	----

Table 2

Comparative study of the proposed Algorithm with HBTJSF, JSF and interleaving w-NAF

Parameters	TBHJSF	HBTJSF	JSF	Interleaving w-NAF
Average base	2.6357	2.4077	2	2
Average no. of columns	0.7152n	0.7888n	n + 1	n + 1
Average no. of base-2 columns	0.3833n	0.4278n	n + 1	n + 1
Average no. of base-3 columns	0.2088n	0.3608n	0	0
Average no. of base-5 columns	0.123n	0	0	0
Average no. of non-zero columns	0.1859n	0.3208n	0.5n	2n/(w+1)
Pre-computation	144	14	2	$2^{w-1} - 2$

Table 3

Costs of some curve operations for ordinarily elliptic curves over prime fields in Jacobian Coordinates(a = - 3) and Tripling-oriented DIK curves

Weierstrass/Jacobian(a = -3)

	Cost	S = 0.8M
Doubling	3M+5S	7M
Tripling	7M+7S	12.6M
Pentupling (1Pentupling = 2.5Doubling)	7.5M+12.5S	17.5M
Addition(mixed)	7M+4S	10.2M

Tripling-oriented DIK

	Cost	S = 0.8M
Doubling	2M+7S	7.6M
Tripling	6M+6S	10.8M
Pentupling (1Pentupling = 2.5Doubling)	5M+17.5S	19M
Addition(mixed)	7M+4S	10.2M

Table 4

Comparisons between HBTPJSF, HBTJSF, JSF and Interleaving w-NAF for 256-bit integers

Weierstrass/Jacobian(a = -3)

	TBHJSF	HBTJSF	JSF	Inter. 4-NAF
Multiplier Counts for doubling	686	770	1792	1792
Multiplier Counts for tripling	673	1164	0	0
Multiplier Counts for pentupling	551	0	0	0
Multiplier Counts for add.	485	838	1306	1044
Total Multiplier Counts	2442	2772	3098	2836
Pre-computation	144	14	2	6

[Though total multiplication counts for Interleaving 4-NAF is 2836, the actual counts is somewhat more than this, because here, the multiplication counts for pre-computation have not considered.]

Tripling-oriented DIK

	TBHJSF	HBTJSF	JSF	Inter. 4-NAF
Multiplier Counts for doubling	746	833	1946	1946
Multiplier Counts for tripling	796	998	0	0
Multiplier Counts for pentupling	598	0	0	0
Multiplier Counts for addition	485	838	1306	1044
Total Multiplier Counts	2625	2669	3252	2990
Pre-computation	144	14	2	6

Appendix – I

TABLE A

PRECOMPUTATION LOOK-UP-TABLE FOR TBHJSF

	P	---	---	-----	-----	---	7P	-----	----	----	11 P	---	13 P	14 P	---
Q	P ± Q	2P ± Q	3P ± Q	4P ± Q	5P ± Q	6P ± Q	7P ± Q	8P ± Q	9P ± Q	10 P ± Q	11 P ± Q	12 P ± Q	13 P ± Q	14 P ± Q	15 P ± Q
--	P ± 2Q	----	3P ± 2Q	4P ± 2Q	5P ± 2Q	6P ± 2Q	7P ± 2Q	8P ± 2Q	9P ± 2Q	10 P ± 2Q	11 P ± 2Q	12 P ± 2Q	13 P ± 2Q	14 P ± 2Q	15 P ± 2Q
--	P ± 3Q	2P ± 3Q	----	4P ± 3Q	5P ± 3Q	6P ± 3Q	7P ± 3Q	8P ± 3Q	9P ± 3Q	10 P ± 3Q	11 P ± 3Q	12 P ± 3Q	13 P ± 3Q	14 P ± 3Q	15 P ± 3Q
--	P ± 4Q	2P ± 4Q	3P ± 4Q	----	5P ± 4Q	6P ± 4Q	7P ± 4Q	8P ± 4Q	9P ± 4Q	10 P ± 4Q	11 P ± 4Q	12 P ± 4Q	13 P ± 4Q	14 P ± 4Q	15 P ± 4Q
--	P ± 5Q	2P ± 5Q	3P ± 5Q	4P ± 5Q	----	6P ± 5Q	7P ± 5Q	8P ± 5Q	9P ± 5Q	10 P ± 5Q	11 P ± 5Q	12 P ± 5Q	13 P ± 5Q	14 P ± 5Q	15 P ± 5Q
--	P ± 6Q	2P ± 6Q	3P ± 6Q	4P ± 6Q	5P ± 6Q	----	7P ± 6Q	8P ± 6Q	9P ± 6Q	10 P ± 6Q	11 P ± 6Q	12 P ± 6Q	13 P ± 6Q	14 P ± 6Q	15 P ± 6Q
7Q	P ± 7Q	2P ± 7Q	3P ± 7Q	4P ± 7Q	5P ± 7Q	6P ± 7Q	----	8P ± 7Q	9P ± 7Q	10 P ± 7Q	11 P ± 7Q	12 P ± 7Q	13 P ± 7Q	14 P ± 7Q	15 P ± 7Q
---	P ± 8Q	2P ± 8Q	3P ± 8Q	4P ± 8Q	5P ± 8Q	6P ± 8Q	7P ± 8Q	----	9P ± 8Q	10 P ± 8Q	11 P ± 8Q	12 P ± 8Q	13 P ± 8Q	14 P ± 8Q	15 P ± 8Q
---	P ± 9Q	2P ± 9Q	3P ± 9Q	4P ± 9Q	5P ± 9Q	6P ± 9Q	7P ± 9Q	8P ± 9Q	----	10 P ± 9Q	11 P ± 9Q	12 P ± 9Q	13 P ± 9Q	14 P ± 9Q	15 P ± 9Q
10Q	P ± 10Q	2P ± 10 Q	3P ± 10 Q	4P ± 10 Q	5P ± 10 Q	6P ± 10 Q	7P ± 10 Q	8P ± 10 Q	9P ± 10 Q	----	11 P ± 10 Q	12 P ± 10 Q	13 P ± 10 Q	14 P ± 10 Q	15 P ± 10 Q
---	P ± 11Q	2P ± 11 Q	3P ± 11 Q	4P ± 11 Q	5P ± 11 Q	6P ± 11 Q	7P ± 11 Q	8P ± 11 Q	9P ± 11 Q	10 P ± 11 Q	---	12 P ± 11 Q	13 P ± 11 Q	14 P ± 11 Q	15 P ± 11 Q
----	P ± 12Q	2P ± 12	3P ± 12	4P ± 12	5P ± 12	6P ± 12	7P ± 12	8P ± 12	9P ± 12	10 P ± 12	11 P ± 12	----	13 P ± 12	14 P ± 12	15 P ± 12

		Q	Q	Q	Q	Q	Q	Q	Q	Q	Q		Q	Q	Q
13Q	P ± 13Q	2P ± 13 Q	3P ± 13 Q	4P ± 13 Q	5P ± 13 Q	6P ± 13 Q	7P ± 13 Q	8P ± 13 Q	9P ± 13 Q	10 P ± 13 Q	11 P ± 13 Q	12 P ± 13 Q	---	14 P ± 13 Q	15 P ± 13 Q
14Q	P ± 14Q	2P ± 14 Q	3P ± 14 Q	4P ± 14 Q	5P ± 14 Q	6P ± 14 Q	7P ± 14 Q	8P ± 14 Q	9P ± 14 Q	10 P ± 14 Q	11 P ± 14 Q	12 P ± 14 Q	13 P ± 14 Q	----	15 P ± 14 Q
----	P ± 15Q	2P ± 15 Q	3P ± 15 Q	4P ± 15 Q	5P ± 15 Q	6P ± 15 Q	7P ± 15 Q	8P ± 15 Q	9P ± 15 Q	10 P ± 15 Q	11 P ± 15 Q	12 P ± 15 Q	13 P ± 15 Q	14 P ± 15 Q	----

Appendix – II
Table B

To show a 900x900 Transition matrix, 300 pages are required. Hence instead of transition matrix, here we will show the probabilities of transition from one state to another for the major states in tabular form.

Probabilities	Transition from states	Transition to states
1/25	S_{0000} ,S_{0010}, S_{0020}, S_{0030}, S_{0040}, S_{0050}	S_{0000} - S_{0010}, S_{0011}- S_{0021}, S_{0022} - S_{0032}, S_{0033} - S_{0043}, S_{0044} - S_{0054}, S_{1100} - S_{1110}, S_{1111} - S_{1121}, S_{1122} - S_{1132}, S_{1133} - S_{1143}, S_{1144} - S_{1154}, S_{2200} - S_{2210}, S_{2211} - S_{2221}, S_{2222} - S_{2232}, S_{2233} - S_{2243}, S_{2244} - S_{2254}, S_{3300} - S_{3310}, S_{3311} - S_{3321}, S_{3322} - S_{3332}, S_{3333} - S_{3343}, S_{3344} - S_{3354}, S_{4400} - S_{4410}, S_{4411} - S_{4421}, S_{4422} - S_{4432}, S_{4433} - S_{4443}, S_{4444} - S_{4454},
	S_{1000}, S_{1010}, S_{1020}, S_{1030}, S_{1040}, S_{1050},	S_{0100} – S_{0110}, S_{0112} – S_{0121}, S_{0123} – S_{0132}, S_{0134} – S_{0143}, S_{0150} – S_{0154}, S_{1200} – S_{1210}, S_{1211}- S_{1221}, S_{1222} - S_{1232}, S_{1233} – S_{1243}, S_{1244} – S_{1254}, S_{2300} – S_{2310}, S_{2311} – S_{2321}, S_{2322} – S_{2332}, S_{2333} – S_{2343}, S_{2344} – S_{2354}, S_{3400} – S_{3410}, S_{3411} – S_{3421}, S_{3422} – S_{3432}, S_{3433} – S_{3443}, S_{3444} - S_{3454}, S_{5000} – S_{5010}, S_{5011} – S_{5021}, S_{5022} – S_{5032}, S_{5033} – S_{5043}, S_{5044} – S_{5054}
	S_{2000}, S_{2010}, S_{2020}, S_{2030}, S_{2040}, S_{2050},	S_{0200} – S_{0210}, S_{0211} – S_{0221}, S_{0222} – S_{0232}, S_{0233} – S_{0243}, S_{0244} – S_{0254}, S_{1300} – S_{1310}, S_{1311}- S_{1321}, S_{1322} – S_{1332}, S_{1333} – S_{1343}, S_{1344} – S_{1354}, S_{2400} - S_{2410}, S_{2411} – S_{2421}, S_{2422} – S_{2432}, S_{2433} – S_{2443}, S_{2444} – S_{2454}, S_{4000} - S_{4010}, S_{4011} –S_{4021}, S_{4022} – S_{4032},S_{4033} – S_{4043}, S_{4044} – S_{4054}, S_{5100} – S_{5110}, S_{5111} – S_{5121}, S_{5122} – S_{5132}, S_{5133} – S_{5143}, S_{5144} – S_{5154}
	S_{3000}, S_{3010}, S_{3020}, S_{3030}, S_{3040}, S_{3050},	S_{0300} – S_{0310}, S_{0311} – S_{0321}, S_{0322} – S_{0332}, S_{0333} – S_{0343}, S_{0344} – S_{0354}, S_{1400} – S_{1410}, S_{1411} – S_{1421}, S_{1422} – S_{1432}, S_{1433} – S_{1443}, S_{1444} – S_{1454}, S_{3000} – S_{3010}, S_{3011} – S_{3021}, S_{3022} – S_{3032}, S_{3033} – S_{3043}, S_{3044} – S_{3054}, S_{4100} – S_{4110}, S_{4111} – S_{4121}, S_{4122} – S_{4132}, S_{4133} – S_{4143}, S_{4144} – S_{4154}, S_{5200} – S_{5210}, S_{5211} – S_{5221}, S_{5222} – S_{5232}, S_{5233} – S_{5243}, S_{5244} – S_{5254}
	S_{4000}, S_{4010}, S_{4020}, S_{4030}, S_{4040}, S_{4050},	S_{0400} – S_{0410}, S_{0411} – S_{0421}, S_{0422} – S_{0432}, S_{0433} – S_{0443}, S_{0444} – S_{0454}, S_{2000} – S_{2010}, S_{2011} – S_{2021}, S_{2022} – S_{2032}, S_{2033} – S_{2043}, S_{2044} – S_{2054}, S_{3100} – S_{3110}, S_{3111} – S_{3121},S_{3122} – S_{3132}, S_{3133} – S_{3143}, S_{3144} – S_{3154}, S_{4200} – S_{4210}, S_{4211} – S_{4221}, S_{4222} – S_{4232}, S_{4233} – S_{4243}, S_{4244} – S_{4254}, S_{5300} – S_{5310}, S_{5311} – S_{5321}, S_{5322} – S_{5332}, S_{5333} – S_{5343}, S_{5344} – S_{5354}
	S_{5000}, S_{5010}, S_{5020}, S_{5030}, S_{5040}, S_{5050},	S_{1000} – S_{1010}, S_{1011} – S_{1021}, S_{1022} – S_{1032}, S_{1033} – S_{1043}, S_{1044} – S_{1054}, S_{2100} – S_{2110}, S_{2111} – S_{2121}, S_{2122} – S_{2132}, S_{2133} – S_{2143}, S_{2144} – S_{2154}, S_{3200} – S_{3210}, S_{3211} – S_{3221}, S_{3222} – S_{3232}, S_{3233} – S_{3243}, S_{3244} – S_{3254}, S_{4300} – S_{4310}, S_{4311} – S_{4321}, S_{4322} – S_{4332}, S_{4333} – S_{4343}, S_{4344} – S_{4354}, S_{5400} – S_{5410}, S_{5411} – S_{5421}, S_{5422} – S_{5432}, S_{5433} – S_{5443}, S_{5444} – S_{5454}
1/9	S_{0003}, S_{0011}, S_{0014}, S_{0022}, S_{0033}, S_{0041}, S_{0044}, S_{0052},	S_{0001}, S_{0021}, S_{0041}, S_{2001}, S_{2021}, S_{2041}, S_{4001}, S_{4021}, S_{4041}, S_{0002}, S_{0022}, S_{0042}, S_{2002}, S_{2022}, S_{2042}, S_{4002}, S_{4022}, S_{4042}, S_{0003}, S_{0023}, S_{0043}, S_{2003}, S_{2023}, S_{2043}, S_{4003}, S_{4023}, S_{4043}, S_{0004}, S_{0024}, S_{0044}, S_{2004}, S_{2024}, S_{2044}, S_{4004}, S_{4024}, S_{4044}, S_{0011}, S_{0031}, S_{0051}, S_{2011}, S_{2031}, S_{2051}, S_{4011}, S_{4031}, S_{4051}, S_{0012}, S_{0032}, S_{0052}, S_{2012}, S_{2032}, S_{2052}, S_{4012}, S_{4032}, S_{4052}, S_{0013}, S_{0033}, S_{0053}, S_{2013}, S_{2033}, S_{2053}, S_{4013}, S_{4033}, S_{4053}, S_{0014}, S_{0034}, S_{0054}, S_{2014}, S_{2034}, S_{2054}, S_{4014}, S_{4034}, S_{4054},
	S_{0303}, S_{0311}, S_{0314}, S_{0322}, S_{0330},S_{0333}, S_{0341}, S_{0344}, S_{0352},	S_{0101}, S_{0121}, S_{0141}, S_{2101}, S_{2121}, S_{2141}, S_{4101}, S_{4121}, S_{4141}, S_{0102}, S_{0122}, S_{0142}, S_{2102}, S_{2122}, S_{2142}, S_{4102}, S_{4122}, S_{4142}, S_{0103}, S_{0123}, S_{0143}, S_{2103}, S_{2123}, S_{2143}, S_{4103}, S_{4123}, S_{4143}, S_{0104}, S_{0124}, S_{0144}, S_{2104}, S_{2124}, S_{2144}, S_{4104}, S_{4124}, S_{4144}, S_{0110}, S_{0130}, S_{0150}, S_{2110}, S_{2130}, S_{2150}, S_{4110}, S_{4130}, S_{4150}, S_{0111}, S_{0131}, S_{0151}, S_{2111}, S_{2131}, S_{2151}, S_{4111}, S_{4131}, S_{4151}, S_{0112}, S_{0132}, S_{0152}, S_{2112}, S_{2132}, S_{2152}, S_{4112}, S_{4132}, S_{4152}, S_{0113}, S_{0133}, S_{0153}, S_{2113}, S_{2133}, S_{2153}, S_{4113}, S_{4133}, S_{4153}, S_{0114}, S_{0134}, S_{0154}, S_{2114}, S_{2134}, S_{2154}, S_{4114}, S_{4134}, S_{4154},
	S_{1103}, S_{1111}, S_{1114}, S_{1122}, S_{1130}, S_{1133}, S_{1141}, S_{1144}, S_{1152},	S_{0201}, S_{0221}, S_{0241}, S_{2201}, S_{2221}, S_{2241}, S_{4201}, S_{4221}, S_{4241}, S_{0202}, S_{0222}, S_{0242}, S_{2202}, S_{2222}, S_{2242}, S_{4202}, S_{4222}, S_{4242}, S_{0203}, S_{0223}, S_{0243}, S_{2203}, S_{2223}, S_{2243}, S_{4203}, S_{4223}, S_{4243}, S_{0204}, S_{0224}, S_{0244}, S_{2204}, S_{2224}, S_{2244}, S_{4204}, S_{4224}, S_{4244}, S_{0210}, S_{0230}, S_{0250}, S_{2210}, S_{2230}, S_{2250}, S_{4210}, S_{4230}, S_{4250}, S_{0211}, S_{0231}, S_{0251}, S_{2211}, S_{2231}, S_{2251}, S_{4211}, S_{4231}, S_{4251}, S_{0212}, S_{0232}, S_{0252}, S_{2212}, S_{2232}, S_{2252}, S_{4212}, S_{4232}, S_{4252}, S_{0213}, S_{0233}, S_{0253}, S_{2213}, S_{2233}, S_{2253}, S_{4213}, S_{4233}, S_{4253}, S_{0214}, S_{0234}, S_{0254}, S_{2214}, S_{2234}, S_{2254}, S_{4214}, S_{4234}, S_{4254},
	S_{1403}, S_{1411}, S_{1414}, S_{1422}, S_{1430}, S_{1433}, S_{1441}, S_{1444}, S_{1452},	S_{0301}, S_{0321}, S_{0341}, S_{2301}, S_{2321}, S_{2341}, S_{4301}, S_{4321}, S_{4341}, S_{0302}, S_{0322}, S_{0342}, S_{2302}, S_{2322}, S_{2342}, S_{4302}, S_{4322}, S_{4342}, S_{0303}, S_{0323}, S_{0343}, S_{2303}, S_{2323}, S_{2343}, S_{4303}, S_{4323}, S_{4343}, S_{0304}, S_{0324}, S_{0344}, S_{2304}, S_{2324}, S_{2344}, S_{4304}, S_{4324}, S_{4344}, S_{0310}, S_{0330}, S_{0350}, S_{2310}, S_{2330}, S_{2350}, S_{4310}, S_{4330}, S_{4350}, S_{0311}, S_{0331}, S_{0351}, S_{2311}, S_{2331}, S_{2351}, S_{4311}, S_{4331}, S_{4351}, S_{0312}, S_{0332}, S_{0352}, S_{2312}, S_{2332}, S_{2352}, S_{4312}, S_{4332}, S_{4352}, S_{0313}, S_{0333}, S_{0353}, S_{2313}, S_{2333}, S_{2353}, S_{4313}, S_{4333}, S_{4353}, S_{0314}, S_{0334}, S_{0354}, S_{2314}, S_{2334}, S_{2354}, S_{4314}, S_{4334}, S_{4354},
	S_{2203}, S_{2211}, S_{2214}, S_{2222}, S_{2230}, S_{2233}, S_{2241}, S_{2244}, S_{2252},	S_{0401}, S_{0421}, S_{0441}, S_{2401}, S_{2421}, S_{2441}, S_{4401}, S_{4421}, S_{4441}, S_{0402}, S_{0422}, S_{0442}, S_{2402}, S_{2422}, S_{2442}, S_{4402}, S_{4422}, S_{4442}, S_{0403}, S_{0423}, S_{0443}, S_{2403}, S_{2423}, S_{2443}, S_{4403}, S_{4423}, S_{4443}, S_{0404}, S_{0424}, S_{0444}, S_{2404}, S_{2424}, S_{2444}, S_{4404}, S_{4424}, S_{4444}, S_{0410}, S_{0430}, S_{0450}, S_{2410}, S_{2430}, S_{2450}, S_{4410}, S_{4430}, S_{4450}, S_{0411}, S_{0431}, S_{0451}, S_{2411}, S_{2431}, S_{2451}, S_{4411}, S_{4431}, S_{4451}, S_{0412}, S_{0432}, S_{0452}, S_{2412}, S_{2432}, S_{2452},

		S_{4412}, S_{4432}, S_{4452}, S_{0413}, S_{0433}, S_{0453}, S_{2413}, S_{2433}, S_{2453}, S_{4413}, S_{4433}, S_{4453}, S_{0414}, S_{0434}, S_{0454}, S_{2414}, S_{2434}, S_{2454}, S_{4414}, S_{4434}, S_{4454},
	S_{3003}, S_{3011}, S_{3014}, S_{3022}, S_{3033}, S_{3041}, S_{3044}, S_{3052},	S_{1001}, S_{1021}, S_{1041}, S_{3001}, S_{3021}, S_{3041}, S_{5001}, S_{5021}, S_{5041}, S_{1002}, S_{1022}, S_{1042}, S_{3002}, S_{3022}, S_{3042}, S_{5002}, S_{5022}, S_{5042}, S_{1003}, S_{1023}, S_{1043}, S_{3003}, S_{3023}, S_{3043}, S_{5003}, S_{5023}, S_{5043}, S_{1004}, S_{1024}, S_{1044}, S_{3004}, S_{3024}, S_{3044}, S_{5004}, S_{5024}, S_{5044}, S_{1011}, S_{1031}, S_{1051}, S_{3011}, S_{3031}, S_{3051}, S_{5011}, S_{5031}, S_{5051}, S_{1012}, S_{1032}, S_{1052}, S_{3012}, S_{3032}, S_{3052}, S_{5012}, S_{5032}, S_{5052}, S_{1013}, S_{1033}, S_{1053}, S_{3013}, S_{3033}, S_{3053}, S_{5013}, S_{5033}, S_{5053}, S_{1014}, S_{1034}, S_{1054}, S_{3014}, S_{3034}, S_{3054}, S_{5014}, S_{5034}, S_{5054},
	S_{3303}, S_{3311}, S_{3314}, S_{3322}, S_{3330}, S_{3333}, S_{3341}, S_{3344}, S_{3352},	S_{1101}, S_{1121}, S_{1141}, S_{3101}, S_{3121}, S_{3141}, S_{5101}, S_{5121}, S_{5141}, S_{1102}, S_{1122}, S_{1142}, S_{3102}, S_{3122}, S_{3142}, S_{5102}, S_{5122}, S_{5142}, S_{1103}, S_{1123}, S_{1143}, S_{3103}, S_{3123}, S_{3143}, S_{5103}, S_{5123}, S_{5143}, S_{1104}, S_{1124}, S_{1144}, S_{3104}, S_{3124}, S_{3144}, S_{5104}, S_{5124}, S_{5144}, S_{1110}, S_{1130}, S_{1150}, S_{3110}, S_{3130}, S_{3150}, S_{5110}, S_{5130}, S_{5150}, S_{1111}, S_{1131}, S_{1151}, S_{3111}, S_{3131}, S_{3151}, S_{5111}, S_{5131}, S_{5151}, S_{1112}, S_{1132}, S_{1152}, S_{3112}, S_{3132}, S_{3152}, S_{5112}, S_{5132}, S_{5152}, S_{1113}, S_{1133}, S_{1153}, S_{3113}, S_{3133}, S_{3153}, S_{5113}, S_{5133}, S_{5153}, S_{1114}, S_{1134}, S_{1154}, S_{3114}, S_{3134}, S_{3154}, S_{5114}, S_{5134}, S_{5154},
	S_{4103}, S_{4111}, S_{4114}, S_{4122}, S_{4130}, S_{4133}, S_{4141}, S_{4144}, S_{4152},	S_{1201}, S_{1221}, S_{1241}, S_{3201}, S_{3221}, S_{3241}, S_{5201}, S_{5221}, S_{5241}, S_{1202}, S_{1222}, S_{1242}, S_{3202}, S_{3222}, S_{3242}, S_{5202}, S_{5222}, S_{5242}, S_{1203}, S_{1223}, S_{1243}, S_{3203}, S_{3223}, S_{3243}, S_{5203}, S_{5223}, S_{5243}, S_{1204}, S_{1224}, S_{1244}, S_{3204}, S_{3224}, S_{3244}, S_{5204}, S_{5224}, S_{5244}, S_{1210}, S_{1230}, S_{1250}, S_{3210}, S_{3230}, S_{3250}, S_{5210}, S_{5230}, S_{5250}, S_{1211}, S_{1231}, S_{1251}, S_{3211}, S_{3231}, S_{3251}, S_{5211}, S_{5231}, S_{5251}, S_{1212}, S_{1232}, S_{1252}, S_{3212}, S_{3232}, S_{3252}, S_{5212}, S_{5232}, S_{5252}, S_{1213}, S_{1233}, S_{1253}, S_{3213}, S_{3233}, S_{3253}, S_{5213}, S_{5233}, S_{5253}, S_{1214}, S_{1234}, S_{1254}, S_{3214}, S_{3234}, S_{3254}, S_{5214}, S_{5234}, S_{5254},
	S_{4403}, S_{4411}, S_{4414}, S_{4422}, S_{4430}, S_{4433}, S_{4441}, S_{4444}, S_{4452},	S_{1301}, S_{1321}, S_{1341}, S_{3301}, S_{3321}, S_{3341}, S_{5301}, S_{5321}, S_{5341}, S_{1302}, S_{1322}, S_{1342}, S_{3302}, S_{3322}, S_{3342}, S_{5302}, S_{5322}, S_{5342}, S_{1303}, S_{1323}, S_{1343}, S_{3303}, S_{3323}, S_{3343}, S_{5303}, S_{5323}, S_{5343}, S_{1304}, S_{1324}, S_{1344}, S_{3304}, S_{3324}, S_{3344}, S_{5304}, S_{5324}, S_{5344}, S_{1310}, S_{1330}, S_{1350}, S_{3310}, S_{3330}, S_{3350}, S_{5310}, S_{5330}, S_{5350}, S_{1311}, S_{1331}, S_{1351}, S_{3311}, S_{3331}, S_{3351}, S_{5311}, S_{5331}, S_{5351}, S_{1312}, S_{1332}, S_{1352}, S_{3312}, S_{3332}, S_{3352}, S_{5312}, S_{5332}, S_{5352}, S_{1313}, S_{1333}, S_{1353}, S_{3313}, S_{3333}, S_{3353}, S_{5313}, S_{5333}, S_{5353}, S_{1314}, S_{1334}, S_{1354}, S_{3314}, S_{3334}, S_{3354}, S_{5314}, S_{5334}, S_{5354},
	S_{5203}, S_{5211}, S_{5214}, S_{5222}, S_{5230}, S_{5233}, S_{5241}, S_{5244}, S_{5252},	S_{1401}, S_{1421}, S_{1441}, S_{3401}, S_{3421}, S_{3441}, S_{5401}, S_{5421}, S_{5441}, S_{1402}, S_{1422}, S_{1442}, S_{3402}, S_{3422}, S_{3442}, S_{5402}, S_{5422}, S_{5442}, S_{1403}, S_{1423}, S_{1443}, S_{3403}, S_{3423}, S_{3443}, S_{5403}, S_{5423}, S_{5443}, S_{1404}, S_{1424}, S_{1444}, S_{3404}, S_{3424}, S_{3444}, S_{5404}, S_{5424}, S_{5444}, S_{1410}, S_{1430}, S_{1450}, S_{3410}, S_{3430}, S_{3450}, S_{5410}, S_{5430}, S_{5450}, S_{1411}, S_{1431}, S_{1451}, S_{3411}, S_{3431}, S_{3451}, S_{5411}, S_{5431}, S_{5451}, S_{1412}, S_{1432}, S_{1452}, S_{3412}, S_{3432}, S_{3452}, S_{5412}, S_{5432}, S_{5452}, S_{1413}, S_{1433}, S_{1453}, S_{3413}, S_{3433}, S_{3453}, S_{5413}, S_{5433}, S_{5453}, S_{1414}, S_{1434}, S_{1454}, S_{3414}, S_{3434}, S_{3454}, S_{5414}, S_{5434}, S_{5454},
1/4	S_{0002}, S_{0004}, S_{0013}, S_{0024}, S_{0031}, S_{0042}, S_{0051}, S_{0053}, S_{0202}, S_{0204}, S_{0213}, S_{0224}, S_{0231}, S_{0242}, S_{0251}, S_{0253}, S_{0402}, S_{0404}, S_{0413}, S_{0424}, S_{0431},S_{0442}, S_{0451}, S_{0453}, S_{1302}, S_{1304}, S_{1313}, S_{1324}, S_{1331}, S_{1342}, S_{1351}, S_{1353}, S_{2402}, S_{2404}, S_{2413}, S_{2424}, S_{2431}, S_{2442}, S_{2451}, S_{2453}, S_{3102}, S_{3104}, S_{3113}, S_{3124}, S_{3131}, S_{3142}, S_{3151}, S_{3153}, S_{4202}, S_{4204}, S_{4213}, S_{4224}, S_{4231}, S_{4242}, S_{4251}, S_{4253}, S_{5102}, S_{5104}, S_{5113}, S_{5124}, S_{5131}, S_{5142}, S_{5151}, S_{5153}, S_{5302}, S_{5304}, S_{5313}, S_{5324}, S_{5331}, S_{5342}, S_{5351}, S_{5353}, (these all even states)	S_{0001}, S_{0031}, S_{3001}, S_{3031}, S_{0002}, S_{0032}, S_{3002}, S_{3032}, S_{0004}, S_{0034}, S_{3004}, S_{3034}, S_{0012}, S_{0042}, S_{3012}, S_{3042}, S_{0013}, S_{0043}, S_{3013}, S_{3043}, S_{0021}, S_{0051}, S_{3021}, S_{3051}, S_{0023}, S_{0053}, S_{3023}, S_{3053}, S_{0024}, S_{0054}, S_{3024}, S_{3054}, S_{0101}, S_{0131}, S_{3101}, S_{3131}, S_{0102}, S_{0132}, S_{3102}, S_{3132}, S_{0104}, S_{0134}, S_{3104}, S_{3134}, S_{0112}, S_{0142}, S_{3112}, S_{3142}, S_{0113}, S_{0143}, S_{3113}, S_{3143}, S_{0121}, S_{0151}, S_{3121}, S_{3151}, S_{0123}, S_{0153}, S_{3123}, S_{3153}, S_{0124}, S_{0154}, S_{3124}, S_{3154}, S_{0201}, S_{0231}, S_{3201}, S_{3231}, S_{0202}, S_{0232}, S_{3202}, S_{3232}, S_{0204}, S_{0234}, S_{3204}, S_{3234}, S_{0212}, S_{0242}, S_{3212}, S_{3242}, S_{0213}, S_{0243}, S_{3213}, S_{3243}, S_{0221}, S_{0251}, S_{3221}, S_{3251}, S_{0223}, S_{0253}, S_{3223}, S_{3253}, S_{0224}, S_{0254}, S_{3224}, S_{3254}, S_{0214}, S_{0401}, S_{0431}, S_{3401}, S_{3431}, S_{0402}, S_{0432}, S_{3402}, S_{3432}, S_{0404}, S_{0434}, S_{3404}, S_{3434}, S_{0412}, S_{0442}, S_{3412}, S_{3442}, S_{0413}, S_{0443}, S_{3413}, S_{3443}, S_{0421}, S_{0451}, S_{3421}, S_{3451}, S_{0423}, S_{0453}, S_{3423}, S_{3453}, S_{0424}, S_{0454},S_{3424}, S_{3454}, S_{1201}, S_{1231}, S_{4201},S_{4231}, S_{1202}, S_{1232}, S_{4202}, S_{4232}, S_{1204}, S_{1234}, S_{4204}, S_{4234}, S_{1212}, S_{1242}, S_{4212}, S_{4242}, S_{1213}, S_{1243}, S_{4213}, S_{4243}, S_{1221}, S_{1251}, S_{4221}, S_{4251}, S_{1223}, S_{1253}, S_{4223}, S_{4253}, S_{1224}, S_{1254}, S_{4224}, S_{4254}, S_{1301}, S_{1331}, S_{4301}, S_{4331}, S_{1302}, S_{1332}, S_{4302}, S_{4332}, S_{1304}, S_{1334}, S_{4304}, S_{4334}, S_{1312}, S_{1342}, S_{4312}, S_{4342}, S_{1313}, S_{1343}, S_{4313}, S_{4343}, S_{1321}, S_{1351}, S_{4321}, S_{4351}, S_{1323}, S_{1353}, S_{4323}, S_{4353}, S_{1324}, S_{1354}, S_{4324}, S_{4354}, S_{2101}, S_{2131}, S_{5101}, S_{5131}, S_{2102}, S_{2132}, S_{5102}, S_{5132}, S_{2104}, S_{2134}, S_{5104}, S_{5134}, S_{2112}, S_{2142}, S_{5112}, S_{5142}, S_{2113}, S_{2143}, S_{5113}, S_{5143}, S_{2121}, S_{2151}, S_{5121}, S_{5151}, S_{2123}, S_{2153}, S_{5123}, S_{5153}, S_{2124}, S_{2154}, S_{5124}, S_{5154}, S_{2301}, S_{2331}, S_{5301}, S_{5331}, S_{2302}, S_{2332}, S_{5302}, S_{5332}, S_{2304}, S_{2334}, S_{5304}, S_{5334}, S_{2312}, S_{2342}, S_{5312}, S_{5342}, S_{2313}, S_{2343}, S_{5313}, S_{5343}, S_{2321}, S_{2351}, S_{5321}, S_{5351}, S_{2323},S_{2353}, S_{5323},S_{5353}, S_{2324}, S_{2354}, S_{5324}, S_{2401}, S_{2431}, S_{5401},S_{5431}, S_{2402}, S_{2432}, S_{5402}, S_{5432}, S_{2404}, S_{2434}, S_{5404}, S_{5434}, S_{2412}, S_{2442}, S_{5412}, S_{5442}, S_{2413}, S_{2443}, S_{5413}, S_{5443}, S_{2421}, S_{2451}, S_{5421}, S_{5451}, S_{2423}, S_{2453}, S_{5423}, S_{5453}, S_{2424}, S_{2454}, S_{5424}, S_{5454}, S_{5354} (from all even states).

	S_{0001}, S_{0012}, S_{0021}, S_{0023}, S_{0054} - S_{0154}, S_{0201}, S_{0203}, S_{0210}, S_{0212}, S_{0214}, S_{0221}, S_{0223}, S_{0230}, S_{0232}, S_{0234}, S_{0241}, S_{0243}, S_{0250}, S_{0252}, S_{0254}, S_{0301}, S_{0302}, S_{0304}, S_{0310}, S_{0312}, S_{0313}, S_{0320}, S_{0321}, S_{0323}, S_{0324}, S_{0331}, S_{0332}, S_{0334}, S_{0340}, S_{0342}, S_{0343}, S_{0350}, S_{0351}, S_{0353}, S_{0354}, S_{0401}, S_{0403}, S_{0410}, S_{0412}, S_{0414}, S_{0421}, S_{0423}, S_{0430}, S_{0432}, S_{0434}, S_{0441}, S_{0443}, S_{0450}, S_{0452}, S_{0454}, S_{1000} - S_{1004}, S_{1011} - S_{1014}, S_{1021} - S_{1024}, S_{1031} - S_{1034}, S_{1041} - S_{1044}, S_{1051} - S_{1054}, S_{1101},S_{1110}, S_{1112},S_{1121}, S_{1123},S_{1132}, S_{1134},S_{1143}, S_{1150},S_{1154}, S_{1201} – S_{1254}, S_{1301},S_{1303}, S_{1310}, S_{1312}, S_{1314}, S_{1321},S_{1323},S_{1330}, S_{1332}, S_{1334}, S_{1341}, S_{1343}, S_{1350}, S_{1352}, S_{1354}, S_{1400}, S_{1401}, S_{1402}, S_{1404}, S_{1410}, S_{1412}, S_{1413}, S_{1420}, S_{1421}, S_{1423}, S_{1424}, S_{1431}, S_{1432}, S_{1434}, S_{1440}, S_{1442}, S_{1443}, S_{1450},S_{1451}, S_{1453},S_{1454}, S_{2001}, S_{2003}, S_{2012}, S_{2014}, S_{2021}, S_{2023}, S_{2032}, S_{2034}, S_{2041}, S_{2043}, S_{2052},S_{2054}, S_{2100} - S_{2154}, S_{2201},S_{2210}, S_{2212},S_{2221}, S_{2223},S_{2232}, S_{2234},S_{2243}, S_{2250},S_{2254}, S_{2300} - S_{2354}, S_{2401},S_{2403}, S_{2410},S_{2412}, S_{2414},S_{2421}, S_{2423},S_{2430}, S_{2432},S_{2434}, S_{2441},S_{2443}, S_{2450},S_{2452},S_{2454}, S_{3001},S_{3002}, S_{3004}, S_{3012}, S_{3013}, S_{3021}, S_{3023}, S_{3024}, S_{3032}, S_{3034}, S_{3042}, S_{3043}, S_{3051}, S_{3053}, S_{3054}, S_{3101}, S_{3103}, S_{3110}, S_{3112}, S_{3114}, S_{3121}, S_{3123}, S_{3130}, S_{3132}, S_{3134}, S_{3141},S_{3143}, S_{3150}, S_{3152}, S_{3154}, S_{3200} - S_{3254}, S_{3301},S_{3310}, S_{3312}, S_{3321}, S_{3323}, S_{3332}, S_{3334}, S_{3343}, S_{3350},S_{3354}, S_{3400} - S_{3454}, S_{4001},S_{4003}, S_{4010},S_{4012}, S_{4014},S_{4021}, S_{4023},S_{4030}, S_{4032},S_{4034}, S_{4041},S_{4043}, S_{4050},S_{4052}, S_{4054},S_{4101},S_{4102}, S_{4104}, S_{4110}, S_{4120},S_{4121}, S_{4123},S_{4124}, S_{4131}, S_{4132},S_{4134},S_{4140}, S_{4142}, S_{4143}, S_{4150}, S_{4151},S_{4153},S_{4154}, S_{4201},S_{4203}, S_{4210}, S_{4212}, S_{4214}, S_{4221}, S_{4223},S_{4230}, S_{4232}, S_{4234}, S_{4241}, S_{4243}, S_{4250}, S_{4252}, S_{4254}, S_{4300} - S_{4354}, S_{4401}, S_{4410}, S_{4412}, S_{4421}, S_{4423}, S_{4432}, S_{4434}, S_{4443}, S_{4450}, S_{4454}, S_{5001}- S_{5004}, S_{5011}- S_{5014}, S_{5021}- S_{5024}, S_{5031}- S_{5034}, S_{5041}- S_{5044}, S_{5051}- S_{5054}, S_{5101}, S_{5103}, S_{5110}, S_{5112}, S_{5114}, S_{5121}, S_{5123}, S_{5130}, S_{5132}, S_{5134}, S_{5141},S_{5143}, S_{5150}, S_{5152}, S_{5154}, S_{5201}, S_{5202}, S_{5204}, S_{5210}, S_{5212}, S_{5213}, S_{5220}, S_{5221}, S_{5223}, S_{5224}, S_{5231}, S_{5232}, S_{5234}, S_{5240}, S_{5242}, S_{5243}, S_{5250}, S_{5251}, S_{5253}, S_{5254}, S_{5301}, S_{5303}, S_{5310}, S_{5312}, S_{5314}, S_{5321}, S_{5323}, S_{5330}, S_{5332}, S_{5334}, S_{5341},S_{5343}, S_{5350}, S_{5352}, S_{5354}, S_{5400} - S_{5454}.(These which are not divisible by 2 or 3 or 5)	S_{0000}, S_{0030}, S_{3000}, S_{3030}(from those state which are not divisible by 2 or 3 or 5)

From Table B, it is clear that the probabilities 1/4, 1/9 and 1/25 occur for transitions from maximum number of the states to states S_{0000}, S_{0030}, S_{3000} and S_{3030}, that is 0, 15, 450 and 465. Hence we can deduce the following probabilities, using $\pi_\infty(i)$ [for $i = 0, 15, 450$ and 465] = $7.07\text{x}10^{-2}$ and $\pi_\infty(i)$[for other values of i] = $2.87\text{x}10^{-3}$.

p_5 = the probability to perform a division by 5 = $\sum_{i=0}^{29}\sum_{j=0}^{5}\sum_{k=0}^{4}\pi_\infty\ (30i+5j+k)$, [if i,j,k ≡ 0 (mod 5)]
= 2. $\pi_\infty(i)$ [for $i = 0$ and 450]+ 10. $\pi_\infty(i)$[for other values of i]
= $2\text{x}7.07\text{x}10^{-2} + 10\text{x}\ 2.87\text{x}10^{-3}$
= $0.1414 + 0.0287 = 0.1701\text{x}10^{-4}$ = 43/250

p_3 = the probability to perform a division by 3 = $\sum_{i=0}^{29}\sum_{j=0}^{5}\sum_{k=0}^{4}\pi_\infty\ (30i+5j+k)$, [if i,j,k ≡ 0 (mod 3)]
= 4. $\pi_\infty(i)$ [for $i = 0, 15, 450$ and 465] + 20. $\pi_\infty(i)$[for other values of i]
= $4\text{x}7.07\text{x}10^{-2} + 20\text{x}\ 2.87\text{x}10^{-3}$
= $0.2828 + 0.0574 = 3402\text{x}10^{-4} \approx 73/250$

p_z = the probability to obtain zero column = $\sum_{i=0}^{29}\sum_{j=0}^{5}\sum_{k=0}^{4}\pi_\infty\ (30i+5j+k)$,
[if i,j,k ≡ 0(mod 5) or i,j,k ≡ 0(mod 3)or i,j,k ≡ 0(mod 2)]

= 4. $\pi_\infty(i)$ [for $i = 0, 15, 450$ and 465] + 159. $\pi_\infty(i)$[for other values of i]
= 4x7.07x10^{-2} + 159x 2.87x10^{-3}
= 0.2828 + 0.4564 = 7391x10^{-4} = 7391.3x10^{-4} ≈ 185/250

p_2 = the probability to perform a division by 2 = 1 – (p_5 + p_3) = 1 – (43/250 + 73/250) = 134/250

p_{nz} = the probability to obtain non − zero column = 1- p_z = 1 – 185/250 = 65/250