An Evaluation of the Sieving Device YASD for 1024-bit Integers [⋆] (Extended Abstract)

Naoyuki Hirota[1], Tetsuya Izu[2], Noboru Kunihiro[1] and Kazuo Ohta[1]

[1] The University of Electro-Communications,
1-5-1, Chofugaoka, Chofu, 182-8585, Japan
{hirota,kunihiro,ota}@ice.uec.ac.jp
[2] FUJITSU Limited,
4-1-1, Kamikodanaka, Nakahara-ku, Kawasaki, 211-8588, Japan
izu@jp.fujitsu.com

Abstract. In 2004, Geiselmann and Steinwandt proposed a hardware design "YASD" for the sieving step in the number field sieve (NFS) method of integer factorization. This design is very attractive since its regular structure is suitable for implementation, however, its performance evaluation for 1024-bit integers was not studied. This paper attempts to evaluate the performance of YASD for 1024-bit integers based on the same assumptions of the orginal YASD. In our estimation, YASD for 1024-bit integers requires at least 16 years with cost 10M dollars.

1 Introduction

The RSA public-key cryptosystem is widely used as a *de fact standard*. It is strongly believed that breaking RSA is as hard as factoring large, say 1024-bit or 2048-bit, integers. Thus the number field sieve (NFS) method, the best factoring algorithm, is an essential threat for the RSA cryptotsytem. Recently, NFS succeeded factoring a 633-bit integer by software implementation [RSA200]. Based on this fact, it is considered that factoring 1024-bit integers is infeasible in at least 10 years. However, this expectation does not take into account recent research results on dedicated factoring devices of NFS orginated from Berestein [Ber01]. NFS consists of 4 steps: the polynomial selection step, the relation finding (or sieving) step, the linear algebra step, and the fical step. Since the sieving step is the most time-cunsuming step, a lot of reserches have been focused on this step.

In 2003, Adi Shamir and Eran Tromer proposed an ASIC-based hardware design "TWIRL" for the sieving step [ST03]. According to their estimation, TWIRL could sieve within 90 days with cost 5000 dollars for a 768-bit integer. More surprisingly, TWIRL could factor even a 1024-bit integer within one year

[⋆] A part of this research is supported by a contract research with the National Institute of Information and Communications Technology (NICT), Japan

with cost 10M dollars (with 1GHz frequency). Since TWIRL uses various kinds of cuircuit (irregular structure), it requires an LSI as large as a full wafer with 300 mm diameter.

On the other hand, in 2004, Willi Geiselmann and Rainer Steinwandt proposed another ASIC-based design "YASD" [GT04]. YASD is very attractive because of its small chip size and regular structure. According to the authors' estimation, this device requires about 300 days with cost 5000 dollars for a 768-bit integer (with 500 MHz frequency). However, no estimation for 1024-bit integers have been provided up to the moment.

Contribution of This Paper

This paper focuses on YASD. In this paper, we evaluate the performance of YASD for 1024-bit integers (YASD1024) based on the same assumptions as those of TWIRL and YASD. Especially, we assumed that the process rule is 0.13 μm, the diameter of a full wafer is 300 mm, the manufacturing cost of a wafer is 5000 dollars. As a result of our estimation, YASD1024 requires at least 16 years with cost 10M dollars (with 500 MHz frequency). In the optimized case (in the sense of Area-Time product), YASD1024 requires an LSI as large as 411.17 cm^2 with $k = 28$ and $m = 10$ (these variables will be defined later). Since we do not consider wiring problem in detail in our estimation, YASD1024 reuires much more time or/and cost even if it is manufactured and proceeded. Thus our estimation may be little better than the best case analysis.

The rest of this paper is as follows: in section 2, we briefly review the number field sieve (NFS) method of factorization and the structure of YASD. Then we analyze and parameterize YASD in section 3 and 4. In section 5, we optimize these parameters for 1024-bit integers.

2 Preliminaries

In this section, we briefly review the number field sieve (NFS) method and the sieving device YASD for later discussion. For details of YASD, see the original paper [GT04].

2.1 Number Field Sieving Method

The number field sieving (NFS) method of integer factorization is the best algorithm, which consists of (1) the polynomial selection step, (2) the relation finding (or sieving) step, (3) the linear algebra step, and (4) the final step. Since (2) and (3) are most time-consuming steps in NFS, dedicated devices have been studied and proposed for these steps. In 2001, Bernstein employed a sorting algorithm for LA step with standard ASIC architectures [Ber01]. Then Lenstra, Shamir, Tomlinson and Tromer enhanced the device by using a routing algorithm [LSTT02]. On the other hand, Geiselmann and Steinwandt applied these algorithms to the

Table 1. Sieving parameter [LTSK+03]

	768-bit	1024-bit
H_a	1.7×10^{13}	5.5×10^{14}
H_b	8.9×10^{6}	2.7×10^{8}
B_r	10^{8}	3.5×10^{9}
B_a	10^{9}	2.6×10^{10}

sieving step, and proposed two designs DSH [GT03] and YASD [GT04]. Shamir and Tromer improved an optical sieving device TWINKLE [Sha99] into a novel ASIC-based device TWIRL [ST03]. Recently, Franke, Kleinjung, Parr, Pelzl, Priplata and Stahlke proposed a challenging device SHARK for the sieving step based on the lattice sieving [FKPP+05].

Let n be an integer to be factored. In the beginning of NFS, two univariate polynomials $f_\mathrm{r}(x)$, $f_\mathrm{a}(x)$ and an integer m such that $f_\mathrm{r}(m) \equiv f_\mathrm{a}(m) \equiv 0 \pmod{n}$ are selected in the polynomial selection step. Then these polynomials are converted to bivariate and homogeneous polynomials $F_\mathrm{r}(x, y)$, $F_\mathrm{a}(x, y) \in \mathbb{Z}[x, y]$. The purpose of the relation finding step is to find a large number of *relations*, namely coprime integer pairs $(a, b) \in [-H_a, H_a] \times [1, H_b] \subset \mathbb{Z} \times \mathbb{N}$ such that $F_\mathrm{r}(a, b)$ is B_r-smooth and $F_\mathrm{a}(a, b)$ is B_a-smooth. Here an integer x is called B-smooth if x is factored over the primes less than B. Procedures corresponding to F_r (F_a) are sometimes called 'rational' ('algebraic'). Parameters H_a, H_b determines the sieving region for factoring, and depend on the size of the target integer n. In this paper, we use the following values as in Table 1 [LTSK+03]. In practicve, the core sieving step picks up possible relations (called *candidates*), and the additional step checks whether it really is a relation by mini-factoring algorithms. After finding a set of relations, Gaussian elimination over a matrix generated from these relations is computed in the linear algebra step. Then, a factor of n is output in the final step.

2.2 YASD

YASD (Yet Another Sieving Device) is a dedicated sieving device of NFS proposed by Geiselmann and Steinwandt in 2004 [GT04]. YASD consists of a mesh ($2^m \times 2^m$) of processing nodes, where each node is connected to its horizontal and vertical neighbors. Each node conceptually has three parts: the Main part, the Mesh part, and the Memory part, and 2 buffers between the Main part and the Mesh part, and between the Mesh part and the Memory part. In YASD, the sieving process is achieved by sending packets from the Main part of certain nodes to the Memory part of certain nodes (target nodes). Each Mesh part of a node can hold a packet. Information of the target nodes are in the packets, and the packets are routed via the Mesh parts controled by the routing algorithm. Memory parts of a mesh can proceed $S = 2^k$ sieving lines in one process, so each node is in charge of $Z = 2^{k-2m}$ consecutive sieving positions (integer pairs).

(x_t, y_t)	c_x, c_y	$\lceil \log_{\sqrt{2}} p \rceil$	footprint	z_t	flag
(16)	(2)	(6)	(26)	(8)	(1)

Fig. 1. Packet Structure

Table 2. Dispersive storing of the factor base

size of p	storing density	size of p	storing density
$2 \sim 2^8$	1×1	$2^{14} \sim 2^{16}$	$2^4 \times 2^4$
$2^8 \sim 2^{10}$	2×2	$2^{16} \sim 2^{18}$	$2^5 \times 2^5$
$2^{10} \sim 2^{12}$	$2^2 \times 2^2$	$2^{18} \sim 2^{20}$	$2^6 \times 2^6$
$2^{12} \sim 2^{14}$	$2^3 \times 2^3$	$2^{20} \sim B_1$	$2^8 \times 2^8$

Main Part A main function of the Main part is to generate packets from the factor base (stored in the Main part) and to send packets to the Mesh Part. More precisely, for a prime p in the factor base, the Main part outputs integers r such that $f_i(r) \equiv 0 \pmod{p}$ $(i = 1, 2)$. These packets have a common property that if (p, r) is a proper packet, then $(p, r+p)$ is also a proper packet. If $(p, r+p)$ is in the current sieving subinterval S, this part sends the packet which holds $(p, r + p)$ to the Mesh Part. Otherwise, the same process is performed for the next prime p'.

Figure 1 shows a packet structure generated by the Main Part. Since nodes are arranged in a square, each node is addressed in (x, y)-coordinates. In, Figure 1, (x_t, y_t) is a coordinate values of its target node and z_t is a sieving position in which the packet will be stored. c_x, c_y are cross border flags when a torus topology is used in the mesh. A "footprint" is the factoring information which consists of the coordinates of the target node (16-bit) concatenated with its 10 lower bits of p (more precisely, the lowest bit '1' is excluded since p is prime). This information is used in postprocess called the "mini-factoring".

The coordinate values of the target node (x_t, y_t) and target sieving position z_t is computed in the Main part. If the target node is itself, namely the current node and the target node are same, the packet is directly sent to the input buffer which connect to the Main Part and the Memory Part. When a packet is sent from the input buffer to the Mesh Part, 25% nodes are kept empty in order to avoid packet congestion,

The factor base is dispersively stored in $2^m \times 2^m$ nodes. At this time the factor bases about p such that $p < Z(= 2^{k-2m})$ are stored in all nodes because such factor bases are necessarily used in all nodes. Also, the factor bases about p such that $Z < p < 2^{k-2m+2}$ are stored sharing with 2^2 nodes, and those such that $2^{k-2m+2} < p < 2^{k-2m+4}$ are stored sharing with 2^4 nodes. At last, those such that $p > 2^k$ are stored only once in the mesh (See Table 2).

Memory Part The Memory part of each node has two 10-bit DRAM entries for each of 256 z-values that the node has to be taken care of. One DRAM is for the algebraic side and the other is for the rational side. These memories are initialized with 0 and hold sums of $\lceil \log_{\sqrt{2}} p \rceil$-values. In addition, the Memory part stores footprints in the special DRAM. This DRAM is shared among two neighbor-nodes. Thus, when storing footprints, in addition to 26-bit for footprint itself, 8-bit of sieving position z_t, 1-bit algebraic/rational flag, and 1-bit to identify the node. Totally, one footprint requires 36-bit to be stored. As well as the Main part, the Memory part requires about two receiving buffer connecting the Memory part and the Mesh part. Differing from the Main part case, 41-bit is sufficient for this buffer because target nodes and cross border flags are no longer needless.

Mesh Part A main function of the Mesh part is to route packets to their target nodes. The routing is controlled by the clockwise transposition routing algorithm and a complete logic is located in the Mesh part. This part needs two 59-bit register to hold 'traveling packets' on the mesh. According to the analysis by Lenstra et al. [LSTT02], a clockwise transposition routing over an $L \times L$-mesh requires at most $2L$ steps for a non-torus mesh and at most $2(L-1)$ steps for a torus mesh.

3 Analysis of YASD

In [GT04], Geiselmann and Steinwandt analyzed the performance of YASD for 768-bit integers (YASD768). According to their estimation, an LSI of YASD 768 requires as large as 4.9 cm $\times$ 4.9 cm and the running time is about 600 days at 500 MHz frequency. However, they refer only to above values of YASD without explanation. They don't mention about the reason of these values logically. So, in this section, we will analyze makeup of the circuit based on its operation and function, and verify their claim for 768-bit.

3.1 Chip Size of YASD

Table 3 shows the number of circuit element, DRAM and transistor, needed in each part. This numbers mentioned in [GT04].

In addition to above numbers, with a 0.13μm process, we have to take $0.3[\mu\mathrm{m}^2]$ for a DRAM bit and $2.8[\mu\mathrm{m}^2]$ per transistor into account [GT04]. Using these numbers, the size of chip is formulated as below.

$$\text{Area} = (5100 \times 2.8 + 66500 \times 0.3) \times 256^2 = 22.4[\mathrm{cm}^2]$$

Therefore, we say that the claim of [GT04] is appropriate. And the difference of claimed number from calculated number by above formulation $1.6[\mathrm{cm}^2]$ can be considered as Input/Output circuit proportional to the number of node.

Table 3. The number of circuit element needed in each part

	Transistor	DRAM
Main Part	2750	55000
Memory Part	1250	11500
Mesh Part	1100	-
per node	5100	66500

3.2 Performance of the Device

In reference [GT04], it is described that the sieving process for the range $S = 2^{24}$ that YASD can deal with at once takes 40000 clocks. If we analyze this number, we need to take routing time T_{rout} for clockwise transposition routing and sending time T_{send} at the Main Part into account. Sending time T_{send} depends on the number of packet which is needed to sent to the Mesh Part.

Sending Time T_{send} To formulate this time T_{send}, we divide primes into three groups according to its size. At first, let $2^m \times 2^m$ be mesh size of YASD and 2^k be the size of sieving range which YASD can deal with at once. Next, in case of generating packets which is calculated from pairs (p, r) such that $p < Z$, in average, Z/p times calculation of p is needed. Describing in section 2.2, in case of primes such that $Z < p < 2^k$, these pair (p, r) is stored as describing Table 2 to share them among neighbor nodes. For example, primes such that $2^{k-2m+2i} < p < 2^{k-2m+2(i+1)}$ $(i = 0, \ldots, m-1)$ will take charge the sieving range of length $2^{k-2m+2(i+1)}$, thus in average, $2^{k-2m+2(i+1)}/p$ times calculation concerning this p is needed. And such pairs are shared among $2(i+1)$ nodes. Therefore, the calculating time of all primes such that $Z < p < 2^k$ is $\sum_{Z<p<2^k} \frac{Z}{p}$ per node. Concerning primes such that $2^k < p < B_{\text{a}}$ at most one calculation is needed. Consequently, concerning primes such that $p < 2^k$, the necessary number of calculation for prime sending which is dealt with one node is $2\sum_{p<2^k} \frac{Z}{p}$ (putting together algebraic and rational side). Likewise, concerning primes such that $p > 2^k$, $\dfrac{\pi(B_{\text{a}}) + \pi(B_{\text{r}}) - 2\pi(2^k)}{2^{2m}}$ times of calculation are needed.

Especially, the calculation of primes such that $p > 2^k$ requires 2 steps. So, sending time T_{send} is formulated as Eq. (1).

$$T_{\text{send}} = 2\sum_{p<2^k} \frac{Z}{p} + \frac{\pi(B_{\text{a}}) + \pi(B_{\text{r}}) - 2\pi(2^k)}{2^{2m-1}} \tag{1}$$

By substituting $Z = 256,\ m = 8,\ k = 24$ into Eq. (1), T_{send} is $\simeq 2968$ clocks. This time is rather smaller than 40000 clocks. So, T_{send} is negligible and routing time T_{rout} described later is dominant.

Routing Time $T_{\mathbf{rout}}$ At first, when we evaluate the routing time, we check up this time base on practical analysis that it takes at most $2L$ steps for routing at $L \times L$-mesh. Next, in this mesh, the number of packet which is needed to send is expressed as follows.

$$\sum_{Z<p<B_{\mathrm{a}}} \frac{2^k}{p} + \sum_{Z<p<B_{\mathrm{r}}} \frac{2^k}{p} \quad \text{, where } p \text{ is a prime} \tag{2}$$

The values $B_{\mathrm{a}},\ B_{\mathrm{r}}$ for 768-bit are respectively $B_{\mathrm{a}} = 10^9,\ B_{\mathrm{r}} = 10^8$, so routing time T_{rout} is $\simeq 4.226 \times 10^7$ from above equation. Then we use values $m = 8,\ k = 24$, which are derived from [GT04].

And we think of following assumption.

1. the mesh size is $2^m \times 2^m$.
2. Initially, the mesh is filled with $2^m \times 2^m$ packets.
3. We don't throw new packets in the mesh until the entire first packet are routed.

We apply to this model and suppose that one routing time is 2^m steps in average. In addition to above model, the YASD device applies to torus topology. According to our software simulations, if applied to torus topology, the YASD can route packets more than twice faster than so-called 'primitive' model. Furthermore, to avoid packet congestion on mesh, it is applied to more efficient releasing rule. In this way usually $\simeq 25\%$ of the node are 'free' (see Figure 2) and nearly twice faster. Therefore, we add the coefficient α that derive from above speeding up method and let α be $\simeq \frac{1}{4}$. So, the routing time T_{rout} becomes $\alpha \times 4.226 \times 10^7/2^8 = 41269$ steps. This value is nearly 40000 clocks. So, we adopt 0.242 as the coefficient α.

Consequently, the routing time T_{rout} can be written as Eq. (3).

$$T_{\mathrm{rout}} = \frac{\displaystyle\sum_{Z<p<B_{\mathrm{a}}} \frac{2^k}{p} + \sum_{Z<p<B_{\mathrm{r}}} \frac{2^k}{p}}{2^m} \times \alpha \quad \text{, where } \alpha = 0.242 \tag{3}$$

4 Formulation of the Performance

In this section, we estimate chip size and the performance time of YASD using 1024-bit parameter. The 1024-bit parameter is described at Table 1.

At first, we analyze the detail of the number of circuit element from 768-bit composition. Table 4 shows the result of this analysis for 1024-bit.

By using variables in Table 4, bit length of the packet PAC is expressed as $PAC = k + L_{\mathrm{log}} + 2m + 10 + 3$(we use this variable PAC later), where fourth term of right-hand member 10 is derived from footprint information and last term 3 is assigned to flag (see Table 1).

In addition, from reference [GT04,GT03], Table 5 shows the number of transistor needed for respective logic.

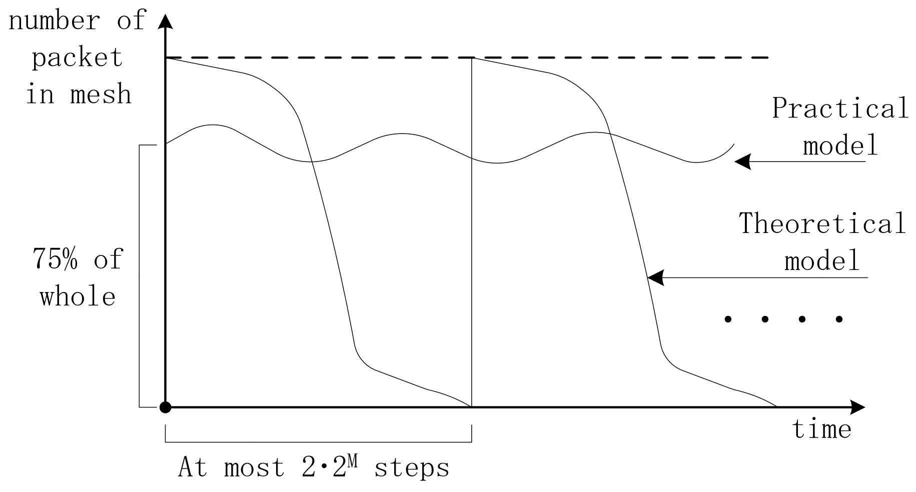

Fig. 2. Routing model

Table 4. Analysis Results

Variables		768-bit	1024-bit
L_p	bit length of maximal prime	30	32
$L_{\log}$	bit length of $\lceil \log_{\sqrt{2}} p \rceil$	6	8
L_{mem}	bit length of $\sum_p \lceil \log_{\sqrt{2}} p \rceil$	10	11
N	total number of factor bases	7.97×10^7	1.32×10^9
F	total number of footprint	0.496×2^k	0.818×2^k
L_{fb}	average of bit length of factor base	38	42

4.1 Analysis of the Number of Circuit Element

Recall that the number of circuit element is described in Table 3. In this part, we describe more detail about the number of circuit element of each part and formulate respectively.

The Number of Transistor for the Main Part For 768-bit integers, the Main Part needs 2750 transistors. The purpose of the Main Part is to generate packets from factor bases (p, r) and send to the Mesh Part. So, it needs $\max(L_p, k)$-bit adder for generating packets and buffers between Main-Mesh Part for sending. According to [GT04], two entry buffers are sufficient. And

[3] D-F/Fs make up resister.
[4] Latches make up buffer.
[5] we guessed this value from H_{add} of reference [GT03], so more precise discussion is needed.

Table 5. The number of transistor needed for respective logic

variable	logic	the number of transistor
H_{add}	1-bit adder	40
H_{dff}	1-bit D-F/F[3]	8
H_{latch}	1-bit Latch[4]	4
H_{comp}	1-bit comparator[5]	20

so forth, Main Part needs circuit to initialize information of factor bases. Then we estimate the number of this circuit to be a constant number. Totally, we can formulate the number of transistors at Main Part as Eq. (4).

$$Tr_{\text{main}} = H_{\text{add}} \times \max(L_p, k) + 2 \cdot PAC \times H_{\text{latch}} + 1000 \tag{4}$$

About the constant number of right-hand member, we also need more discussion.

The Number of DRAM for the Main Part The DRAM of the Main Part is used for holding information of factor bases. The total number of factor bases N depend on bit length of composite integer n. In YASD, each node holds this factor bases dispersively. Therefore, we estimate the number of DRAM as Eq. (5). At this equation, coefficient 1.1 means memory margin.

$$DRAM_{\text{main}} = L_{\text{fb}} \times \frac{N}{2^{2m}} \times 1.1 \tag{5}$$

The number of Transistor for the Memory Part The function of the Memory Part is to extract $\lceil \log_{\sqrt{2}} p \rceil$ and z_t from packet and add the value $\lceil \log_{\sqrt{2}} p \rceil$ to certain address represented as z_t, so it needs L_{mem}-bit adder. And it also needs buffers between Mesh-Memory Part. In this case, target address (x_t, y_t) and cross border flags (c_x, c_y) are no longer needless because the packet has already reached to target node. Therefore, the bit length $PAC - 2m - 2$ is sufficient for the buffer between Mesh-Memory Part and two entry buffers are sufficient as well as Main Part. Finally, we consider that the number of circuit to output sieving result from memories is constant. Consequently, we can formulate the number of transistor for Memory Part as Eq. (6).

$$Tr_{\text{mem}} = H_{\text{add}} \times L_{\text{mem}} + 2 \cdot (PAC - 2m - 2) \times H_{\text{latch}} + 500 \tag{6}$$

The Number of DRAM for the Memory Part The Memory Part provides two L_{mem}-bit DRAM entries for each of the $Z = 2^{k-2m}$ values which the node has to take care of. The ‘two’ means the algebraic and the rational side. And it needs the DRAM for footprint. According to [GT04], it is efficient to share the DRAM for footprint among two nodes. In this discussion, we only think the number of the DRAM per node. A footprint consist of target node address (x_t, y_t) and the bits no. 1-10 of p. Since p is prime, the least significant bit are always set. So actually footprint holds the least 10 bits of $\frac{p-1}{2}$. However, at the time of storing footprint, in addition to footprint itself, $Z = 2^{k-2m}$-bit of target sieving position, 1-bit of the algebraic/rational side flag, and 1-bit flag to identify the node are needed. Therefore, the total bit length for storing footprint is 38-bit. Although the average number of footprint stored in a node is about 127 by software calculation, the reference [GT04] says that it needs 325 entries for footprint among two nodes. So we take this into account and formulate as Eq. (7). The Coefficient 1.3 derives from $325/(2 \cdot 127)$.

$$DRAM_{\text{mem}} = 2 \cdot Z \times L_{\text{mem}} + \frac{F}{2^{2m}} \times (k + 10 + 2) \times 1.3 \tag{7}$$

The Number of Transistor for the Mesh Part The function of the Mesh Part is compare/exchange operation. Exchanging logic consists of two PAC-bit resisters. Although it needs the circuit which take care of cross border flag (c_x, c_y), we consider that its circuit is as small as negligible.

$$Tr_{\text{mesh}} = 2 \cdot PAC \times H_{\text{dff}} + m \times H_{\text{comp}} \tag{8}$$

Changes of $\boldsymbol{L_p, L_{\text{mem}}, L_{\text{fb}}}$ These variables are changed when we change factoring number n from 768-bit to 1024-bit. We discuss about these variables in this part.

[L_p] For 1024-bit, the smoothness bound of algebraic side B_1 is 2.6×10^{10}. So, we can consider maximal prime as this value, so that L_p is $\lceil \log_2 \log_{\sqrt{2}}(2.6 \times 10^{10}) \rceil = 7$. Although 7-bit is sufficient for L_p, we have to take that L_p holds sum of ceiling value into account. So we can say that 8-bit is adequate for L_p.

[L_{mem}] We discuss the maximal value of sum of $\lceil \log_2 \log_{\sqrt{2}} p \rceil$. This value is originally derives from $\simeq F(a, b)$. $F(a, b)$ is generated from $f(X)$ referred to [LTSK+03]. The order of the maximal value of $F(a, b)$ is about 10^{105}. This value is given by substituting $a = A = 5.5 \times 10^{14}, b = B = 2.7 \times 10^8$. Therefore, L_{mem} becomes $\lceil \log_2 \log_{\sqrt{2}}(10^{105}) \rceil = 10$. Since L_{mem} also holds the sum of ceiling value, 11-bit is adequate for L_{mem}.

[L_{fb}] The factor base information consist of the difference of consecutive primes $\frac{\Delta p}{2}$ and the root r. The majority of factor bases is so-called Hugish prime, which is $p > B_{\text{r}}$ and so only has algebraic root[6]. So, about $\frac{\Delta p}{2}$, it is sufficient to make 2-bit longer as well as L_{log}. And about algebraic root r, it is sufficient to make 2-bit longer as well as L_p. Summing up these, the bit length of factor base L_{fb} can be considered as 42-bit.

[6] The data structure of Hugish prime is as $\boxed{\frac{\Delta p}{2}}\boxed{\text{algebraic root}}\boxed{\text{flag}}$.

4.2 Area Formula

From above discussions, we can formulate the area of general YASD (for N-bit integers) as follows.

$$\begin{aligned}\text{Area}^{(N)}(k,m) = \{&2.8 \times (68m + 72k + 32L_{\log} + 40L_{mem} + 1916) \\ &+ 0.3 \times (1.1L_{fb}\frac{n}{2^{2m}} + 2^{k-2m+1}L_{mem} + 1.3(k+12)\frac{F}{2^{2m}})\} \times 2^{2m} \\ &+ 2000 \times 2^{2m} \qquad (\mu\text{m}^2) \end{aligned} \tag{9}$$

Here, 2000×2^{2m} corresponds to Input/Output circuit proportional to the number of node (see section 3.1).

5 Analysis of YASD for 1024-bit Integers

In this section, we formalize the circuit area and processing time of YASD for 1024-bit integers (YASD1024). Then we optimize the parameters k, m in the sense of Area-Time product.

5.1 Circuit Area

By substituing parameters in Table 4 into the equation (9), we obtain the following formula of circuit area for YASD1024:

$$\begin{aligned}\text{Area}^{(1024)}(k,m) = &(162.4m + 89.6k + 12959.2) \times 2^{2m} \\ &+ (0.321k + 9.246) \times 2^k + 1.83 \times 10^{10} \qquad (\mu\text{m}^2)\end{aligned} \tag{10}$$

5.2 Processing Time

According to the equation (3), the total number of packet being sent is as follows.

$$\sum_{p>Z}^{2.6\times10^{10}} \frac{2^k}{p} + \sum_{p>Z}^{3.5\times10^{9}} \frac{2^k}{p} = (6.27 - 2\log\log Z) \times 2^k \tag{11}$$

Therefore, the routing time $T_{\text{rout}}(k,m)$ is given as Eq.12.

$$T_{\text{rout}}(k,m) = (6.24 - 2\log\log 2^{k-2m}) \times 2^{k-m} \times 0.24 \tag{12}$$

In addition, since the sending time T_{send} is obtained from Eq. (1), the number of steps needed for one sieving process can be obtained.

In NFS, relations are collected from a region $-H_a \le a \le H_a,\ 1 \le b \le H_b$. However, there is condition that the pair (a,b) must be coprime. So in this region, if a and b are both even number, it is no necessity to sieve for such a and b. By this means, 25% of the whole sieving region can be excluded.

And in order to compare to the TWIRL, we assume the condition as below.

Table 6. Area and time values

		$m = 5$	6	7	8	9	10	11	12	13
	Area (cm^2)	184	184	186	194	227	357			
$k = 22$	Time (year)	365	216	128	77	48	32	—	—	—
	AT product	67044	39729	23863	14988	10832	11355			
	Area (cm^2)	186	187	189	197	229	361	895		
24	Time (year)	309	182	108	64	39	24	16	—	—
	AT product	57392	34006	20314	12590	8842	8634	14215		
	Area (cm^2)	195	195	197	206	239	372	911	3087	
26	Time (year)	260	154	91	54	32	19	12	8	—
	AT product	50679	30156	18006	11082	7643	7176	10884	24526	
	Area (cm^2)	232	233	235	243	276	411	956	3154	12029
28	Time (year)	217	130	77	46	27	16	10	6	4
	AT product	50380	30233	18102	11074	7445	6583	9214	18842	47789
	Area (cm^2)	386	386	388	397	431	567	1117	3338	12303
30	Time (year)	179	109	65	39	23	13	8	5	3
	AT product	68947	41930	25241	15304	9813	7635	8940	16090	36749

1. use 582 pieces of 300mm silicon wafer.
2. operating frequency is 500[MHz].

Totally, the total processing time $\mathrm{Time}^{(1024)}(k, m)$ is formulated as follows:

$$\begin{aligned}\mathrm{Time}^{(1024)}(k, m) &= \frac{T_{\mathrm{rout}}}{500[\mathrm{MHz}]} \cdot \frac{2H_a \times H_b}{2^k} \cdot \frac{3}{4} \cdot \frac{1}{582} \cdot \frac{1}{365 \cdot 24 \cdot 3600} \\ &= \frac{2912.72(6.24 - 2 \log \log 2^{k-2m})}{2^m} \quad \text{(years)} \qquad (13)\end{aligned}$$

5.3 Optimizations of Parameters

In this section, we optimize parameters k, m in the sense of Area-Time product. Table 6 shows concrete values of circuit area and processing time of YASD1024 . From these data, we obtain optimized parameters as $k = 26$ and $m = 10$. In this case, an LSI requires as large as about 411.17 cm^2 and 16 years for the sieving.

6 Concluding Remarks

This paper estimates the circuit area and the processing time of YASD for 1024-bit integers. In the optimized parameters, namely $k = 28$ and $m = 10$, YASD requires at least 411 cm^2 and 16 years for the sieving. Since we did not consider wiring problem in detail in our estimation, YASD1024 will reuire much more area and time even if it is manufactured and proceeded.

Acknowledgments

The authors thank Takeshi Shimoyama for his helpful suggestions. We also thank anonymous reviewers for their comments.

References

[Ber01] Daniel J. Bernstein, "Circuits for Integer Factorization: A Proposal", preprint, 2001. `http://cr.yp.to/papers/nfscircuit.pdf`

[RSA200] Friedrich Bahr, M. Böhm, Jens Franke, and Thorsten Kleinjung, "RSA200", May 2005. `http://www.crypto-world.com/announcements/rsa200.txt`

[FKPP+05] Jens Franke, Thorsten Kleinjung, Christof Paar, Jan Pelzl, Christine Priplata, and Colin Stahlke, "SHARK: A Realizable Special Hardware Sieving Device for Factoring 1024-bit Integers", *CHES 2005*. pp.119-130, LNCS 3659, 2005.

[GT04] Willi Geiselmann, and Rainer Steinwandt, "Yet Another Sieving Device", *CT-RSA 2004*, pp.278-291, LNCS 2964, 2004.

[GT03] Willi Geiselmann, and Rainer Steinwandt, "A Dedicated Sieving Hardware", *PKC 2003*, pp. 254-266, LNCS 2567, 2002.

[LSTT02] Arjen K. Lenstra, Adi Shamir, Jim Tomlinson, and Eran Tromer, "Analysis of Bernstein's Factorization Circuit", *ASIACRYPT 2002*, pp. 1-26, LNCS 2501, 2002.

[LTSK+03] Arjen K. Lenstra, Eran Tromer, Adi Shamir, Wil Kortsmit, Bruce Dodson, James Hughes, and Paul C. Leyland, "Factoring Estimates for a 1024-Bit RSA Modulus", *ASIACRYPT 2003*, pp.55-74, LNCS 2894, 2003.

[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 Number with the TWIRL Device", *CRYPTO 2003*, pp. 1-26, LNCS 2729, 2003.