社団法人　情報処理学会　研究報告
IPSJ SIG Technical Report

2005－ＣＳＥＣ－30（49）
2005／7／22

ルーティングを用いた素因数分解回路について

伊豆　哲也[†]　　國廣　　昇[††]　　太田　和夫[††]　　下山　武司[†]

[†] 富士通株式会社, 〒 211-8588 川崎市中原区上小田中 4-1-1
[††] 電気通信大学 情報通信工学科, 〒 182-8585 調布市調布ヶ丘 1-5-1
E-mail: [†]{izu,shimo}@jp.fujitsu.com, [††]{kunihiro,ota}@ice.uec.ac.jp

あらまし　近年, RSA ベースの PKI に対する新しい脅威として, 素因数分解専用ハードウェアが注目を集めている. いくつかの装置で使用されている時計回り置換というルーティングアルゴリズムは停止性に対する証明を欠いており, 例外処理のための追加装置を必要としている. 本稿は時計回り置換におけるパケットの交換ルールを解析し, “farthest-first” という性質を満たすいくつかの代替交換法を提案する. 理論的な証明はないものの, 実験的にはこれまでの提案アルゴリズムよりも優れた振る舞いを見せている. また関係式探索ステップに対し, 部分トーラスを用いた実装法も提案する.
キーワード　素因数分解, ASIC, RSA, 時計回り置換, YASD

Analysis on the Clockwise Transposition Routing for Dedicated Factoring Devices

Tetsuya IZU[†], Noboru KUNIHIRO[††], Kazuo OHTA[††], and Takeshi SHIMOYAMA[†]

[†] FUJITSU Limited, 4-1-1, Kamikodanaka, Nakahara-ku, Kawasaki, 211-8588, Japan
[††] Dept. of Information and Communication Eng., The University of Electro-Communications, 1-5-1, Choufugaoka, Chofu, 182-8585, Japan
E-mail: [†]{izu,shimo}@jp.fujitsu.com, [††]{kunihiro,ota}@ice.uec.ac.jp

Abstract　Recently, dedicated factoring devices have attracted much attention since it might be a threat for a current RSA-based PKI. In some devices, the clockwise transposition is used as a key technique, however, because of the lack of theoretic proof of the termination, some additional circuits are required. In this paper, we analyze the packet exchanging rule for the clockwise transposition and propose some possible alternatives with keeping the “farthest-first” property. Although we have no theoretic proof of the termination, experimental results show actual availability in the clockwise transposition. We also propose an improvement on the routing algorithm for the relation finding step, which establishes two times speed-up.
Key words　Integer factoring, ASIC, RSA, clockwise transposition, YASD

1. Introduction

The integer factoring problem is one of the most fundamental topics in the area of cryptology since the hardness of this problem assures the security of some public-key cryptosystems such as the famous RSA. The number field sieve method (NFS) [LLPM90] is the best algorithm for integer factoring [*1]. NFS has 4 major steps, the polynomial selection step, the Relation Finding (RF) step or the sieving step, the Linear Algebra (LA) step, and the final step. Among them, RF and LA steps are theoretically and experimentally dominant steps. Because of these steps, factoring 1024-bit integers is considered infeasible in next 10 years (by the same approach).

In order to overcome the difficulty, ASIC-based dedi-

(*0): A part of this work is financially supported by a consignment research from the National Institute of Information and Communications Technology (NICT), Japan.

(*1): Very recently, a new world record, a 663-bit integer was factored by NFS [RSA200].

cated factoring devices have been studied actively. In 2001, Bernstein employed a sorting algorithm for LA step with standard ASIC architectures [Ber01]. Then Lenstra et al. enhanced the device by using a routing algorithm [LSTT02]. Furthermore, the design is substantially improved by Geiselmann-Steinwandt [GS03b] On the other hand, Geiselmann-Steinwandt applied these algorithms to RF step, and proposed two designs DSH [GS03a] and YASD [GS04]. Shamir-Tromer improved an optical sieving device TWINKLE [Sha99] into a novel ASIC-based device TWIRL [ST03]. Both YASD and TWIRL handle RF step corresponding to 768-bit integers, properties are quite different: the speed of TWIRL is about 6.3 times faster than YASD, but required circuit area of YASD is smaller than that of TWIRL. Recently, Franke et al. proposed a challenging device SHARK for RF step based on the lattice sieving [FKP+05]. By these contributions, it is expected that the linear algebra step is easily processed compared to the relation finding step in factoring large integers.

A purpose of this paper is to analyze the clockwise transposition routing used in [LSTT02], [GS04], since there is no theoretic proof of the termination in finite steps. In fact, Geiselmann et al. showed a concrete "livelock" example for which the algorithm falls into infinite loop. We show possible alternatives for packet exchanging rules in the routing. Althogh we have no theoretic proof of the termination, experimental results show the availability of these alternatives. We also propose an improvements of the clockwise transposition for RF step, a use of sub-tori in the routing, which possibly establishes two times speed-up.

This paper is organized as follows: in section 2, we briefly introduce the clockwise transposition routing, and analyze the packet exchanging rules in section 3. Section 4 shows our experimental results. We also propose an improvement of YASD in section 5.

2. Preliminaries

This section briefly introduces the clockwise transposition routing [LSTT02].

2.1 Clockwise Transposition Routing

In some factoring devices for both RF and LA steps, the clockwise transposition routing algorithm on a mesh is used as a key technique. A *mesh* has a set of $m \times m$ processors (called *nodes*) in a two-dimensional network. Each node is connected to its upper, right, lower, and left nodes (if exist) and is able to hold a *packet*, a paired data of value and target node (to be routed), or Nil. When a packet is reached its target node, the data is took into the node and the packet is changed to Nil, and exchange the packet to its one of the neighbors. For a given mesh filled with packets, a purpose of the routing is to deliver all packets to their target nodes. Since we are intereted in the behavior of packets, we omit describing packet values. Moreover, for simplicity, we identify a node in the i-th row and the j-th column as (i, j) $(0 \leqq i, j < m)$.

The clockwise transposition routing [LSTT02] is an algorithm for the routing problem. The algorithm is proceeded by a repetition of the following 4 steps until all packets are delivered to their target nodes (also see Table 1), where t denotes the time:

$t \equiv 0 \pmod 4$ For every column j and odd row i, a node (i, j) compares and exchanges packets to its upper node $(i-1, j)$ (if exist).

$t \equiv 1 \pmod 4$ For every row i and odd column j, a node (i, j) compares and exchanges packets to its right node $(i, j+1)$ (if exist).

$t \equiv 2 \pmod 4$ For every column j and odd row i, a node (i, j) compares and exchanges packets to its lower node $(i+1, j)$ (if exist).

$t \equiv 3 \pmod 4$ For every row i and odd column j, a node (i, j) compares and exchanges packets to its left node $(i, j-1)$ (if exist).

Compared directions changes in "clockwise" manner for nodes in odd rows and odd columns. Here, the packet exchanging is ruled by "farthest-first exchange" [LSTT02]. Details of the rules are discussed in the following sections.

Unfortunately, there is no theoretic proof whether the clockwise transposition terminates in finite steps or not. However, it is claimed that the algorithm terminates in $2m$ steps with high probability [LSTT02]. Although Geiselmann et al. showed a concrete "livelock" example for which the algorithm falls into an infinite loop, and a "pathology" for which the algorithm does not terminate in $2m$ steps, experimental results show that such exceptional cases are very rare in factoring so that we can expect the termination of the algorithm with high probability.

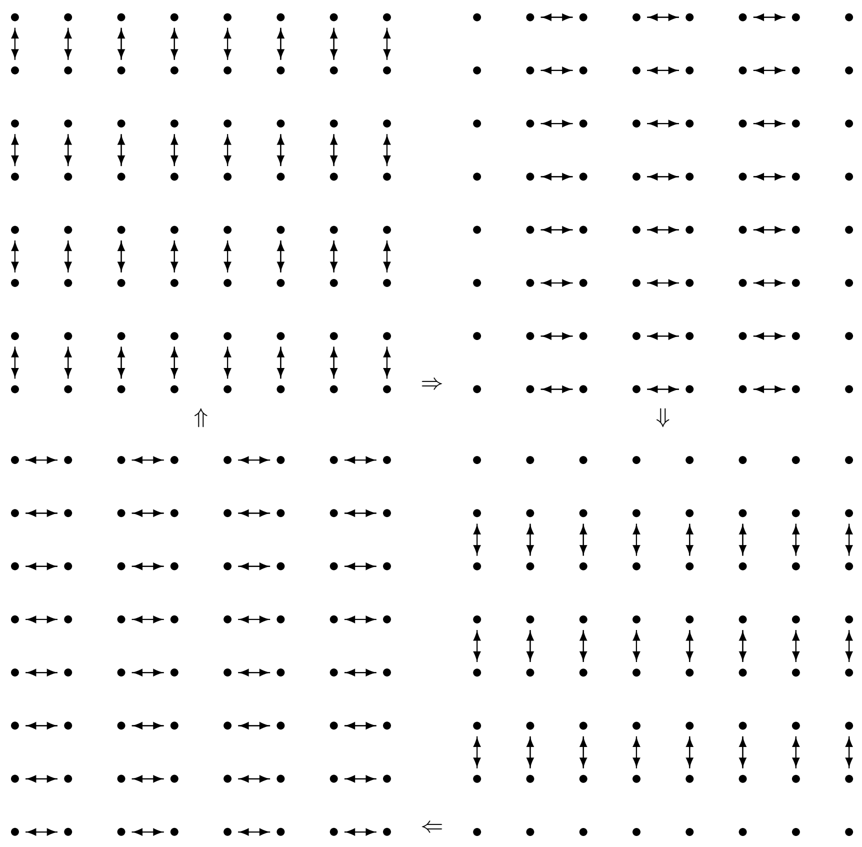
2.2 Torus

In [GS04], Geiselmann-Steinwandt added a torus structure into the mesh: the leftmost nodes are connected to the rightmost nodes and the uppermost nodes are connected to the lowermost nodes. In this torus structure, since the maximum distance from a node to its target becomes half, the clockwise transposition for RF step only requires reduced steps. This structure is also applicable to LA step [GKST05]. In the followings, a mesh with this torus structure is described as a torus. Strongly note that the wiring problem for realizing the torus structure is not a serious problem [GS04], [GKST05].

3. Packet Exchange

This section analyzes packet exchanging rules for the clock-

Table 1 Clockwise transposition routing



wise trasnposition routing. The rule is very naive in terms of that the termination of the algorithm depends on many factors including the exchanging rule and initial packets. In addition, the beginning step also has an effect on the termination.

We denote a target node of the node (i, j) as $p(i, j)$ in the followings.

3.1 Farthest-first Rule

The packet exchanging rule for the clockwise transpotion routing is described as follows [GKST05]: suppose we are comparing two nodes (i, j) and $(i, j + 1)$ horizontally. We exchange packets when

- $p(i, j) =$ Nil and $j + 1 > j_1$
- $p(i, j + 1) =$ Nil and $j_0 > j$
- $p(i, j),\ p(i, j + 1) \neq$ Nil and $j_0 \geqq j_1$

where $p(i, j) = (i_0, j_0),\ p(i, j + 1) = (i_1, j_1)$ if they are not Nil. The 3rd rule is described as the "farthest-first along the direction" rule [GKST05]. For virtical cases, the similar rule is easily established. Note that in the original rule [LSTT02], the 3rd condition was given as "$j_0 > j_1$". However, this condition does not work well for the livelock and the pathology examples described in the next section.

3.2 Livelock and Pathology Examples

Geiselmann et al. showed a livelock for which the routing algorithm fails into an finite loop, and a pathology for which the routing algorithm requires more than $2m$ steps on an $m \times m$ mesh [GKST05]. The liverock and the pathology for $m = 4$ are in Table 2. In fact, the routing actually falls into a $4m$-step loop for the livelock.

As we have mentioned, the clockwise transposition is naive: for the livelock example, the algorithm terminates if we change the first step from 'upper' procedure to other procedures.

Treatments for such livelock cases differs in RF step and LA step. In RF step, packets in an infinite loop can be omitted since the step does not require all packets [GS04]. However, on the other hand, in LA step, any omission is not permitted. Thus Geiselmann et al. proposed additional circuits to treat such cases [GKST05].

3.3 Exchanging Rule in Torus

In order to apply the clockwise transposition in the torus, the exchanging rule should be modified. Geiselman et al. proposed the following algorithm [GKST05]: before the routing, search the shortest paths of all packets to their target nodes. If the path crosses the borders (i.e. wires between the leftmost and the rightmost nodes, and the uppermost and the lowermost nodes) add (subtract) m to (from) corresponding addresses of target nodes. Then apply the same exchanging rule as before.

However, this modification does not solve the termination problem either theoretically nor experimentaly. Thus

Table 2 The livelock and the pathology example [GKST05]

Livelock ($m = 4$)

	0	1	2	3
0	(3,0)	(2,0)	(1,0)	(0,0)
1	(3,1)	(2,1)	(1,1)	(0,1)
2	(3,2)	(2,2)	(1,2)	(0,2)
3	(3,3)	(2,3)	(1,3)	(0,3)

Pathology ($m = 4$)

	0	1	2	3
0	(0,0)	(1,0)	(2,0)	(3,0)
1	(0,1)	(1,1)	(2,1)	(3,1)
2	(0,2)	(1,2)	(2,2)	(3,2)
3	(0,3)	(1,3)	(2,3)	(3,3)

we would like to explore other possible rules with keeping the "farthest-first" property.

3.4 Rephrasing Exchanging Rule

In order to use the clockwise transposition routing in the factoring devices, exchanging rules without any livelocks are required. Thus we start from rephrasing the previous "farthest-first rule" by a 1-dimensional ℓ_0 distance function.

Suppose we are comparing two nodes (i, j) and $(i, j+1)$ with their target nodes being non-Nɪʟ, namely $p(i,j) = (i_0, j_0)$, $p(i, j+1) = (i_1, j_1)$. Intuitively, the farthest-first rule along rows should be described as

(1) $|j_1 - (j+1)| > |j_0 - j|$ and $|j_1 - (j+1)| \geqq |j_1 - j|$, or

(2) $|j_0 - j| > |j_1 - (j+1)|$ and $|j_0 - j| \geqq |j_0 - (j+1)|$.

However, and interestingly, these conditions are not equivalent to the 3rd condition $j_0 \geqq j_1$. In fact, we have the following proposition, which is a beginning of our discussion. All proofs of the following propositions are omitted, since most of them are obtained by elementary arithmetics.

Suppose a node (i, j) has a packet with its target node (i_0, j_0) and a node $(i, j+1)$ has a packet with its target node (i_1, j_1). Then the following "farthest-first along the compared direction" rule

(1) $|j_1 - (j+1)| > |j_0 - j|$ and $|j_1 - (j+1)| \geqq |j_1 - j|$, or

(2) $|j_0 - j| > |j_1 - (j+1)|$ and $|j_0 - j| \geqq |j_0 - (j+1)|$,

is equivalent to $j_0 \geqq j_1$ and $j_0 + j_1 \neq 2j + 1$. Especially, the rule is not equivalent to the condition $j_0 \geqq j_1$.

On the other hand, the following very similar conditions are not equivalent to the 3rd condition $j_0 \geqq j_1$ either.

In the same assumption to Proposition 1, conditions

(1) $|j_1 - (j+1)| \geqq |j_0 - j|$ and $|j_1 - (j+1)| \geqq |j_1 - j|$, or

(2) $|j_0 - j| \geqq |j_1 - (j+1)|$ and $|j_0 - j| \geqq |j_0 - (j+1)|$,

are equivalent to $j_0 + 1 \geqq j_1$. Especially, the rule is not equivalent to the condition $j_0 \geqq j_1$.

In proposition 1, 2, distances between nodes are measured by the ℓ_0-distance, namely absolute values of the difference of coordinate values. By changing the distance function, we have other possible alternatives for the "farthest-first" rule as follows:

(a) 1. $d((i,j+1), p(i,j+1)) > d((i,j), p(i,j))$ and $d((i,j+1), p(i,j+1)) \geqq d((i,j), p(i,j+1))$, or

2. $d((i,j), p(i,j)) > d((i,j+1), p(i,j+1))$ and $d((i,j), p(i,j)) \geqq d((i,j+1), p(i,j))$,

(a') 1. $d((i,j+1), p(i,j+1)) \geqq d((i,j), p(i,j))$ and $d((i,j+1), p(i,j+1)) \geqq d((i,j), p(i,j+1))$, or

2. $d((i,j), p(i,j)) \geqq d((i,j+1), p(i,j+1))$ and $d((i,j), p(i,j)) \geqq d((i,j+1), p(i,j))$.

By a definition $d((i,j), \textsc{Nil}) = 0$, the above rules include Nɪʟ cases. Thus we can describe the exchanging rule mathematically. Moreover, by applying other distance functions $d(\cdot,\cdot)$, other exchanging rules can be obtained. In the followings, we use 4 distance functions on an $m \times m$ mesh or torus:

- 1-dimensional distance in a mesh: $d_1^{\mathrm{m}}((i,j),(i',j')) = |i - i'|$ or $|j - j'|$
- 1-dimensional distance in a mesh: $d_1^{\mathrm{t}}((i,j),(i',j')) = \min(|i - i'|, m - |i - i'|)$ or $\min(|j - j'|, m - |j - j'|)$
- 2-dimensional distance in a mesh: $d_2^{\mathrm{m}}((i,j),(i',j')) = |i - i'| + |j - j'|$
- 2-dimensional distance in a mesh: $d_2^{\mathrm{t}}((i,j),(i',j')) = \min(|i - i'|, m - |i - i'|) + \min(|j - j'|, m - |j - j'|)$

Note that $0 \leqq d_1^{\mathrm{m}}(\cdot,\cdot) < m$, $0 \leqq d_1^{\mathrm{t}}(\cdot,\cdot) < m/2$, $0 \leqq d_2^{\mathrm{m}}(\cdot,\cdot) < 2m$, and $0 \leqq d_2^{\mathrm{t}}(\cdot,\cdot) < m$.

Let us consider other possible way to rephrase the 3rd condition $j_0 \geqq j_1$. In Proposition 1 and 2, the additinal conditions $j_0 + j_1 = 2j + 1$ and $j_1 = j_0 + 1$ imply $d^{\mathrm{before}} = d^{\mathrm{after}}$, where $d^{\mathrm{before}} = d((i,j), p(i,j)) + d((i,j+1), p(i,j+1)) = d((i,j)$, $d^{\mathrm{after}} = d((i,j), p(i,j+1)) + d((i,j+1), p(i,j))$. By treating this case seperately, we have the following satisfactory rule equivalent to $j_0 \geqq j_1$ for $d = d_1^{\mathrm{m}}$.

(b) 1. $d^{\mathrm{before}} > d^{\mathrm{after}}$, or

2. $d^{\mathrm{before}} = d^{\mathrm{after}}$ and $d((i,j+1), p(i,j+1)) > d((i,j), p(i,j))$ and $d((i,j+1), p(i,j+1)) \geqq d((i,j), p(i,j+1))$, or

3. $d^{\mathrm{before}} = d^{\mathrm{after}}$ and $d((i,j), p(i,j)) > d((i,j+1), p(i,j+1))$ and $d((i,j), p(i,j)) \geqq d((i,j+1), p(i,j))$.

We also have the following similar conditions:

(b') 1. $d^{\mathrm{before}} > d^{\mathrm{after}}$, or

2. $d^{\mathrm{before}} = d^{\mathrm{after}}$ and $d((i,j+1), p(i,j+1)) \geqq d((i,j), p(i,j))$ and $d((i,j+1), p(i,j+1)) \geqq d((i,j), p(i,j+1))$, or

3. $d^{\mathrm{before}} = d^{\mathrm{after}}$ and $d((i,j), p(i,j)) \geqq d((i,j+$

$1), p(i, j+1))$ and $d((i,j), p(i,j)) \geqq d((i, j+1), p(i,j))$.

Consequently, we obtain 4 mathematical descriptions of the "farthest-first" rule and 4 distance functions, namely 16 possible exchanging rules. Since we have no theoretic proofs, terminations are not assured. However, experimental results and practical availability will be shown in the next section.

4. Experimental Results

This section shows experimental results of some packet exchanging rules. In the previous section, we established 4 alternative exchanging rules and 4 distance functions, namely 16 rules. With these rules plus the original 2 rules, we compute required steps in some cases.

Firstly, we routined the livelock example under these exchanging rules and with changing the initial step. Numerical results are summarized in Table 3 ($m = 4$) and Table 4 ($m = 8$), where 'u', 'r', 'lo', and 'le' stands for upper, right, lower, and left step as an initial step, respectively, and 'NT' stands for non-termination. As described in [GKST05], the naiveness of the algorithm can be observed. For example, changing an initial step has an effect of the termination. Interstingly, begining from 'u' inclines to fall into 'NT'. Compared to these results, the number of NT seems more in Table 3. But this may be because of the smallness of m and does not show any algorithmic defects.

Next, we routined a mesh filled with m^2 non-NIL packets with $m = 8$ (Table 5), and m^2 non-NIL packets with $m = 8$. Of courese, although these are just examples, we can observe some properties. First, the original rule with d_1^{m} works well in a sense that it does not fall into 'NT'. Second, rules (a), (a') work worse than (b), (b'). Moreover, an effect of the torus structure is observed. In these examples, rules (b), (b') combined with distance functions d_1^{t}, d_2^{t} work better than other cases. However, changing initial step seems to have less effect here.

5. Improvements

The clockwise transposition is used for both RF and LA steps [LSTT02], [GS04]. However, the efficiency of RF step case, YASD, is not compatible to TWIRL: YASD is 6.3 times slower than TWIRL without considering the frequency and 3.2 times with considering the frequency. This section proposes an improvement on YASD, which establishes two times speed-up.

5.1 Structure of YASD

First, we describe procedures in RF step. Suppose we are going to find relations (a, b) from a given interval $[a_0, a_0+S-1]$ and a fixed value b (here we assume a_0 being even without loss of generality), where a pair (a, b) is called a relation if it satisfies three conditions (i) $\gcd(a, b) = 1$, (ii) $F_{\mathrm{r}}(a, b)$ is B_{r}-smooth for a given multivariable polynomial $F_{\mathrm{r}}(x, y)$ and an integer B_{r}, and (iii) $F_{\mathrm{a}}(a, b)$ is B_{a}-smooth for a given multivariable polynomial $F_{\mathrm{a}}(x, y)$ and an integer B_{a}. An integer x is described as B-smooth if x is a product of prime integers smaller than B. Since $\log x \approx \sum_{p<B,\ p|x} \log p$, the sieving method for RF step proceeds as follows: first, we prepare S registers $s[a_i]$ $(a_i \in [a_0, a_0+S-1])$. For each prime $p < B$, find the smallest integer $\overline{a} \in [a_0, a_0+S-1]$ such that $F(\overline{a}, b) = 0 \pmod p$, here $F(x, y) = F_{\mathrm{r}}(x, y)$ or $F_{\mathrm{a}}(x, y)$, and $B = B_{\mathrm{r}}$ or B_{a}. Since the polynomial $F(x, y)$ has a property that

$$F(a, b) = 0 \pmod p \quad \Rightarrow \quad F(a+p, b) = 0 \pmod p,$$

we set $s[\overline{a}] \leftarrow s[\overline{a}] + \log p$, $s[\overline{a}+p] \leftarrow s[\overline{a}+p] + \log p, \ldots$. Finally, pick up a's such that $s[a] \approx \log F(a, b)$, which can be treated as candidates as the relations.

YASD is a dedicated factoring device for RF step by using the clockwise transposition [GS04]. Each node has three parts, the main part, the mesh part, and the memory part. The main part generates pairs (a, p) such that $F(a, b) = 0 \pmod p$ as packets and these packets are sent to the mesh part. The mesh part proceeds the clockwise routing as in the previous sections. When a packet reaches the target node, it is delivered to the memory part which consists of u registers $s[a_1], \ldots, s[a_1+u-1]$ and the log value $\log p$ is accumulated to the corresponding register $s[a]$. Note that there is no need to hold all primes in all nodes: In fact it is sufficient to hold at least 1 node for large primes.

5.2 Use of Sub-torus

This section proposes to use sub-tori for RF step similar to for LA step proposed in [GKST05]. For an $m \times m$ mesh, we divide nodes into 4 sets $T^{\mathrm{o,o}}$, $T^{\mathrm{o,e}}$, $T^{\mathrm{e,o}}$, $T^{\mathrm{e,e}}$, nodes in odd-rows and odd-columns, nodes in odd-rows and even-columns, nodes in even-rows and odd-columns, and nodes in even-rows and even-columns, respectively. Then, we give the torus structure to these sets. Thus we have 4 sub-tori in the mesh. Since the size of sub-tori is halved, efficient routings on these sub-tori are expected. But a problem arises: how to generate packets in which a packet with its target node being odd-odd, for example, is sent to an odd-odd node from a main part. One idea is to let 1 main part to hold 4 nodes (odd-odd, odd-even, even-odd, and even-even). Then all packets can be easily sent to the proper sub-torus. However the frequency is reduced to 1/4. So, we do not want to change the number of main parts and nodes.

For this problem, we have an algorithmic and hardware-oriented solutions. In YASD, each prime is held by at least 1 node. We increase the frequency of each prime 4 times so that all primes can be sent to all types of sub-tori. As a drawback we require 4 times larger memory for main parts

Table 3 Required steps for the livelock example ($m = 4$)

Distance	Original				(a)				(a')				(b)				(b')			
Function	u	r	lo	le	u	r	lo	le	u	r	lo	le	u	r	lo	le	u	r	lo	le
d^{m}_1	NT	15	14	19	NT	15	14	19	NT	23	22	21	NT	15	14	19	NT	15	14	21
d^{m}_2		—			NT	NT	NT	NT	21	16	20	18	38	NT	NT	NT	17	12	19	15
d^{t}_1	NT	NT	NT	NT	NT	12	11	14	NT	23	7	25	NT	15	7	17	NT	16	7	18
d^{t}_2		—			NT	NT	NT	NT	NT	11	7	13	NT	NT	NT	NT	NT	16	7	18

Table 4 Required steps for the livelock example ($m = 8$)

Distance	Original				(a)				(a')				(b)				(b')			
Function	u	r	d	l	u	r	d	l	u	r	d	l	u	r	d	l	u	r	d	l
d^{m}_1	NT	42	36	38	NT	NT	NT	58	NT	NT	NT	NT	NT	42	37	41	NT	40	41	44
d^{m}_2		—			NT	NT	NT	NT	41	44	39	46	93	NT	NT	NT	43	44	39	43
d^{t}_1	NT	NT	NT	NT	NT	NT	35	NT	NT	60	NT	NT	NT	28	26	32	NT	34	26	37
d^{t}_2		—			62	NT	77	NT	33	32	33	37	40	NT	31	NT	25	31	21	26

Table 5 Required steps for a mesh with m^2 non-Nil packets ($m = 8$)

Distance	Original				(a)				(a')				(b)				(b')			
Function	u	r	d	l	u	r	d	l	u	r	d	l	u	r	d	l	u	r	d	l
d^{m}_1	24	22	21	20	45	40	NT	NT	NT	NT	43	NT	22	22	25	22	31	24	22	26
d^{m}_2		—			NT	NT	NT	NT	30	44	33	30	39	38	28	34	31	28	28	27
d^{t}_1	NT	NT	NT	NT	NT	36	39	NT	NT	NT	NT	NT	19	23	18	21	20	22	21	23
d^{t}_2		—			NT	NT	NT	NT	25	34	29	27	23	41	30	19	21	22	22	20

Table 6 Required steps for a mesh with $m^2/8$ non-Nil packets ($m = 8$)

Distance	Original				(a)				(a')				(b)				(b')			
Function	u	r	d	l	u	r	d	l	u	r	d	l	u	r	d	l	u	r	d	l
d^{m}_1	16	17	17	16	NT	18	19	20	NT	NT	NT	NT	16	17	17	16	16	17	17	16
d^{m}_2		—			NT	18	19	15	16	20	19	20	16	17	17	16	16	17	17	16
d^{t}_1	NT	NT	NT	NT	NT	NT	17	11	NT	NT	35	NT	11	10	10	12	11	12	10	12
d^{t}_2		—			NT	NT	17	13	11	10	12	12	11	11	10	12	11	12	11	12

(to hold primes).

The other solution is to put a cyclic permutaion buffer device between 4 main parts and 4 nodes. Let us explain in detail. First, we change the memory part so that $T^{o,o}$ has registers corresponding to $a \in [a_0, a_0 + S - 1]$ such that $a \equiv 0 \bmod 4$. Similarly, $T^{\mathrm{o,e}}$, $T^{\mathrm{e,o}}$, $T^{\mathrm{e,e}}$ correspond to $a \equiv 1, 2, 3 \bmod 4$. Next, we divide primes (except 2) into two sets $P_1 = \{\, p \mid p \equiv 1 \bmod 4 \,\}$ and $P_3 = \{\, p \mid p \equiv 3 \bmod 4 \,\}$. Moreover, we divide these sets into 8 sets $P_1^{(i)} = \{\, p \in P_1 \mid p \equiv i \bmod 4 \,\}$, $P_3^{(i)} = \{\, p \in P_3 \mid p \equiv i \bmod 4 \,\}$ $(i = 0, 1, 2, 3)$. The numbers of primes in these sets $P_j^{(i)}$ will be almost same. Then, suppose a main part generates packets for a prime $p \in P_1^0$ and we have $\overline{a} \in [a_0, a_0 + S - 1]$ such that $F(\overline{a}, b) = 0 \pmod{p}$ and $\overline{a} \equiv 0 \pmod{4}$, Namely the target node of this packet belongs to $T^{\mathrm{o,o}}$. Then, since the next packet corresponds to $\overline{a} + p$, the target node of this packet belongs to $T^{\mathrm{o,e}}$. Thus the target node of this main part will be chaged in a cyclic way: $T^{\mathrm{o,o}}$, $T^{\mathrm{o,e}}$, $T^{\mathrm{e,o}}$, $T^{\mathrm{e,e}}$, $T^{\mathrm{o,o}}, \ldots$. Moreover, 4 main parets can generate packets in this way in simultaneously (see Fig. 1). Here the cyclic permutaion device can be easily implemented so that a drawback of this solution is rather small. By this improvement. we can establish the reduction about 1/2.

6. Concluding Remarks

This paper analyzes the dedicated hardware device based on the clockwise transposition. First, we discuss possible alternatives for packet exchanging. Although we have no theoretic proof of the termination, experimental results show actual availability of some exchanging rules in the clockwise transposition for integer factoring. We also proposed an improvement on the routing algorithm for the relation finding step, which establishes two times speed-up.

References

[Ber01] Daniel J. Bernstein, "Circuits for integer factorization: a proposal", preprint, 2001. `http://cr.yp.to/papers/nfscircuit.pdf`

[RSA200] F. Bahr, M. Boehm, J. Franke, and T. Kleinjung, "RSA200", May 2005. `http://www.crypto-world.com/announcements/rsa200.txt`

[FKP+05] Jens Franke, Thorsten Kleinjung, C. Paar, J. Pelzl, C. Priplata, C. Stahlke, "SHARK: A Realizable Special Hardware Sieving Device for Factoring 1024-bit Integers", *Workshop on Special Purpose hardware for Attacking Cryptographic Systems (SHARCS)*, pp.27-37, 2005. Also, to ap-

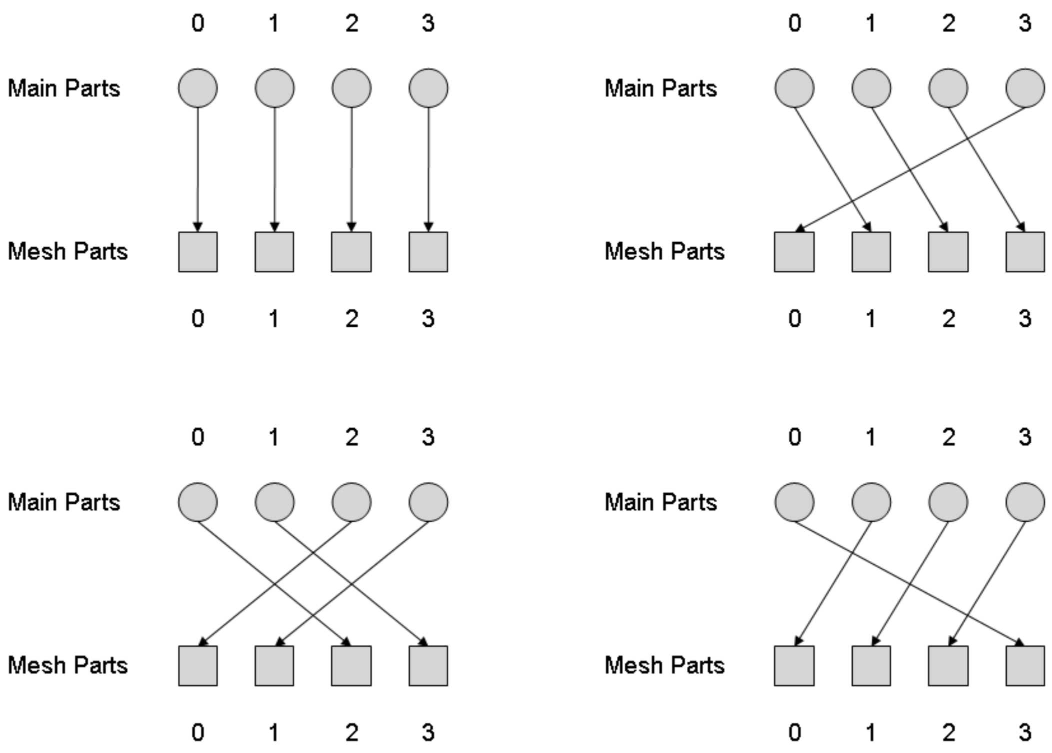


Figure 1 Cyclic wiring between main parts and mesh parts

pear at *CHES 2005*.

[GKST05] Willi Geiselmann, Hubert Köpfer, Rainer Steinwandt, and Eran Tromer, “Improved Routing-based Linear Algebra for the Number Field Sieve”, *IEEE ITCC 2005*, 2005.

[GS03a] Willi Geiselmann and Rainer Steinwandt, “A dedicated sieving hardware”, *PKC 2003*, LNCS 2567, pp.254-266, Springer-Verlag, 2003.

[GS03b] Willi Geiselmann and Rainer Steinwandt, “Hardware to Solve Sparse Systems of Linear Equations over GF(2)”, *CHES 2003*, LNCS 2779, pp.51-61, Springer-Verlag, 2003.

[GS04] Willi Geiselmann and Rainer Steinwandt, “Yet another sieveing device”, *CT-RSA 2004*, LNCS 2964, pp.278–291, Springer-Verlag, 2004.

[LL93] Arjen K. Lenstra and Hendrik W. Lenstra, editors. The development of the number field sieve, Vol. 1554 of Lecture Notes in Mathematics (LNM), Springer-Verlag, 1993.

[LLPM90] Arjen K. Lenstra, Hendrik W. Lenstra, M.S. Manasse and John M. Pollard, “The Number Field Sieve”, *STOC 1990*, pp.564-572, 1990.

[LSTT02] Arjen K. Lenstra, Adi Shamir, Jim Tomlinson, and Eran Tromer, “Analysis of Bernstein’s circuit”, *ASIACRYPT 2002*, LNCS 2501, pp.1–26, Springer-Verlag, 2002.

[Sha99] Adi Shamir, “Factoring large numbers with the TWINKLE device (extended abstract)”, *CHES 1999*, LNCS 1717, pp.2-12, Springer-Verlag, 1999.

[ST03] Adi Shamir and Eran Tromer, “Factoring large numbers with the TWIRL device”, *CRYPTO 2003*, LNCS 2729, pp.1–26, Springer-Verlag, 2003.