# Multiplier System in the Tile Assembly Model with Reduced Tileset-Size

Xiwen Fang and Xuejia Lai*

Department of Computer Science and Engineering, Shanghai Jiao Tong University,
Shanghai 200240, China
lai-xj@cs.sjtu.edu.cn

**Abstract.** Previously a 28-tile multiplier system which computes the product of two numbers was proposed by Brun. However the tileset-size is not optimal. In this paper we prove that multiplication can be carried out using less tile types while maintaining the same time efficiency: we propose two new tile assembly systems, both can deterministically compute $A * B$ for given $A$ and $B$ in constant time. Our first system requires 24 computational tile types while our second system requires 16 tile types, which achieve smaller constants than Brun's 28-tile multiplier system.

**Keywords:** tile assembly model, DNA computing, multiplier, tileset-size.

## 1 Introduction

### 1.1 Background and Related Work

Since Adleman's pioneering research which shows DNA could be used to solve Hamiltonian path problem [1], many researchers have explored the ability of biological molecules to perform computation [2,3,4]. The theory of tile self-assembly model which was developed by Winfree and Rothemund [5,6,7] provides a useful framework to study the self-assembly of DNA. This model has received much attention over the past few years. Researchers have demonstrated DNA implementations of several tile systems: Barish et al. [8] have demonstrated DNA implementations of copying and counting; Rothemund et al. [9] have demonstrated DNA implementation of *xor* tile system. Several systems solving satisfiability problem are also proposed [10,11,12].

The efficiency of a tile asssembly system involves two factors: the tileset-size and the assembly time. In [13], Brun proposed a multiplier system that computes the product of two numbers, which requires 28 distinct computational tile types besides the tiles constructing the seed configuration. The computation can be carried out in time linear in the input size. However, the tileset-size of Brun's system is not optimal, i.e. multiplier system can be implemented using less tile types.

* Corresponding author.

In this paper we present two new multiplier systems which achieve smaller constants for the multiplication problem than the previous 28-tile multiplier system proposed by Brun, while maintaining the same time efficiency. In our first system, we show that multiplication can be carried out using 24 tile types instead of 28 tiles types, then we propose a second multiplier system and show that the tileset-size can be further reduced to 16.

The remaining of this paper is organized as follow: in section 1.2 we briefly introduce the concept of tile assembly model to assist the reader. In section 1.3 we introduce the corresponding algorithms. Several subsystems are discussed in section 2. In section 3 we present two new multiplier systems and compare our system with existing system[13] in terms of tileset-size and assembly time. Our contributions are summarized in section 4.

### 1.2 Tile-Assembly Model

To assist the reader, in this section we briefly introduce the concept of tile assembly model. We refer to $\Sigma$ as a finite alphabet of symbols called binding domains. We assume $null \in \Sigma$. Each tile has a binding domain on its north, east, south and west side. We represent the set of directions as $D = \{N, E, S, W\}$.

A tile over a set of binding domains is a 4-tuple. For a tile $t$, for $\langle\delta_N, \delta_E, \delta_S, \delta_W\rangle \in \Sigma^4$, we will refer to $bd_d(t)$ as the binding domain of tile $t$ on $d$ 's side.

A **strength function** $g : \Sigma \times \Sigma \rightarrow N$ denotes the strength of the binding domains. $g$ is commutative and $\forall \delta \in \Sigma, g(\delta, null) = 0$. Let $T$ be a set of tiles containing *empty*. A tile system $S$ is a triple $< T, g, \epsilon >$. $\epsilon \in \mathbb{N}$ is the temperature. A tile can attach to a configuration only in empty positions if and only if the total strength of the appropriate binding domains on the tiles in neighboring positions meets or exceeds the temperature $\epsilon$.

Given a set of tiles $\Gamma$, a **seed configuration** $S' : \mathbb{Z}^2 \rightarrow \Gamma$ and $S =< T, g, \epsilon >$, configurations could be produced by $S$ on $S'$. If no more attachment is possible, then we have the **final configuration**.

The reader may refer to [5,6,7] for more discussion of the concept of tile assembly model.

Let $||$ be a special binding domain which connects the tiles constructing seed configuration. To simplify the discussion, for all of the systems involved in this paper, we define the strength funcion $g$ as follow:

$$\forall \delta \in \Sigma, g(\delta, null) = 0; g(||, ||) = 2; \forall \delta \in \Sigma, \delta \neq ||, g(\delta, \delta) = 1. \qquad (1)$$

### 1.3 Preliminary Algorithms

Intuitively, a multiplier system could be implemented by combining subsystems which compute $f(x) = 2x$ and $f(x, y) = 2x + y$ respectively.

Given $n_A$-bit binary integer $A$ and $n_B$-bit binary integer $B$. We denote by $A_i$ and $B_i$ the $i_{th}$ digit of $A$ and $B$. $A = \sum_{i=0}^{n_A-1} 2^i A_i$, $B = \sum_{i=0}^{n_B-1} 2^i B_i$. We use Algorithm 1 to compute $A * B$.

```
    Input: $n_A$-bit binary integer $A$ , $n_B$-bit binary integer $B$ ($n_B \geq 2$)
    Output: $S = A * B$
 1  $i \leftarrow n_B - 2$.
 2  $S \leftarrow A$
 3  while $i \geq 0$ do
 4      if ($B_i = 0$) then
 5          $S \leftarrow 2S$
 6      end
 7      else
 8          $S \leftarrow 2S + A$
 9      end
10      $i \leftarrow i - 1$
11  end
```

**Algorithm 1.** Given binary integers $A$ and $B$, compute $A * B$

## 2 Subsystems of Multiplier System

Our multiplier system is a combination of several subsystems. In this section we will define these subsystems respectively.

To simplify the definition of the seed configuration, for $n_A$-bit integer $A$, we denote by $A_i (0 \leq i \leq n_A - 1)$ the $i_{th}$ digit of $A$. We also define $A_i (n_A \leq i \leq n - 1, n > n_A)$ as follow: we pad the $n_A$-bit binary integer $A$ with $n$-$n_A$ 0-bits and denote these 0-bits by $A_i (n_A \leq i \leq n - 1)$. Thus $A = \sum_{i=0}^{n-1} 2^i A_i$. The same definition holds for any other input variables in this paper.

### 2.1 Shifter Tile System

In this section we propose an 8-computational-tile system for computing $f(A, B) = (2A, B)$.

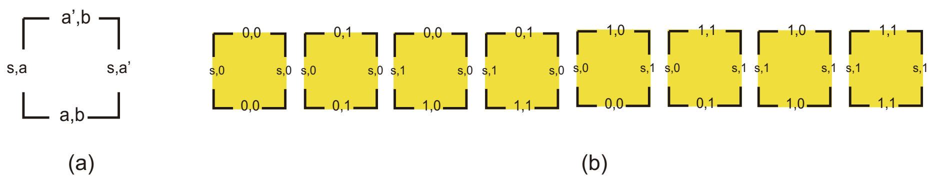

**Fig. 1.** Shifter tile system

Let $\Sigma_S$ ={(0, 0), (0, 1), (1, 0), (1, 1), (s, 0), (s, 1)}, and $T_s$ be a set of tiles over $\Sigma_S$ defined as follow:

$$T_s = \{\langle (a', b), (s, a'), (a, b), (s, a) \rangle | a, b, a' \in \{0, 1\}\}. \quad (2)$$

Fig.1(a) shows the concept behind the system which includes variable $a$, $b$, $a'$. All of the tiles has two input sides (south and east) and two output sides (north and west), for input $a'$ on the east side, we output the same value on the north side; for input $a$ on the south side, we output the same value on the west side. We have $a, b, a' \in \{0, 1\}$, thus there are 8 tiles in the system (Fig.1(b)).

Let $\Gamma = \{\alpha_0\beta_0 = \langle(0,0), ||, null, ||\rangle$, $\alpha_0\beta_1 = \langle(0,1),\ ||,\ null,\ ||\rangle$, $\alpha_1\beta_0 = \langle(1,0),\ ||,\ null,\ ||\rangle$, $\alpha_1\beta_1 = \langle(1,1),\ ||,\ null,\ ||\rangle$, $\alpha\beta_{bound} = \langle(0,0),\ ||,\ null,\ null\rangle$, $\gamma_0 = \langle||,\ null,\ null,\ ||\rangle\}$ $\gamma_1 = \langle null,\ null,\ ||,\ (s,0)\rangle\}$. Let $A_i, B_i \in \{0,1\}$, $n \geq max(n_A, n_B) + 1$. Let the seed configuration $S : \mathbb{Z}^2 \to \Gamma$ be such that

$$\begin{cases} S(1,1) = \gamma_1; S(1,0) = \gamma_0; \\ \forall i = 0,1,\ldots,n-2 : S(-i,0) = \alpha_{A_i}\beta_{B_i}; S(-n+1,0) = \alpha\beta_{bound}; \\ \text{For all other } (x,y) \in \mathbb{Z}^2, S(x,y) = empty. \end{cases} \tag{3}$$

**Theorem 1.** *Let $\epsilon_S = 2$. Given a seed configuration $S$ encoding $A$ and $B$ and the strength function $g$ as defined in* (3) *and* (1)*, the system $S_S =< T_S, g, \epsilon_S >$ computes the function $f(A,B) = (2A, B)$.*

## 2.2 Adder Tile System

In this section we propose a 8-tile system computing $f(A,B) = (A+B, B)$.

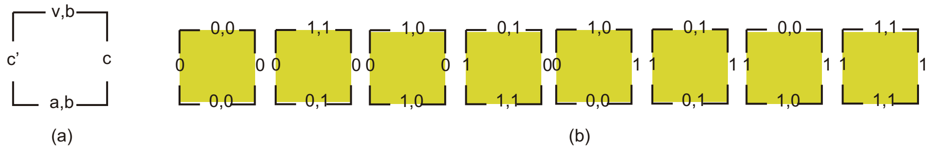

**Fig. 2.** Adder tile system (a)The tile has two input sides(south and east) and two output sides (north and west). The south side contains the value of the current bit of $A$ and $B$; the east side is the carry bit. (b) There are 8 tiles in the system.

Fig. 2(a) shows the concept of this system which include variable $a$, $b$, $c$. For given $i$, on the input sides, the variable $a$ and $b$ represent $A_i$ and $B_i$; the variable $c$ represents the corresponding carry bit. We have $V = A+B$. On the output sides, $v$ represents $V_i$ and $c'$ represents the next carry bit. Let $a, b, c, c', v \in \{0,1\}$. There are three variables $a, b, c$ on the input sides, thus there are 8 tiles in the system. According to addition rule, we define two functions $f_v(a,b,c)$ and $f_{c'}(a,b,c)$ which computes $v$ and $c'$ respectively:

$$\begin{cases} if(a+b+c=0), f_v(a,b,c) = 0, f_{c'}(a,b,c) = 0; \\ if(a+b+c=1), f_v(a,b,c) = 1, f_{c'}(a,b,c) = 0; \\ if(a+b+c=2), f_v(a,b,c) = 0, f_{c'}(a,b,c) = 1; \\ if(a+b+c=3), f_v(a,b,c) = 1, f_{c'}(a,b,c) = 1. \end{cases} \tag{4}$$

Let $\Sigma_+ = \{(0,0),(0,1),(1,0),(1,1),0,1\}$. We have $v = f_v(a,b,c), c' = f_{c'}(a,b,c)$. Let $T_+$ be a set of tiles over $\Sigma_+$ defined as follow:

$$T_+ = \{\langle (v,b), c, (a,b), c' \rangle\} \tag{5}$$

Fig.2(b) shows all of the eight tiles.

Let the seed configuration $S$ be defined as it is in (3) except that $\gamma_1 = \langle null, null, ||, 0\rangle$.

**Theorem 2.** *Let $\epsilon_+ = 2$, $g$ defined in* (1)*, $\Sigma_+$ defined as above, and $T_+$ be a set of tiles over $\Sigma_+$ as defined in (5). Given a seed configuration $S$ encoding $A$ and $B$ which is defined above, the system $S_+ =< T_+, g, \epsilon_+ >$ computes the function $f(A,B) = (A+B, B)$*

## 2.3 Shifter-Adder Tile System

In this section we propose a 16-computational-tile system computing $f(A,B) = (2A+B, B)$. Fig. 3(a) shows the concept behind the tile system. For given $i$, on the input sides, the variable $a$ and $b$ represent $A_i$ and $B_i$; $a'$ represents $A_{i-1}$; the variable $c$ represents the corresponding carry bit. Let $V = 2A + B$, thus for each bit we compute $A_{i-1} + B_i$. On the output sides, $v$ represents $V_i$ and $c'$ represents the next carry bit. Let $a, b, c, a', c', v \in \{0,1\}$, we denote the function which computes $v$ and $c'$ by $f_v(a', b, c)$ and $f_{c'}(a', b, c)$ respectively. According to the addition rule, the functions $f_v$ and $f_{c'}$ are defined as follow:

$$\begin{cases} if(a'+b+c=0), f_v(a',b,c)=0, f_{c'}(a',b,c)=0; \\ if(a'+b+c=1), f_v(a',b,c)=1, f_{c'}(a',b,c)=0; \\ if(a'+b+c=2), f_v(a',b,c)=0, f_{c'}(a',b,c)=1; \\ if(a'+b+c=3), f_v(a',b,c)=1, f_{c'}(a',b,c)=1. \end{cases} \tag{6}$$

Let $\Sigma_{S+} = \{(0,0),(0,1),(1,0),(1,1)\}$. We have $v = f_v(a',b,c)$ and $c' = f_{c'}(a',b,c)$. Let $T_{S+}$ be a set of tiles over $\Sigma_{S+}$ defined as follow:

$$T_+ = \{\langle (v,b), (c,a'), (a,b), (c',a) \rangle\} \tag{7}$$

The input sides involve 4 variables, i.e. $a$, $b$, $a'$,$c$, thus there are 16 tiles in this system(Fig. 3(b)). Let the seed configuration $S$ be defined as it is in (3) except that $\gamma_1 = \langle null, null, ||, (0,0)\rangle$.

**Theorem 3.** *Let $\epsilon_{S+} = 2$, $g$ defined in* (1)*, Given a seed configuration $S$ encoding $A$ and $B$ as above, the system $S_{S+} =< T_{S+}, g, \epsilon_{S+} >$ computes the function $f(A,B) = (2A+B, B)$*

Fig. 4(a) shows a sample seed configuration which encodes $A = 0101011_2$, $B = 0100011_2$. Fig. 4(b) shows a sample execution of $S_{S+}$ on the seed configuration. We could read the result from the top row of the final configuration, i.e. $2A+B = 1111001_2$.

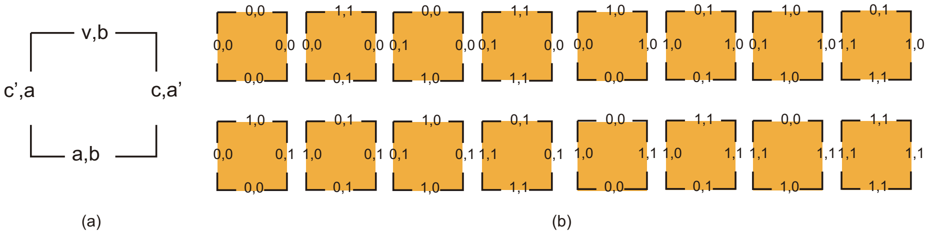

**Fig. 3.** Shifter-Adder tile system (a) The tile has two input sides(south and east) and two output sides (north and west). The south side contains the value of the current bit of $A$ and $B$; the east side contains the value of the former bit of $A$ and the carry bit. (b) There are 16 tiles in the system.

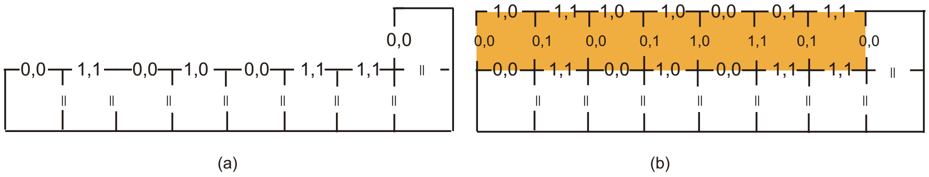

**Fig. 4.** (a) seed configuration: $A = 0101011_2$, $B = 0100011_2$.(b) The top row reads the solution: $2A + B = 1111001_2$.

## 3 Multiplier Tile System

### 3.1 Multiplier Tile System: Version 1

We use subsystems $S_S$ and $S_S+$ to build the multiplier system. All of the tiles have two input sides(south and east) and two output sides (north and west).

The method of establishing the seed-configuration is described as follow: We use the two sides of the L-configuration to encode inputs. One of the input number, i.e. $A$, is encoded on the bottom row. On the same row there are also $n_B$ extra tiles representing 0 in the most significant bit places because the product of the $n_A$-bit number $A$ and the $n_B$-bit number $B$ may be as large as $n_A + n_B$ bits.

Let $n = n_A + n_B$. Let $\Sigma_\times = \{(0,0), (0,1), (1,0), (1,1), (s,0), (s,1)\}$.

Let $\Gamma = \{\ \alpha_0 = \langle(0,0), ||, null, ||\rangle,\ \alpha_1 = \langle(1,1), ||, null, ||\rangle,\ \alpha_{bound} = \langle(0,0), ||,\ null, null\rangle,\ \beta_0 = \langle||, null, ||, (s,0)\rangle,\ \beta_1 = \langle||, null, ||, (0,0)\rangle,\ \beta_{bound0} = \langle null, null, ||,\ (s,0)\rangle,\ \beta_{bound1} = \langle null, null, ||, (0,0)\rangle,\ \gamma_0 = \langle||, null, null, ||\rangle\}$. Let the seed configuration $S : \mathbb{Z}^2 \to \Gamma$ be such that

$$\begin{cases} S(1,0) = \gamma_0 \\ \forall i = 0, 1, \ldots, n-2 : S(-i, 0) = \alpha_{A_i}; S(-n+1, 0) = \alpha_{bound} \\ \forall i, 1 \le i \le n_B - 2, S(1, n_B - i - 1) = \beta_{B_i}; S(1, n_B - 1) = \beta_{boundB_0} \\ \text{For all other } (x,y) \in \mathbb{Z}^2, S(x,y) = empty. \end{cases} \tag{8}$$

The seed configuration is of length $n_A + n_B + 1$ and height $n_B$.

**Theorem 4.** *Let $\epsilon_\times = 2$, $g$ defined in* (1)*, and $T_\times = T_S \cup T_{S+}$. Given a seed configuration encoding $A$ and $B$ which is defined in* (8)*, the system $S_\times =< T_\times, g_\times, \epsilon_\times >$ outputs $(A * B, A)$.*

*Proof.* We have $T_\times = T_S \cup T_{S+}$. Let $bd_{E,W}(T_S) = \{bd_E(t), bd_W(t) | t \in T_S\}$, $bd_{E,W}(T_{S+}) =\{bd_E(t),bd_W(t)|\ t \in T_{S+}\}$ . These two sets are disjoint, thus $S_S$ and $S_{S+}$ work together without interfering.

For each tile $t(t \in T_\times)$, $bd_S(t)$ has two bits. Let $bd_Sl(t)$ be the first bit and $bd_Sr(t)$ be the second bit. Let $bd_S(F(i)) = (bd_SlF(i), bd_SrF(i))$. Let $bd_Sl(F(i)) = \sum_{j=0}^{n-1} 2^j * bd_Sl(F(-j, i))$, $bd_Sr(F(i)) = \sum_{j=0}^{n-1} 2^j * bd_Sr(F(-j, i)))$. The definition also holds for $bd_Nl(t)$, $bd_Nr(t)$, $bd_N(F(i))$ , $bd_Nl(F(i))$ and $bd_Nr(F(i))$.

Let $b_k = \sum_{i=n_B-k}^{n_B-1} B_i * 2^{i-n_B+k}$. Thus $B = b_{n_B}$, $B_{n_B-1} = b_1 = 1$, $2 * B_{n_B-1} + B_{n_B-2} = b_2$. We are going to prove that given the seed configuration $S$ encoding $A$ and $B$ which is defined in(8), for $1 \le i \le n_B - 1$, $bd_N(F(i)) = (A * b_{i+1}, A)$:

i) For $i = 1$, we have $bd_S(F(1)) = bd_N(F(0)) = (A, A)$.

If $B_{n_B-2} = 0$, i.e. $B_{n_B-i-1} = 0$,$b_2 = 10_2$, then $bd_W(F(1, 1)) = (s, 0)$;

If $B_{n_B-2} = 1$, i.e. $B_{n_B-i-1} = 1$, $b_2 = 11_2$, then $bd_W(F(1, 1)) = (0, 0)$ .

Thus according to Theorem 2 and Theorem 4, if $B_{n_B-2} = 0$ , we have $bd_N(F(1)) = (2A, A) = (A * b_2, A)$ ; if $B_{n_B-2} = 1$, we have $bd_N(F(1)) = (2A + A, A) = (A * b_2, A)$.

Thus $bd_N(F(i)) = (A * b_{i+1}, A)$ holds for $i = 1$.

ii) Assume $bd_N(F(i)) = (A * b_{i+1}, A)$ holds for $i = k - 1$, i.e. $bd_N(F(k - 1)) = (A * b_k, A)$ .

For $i = k$, if $b_{(k+1)_0} = B_{n_B-k-1} = 0$, then $bd_W(F(1, i)) = (s, 0)$; If $b_{(k+1)_0} = B_{n_B-k-1} = 1$, then $bd_W(F(1, i)) = (0, 0)$ .

Thus according to Theorem 2 and Theorem 4, if $b_{(k+1)_0} = B_{n_B-k-1} = 0$ , we have $bd_N(F(k)) = (2bd_Nl(F(k - 1)), A) = (2 * A * b_k, A) = (A * b_{k+1}, A)$ ; if $b_{(k+1)_0} = B_{n_B-k-1} = 1$, we have $bd_N(F(k)) = (2bd_Nl(F(k - 1)) + A, A) = (2 * (A * b_k) + A, A) = (A * b_{k+1}, A)$.

iii) Therefore $bd_N(F(i)) = (A * b_{i+1}, A)$ holds for $1 \le i \le k$. Let $k = n_B - 1$, we have $bd_N(F(n_B - 1)) = (A * b_{n_B}, A) = (A * B, A)$.

Thus we proved that given the seed configuration encoding $A$ and $B$, the system outputs $(A * B, A)$. □

Fig. 5(a) shows a sample seed configuration which encodes $A = 1001_2$, $B = 1011_2$: $A$ is encoded on the bottom row with 4 extra 0 tiles in the most significant bit place. We encode $B$ from $B_{n_B-2}$ to $B_0$. $B_3$ is the most significant bit, and in the seed configuration we discard this bit. $B_2 = 0$, $S(1, 1) = \beta_0$; $B_1 = 1$, $S(1, 2) = \beta_1$; $B_0 = 1$, $S(1, 3) = \beta_{bound1}$. Fig. 5(b) shows a sample execution of the multiplier system on the seed configuration and the product could be read from the top row of the final configuration, i.e. $1001_2 * 1011_2 = 1100011_2$.

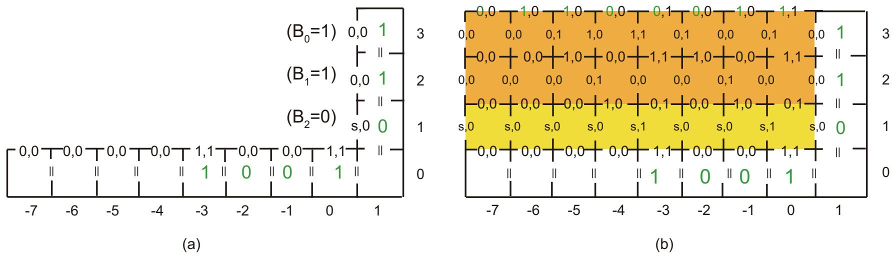

**Fig. 5.** Given a sample input of $A = 1001_2$, $B = 1011_2$, with $A$ on bottom row and $B$ on the right most column. (b) $1001_2 * 1011_2 = 1100011_2$.

## 3.2 Multiplier Tile System: Version 2

In this section we combine the shifter system $S_S$ and the adder system $S_+$ to create another simplifier multiplier system which computes $f(A,B) = (A*B, A)$ and uses only 16 tiles.

We discuss how to establish the seed configuration for this system:

Let $\Gamma = \{\ \alpha_0 = \langle(0,0), ||, null, ||\rangle,\ \alpha_1 = \langle(1,1), ||, null, ||\rangle,\ \alpha_{bound} = \langle(0,0), ||, null, null\rangle,\ \beta_0 = \langle ||, null, ||, (s,0)\rangle,\ \beta_1 = \langle ||, null, ||, 0\rangle,\ \beta_{bound0} = \langle null, null, ||, (s,0)\rangle,\ \beta_{bound1} = \langle null, null, ||, 0\rangle,\ \gamma_0 = \langle ||, null, null, ||\rangle\}$. Let $n = n_A + n_B$. Let the seed configuration $S : \mathbb{Z}^2 \to \Gamma$ be such that

$$\begin{cases} S(1,0) = \gamma_0 \\ \forall i = 0,1,\ldots,n-2, S(-i,0) = \alpha_{A_i}; S(-n+1,0) = \alpha_{bound} \\ j \leftarrow 1; k \leftarrow n_B - 2; \\ while(k \geq 1)\{ \\ if(B_k = 0)\{S(1,j) = \beta_0; j \leftarrow j+1;\} \\ else\{S(1,j) = \beta_0; S(1,j+1) = \beta_1; j \leftarrow j+2;\} \\ k \leftarrow k-1;\} \\ if(B_0 = 0), S(1,j) = \beta_{bound0} \\ else\{S(1,j) = \beta_{bound0}; S(1,j+1) = \beta_{bound1};\} \\ \text{For all other } (x,y) \in \mathbb{Z}^2, S(x,y) = empty. \end{cases} \tag{9}$$

**Theorem 5.** *Let $\epsilon_{\times 2} = 2$, $g$ defined in (1), and $T_{\times 2} = T_S \cup T_+$. Given a seed configuration encoding $A$ and $B$ as defined in (9), the system $S_{\times 2} =< T_{\times 2}, g_{\times 2}, \epsilon_{\times 2} >$ outputs $(A * B, A)$.*

*Proof.* According to the establishment of the seed configuration, in this system we carry out the shift-addition in two steps: we shift the first input number by one bit, and then we add the second input number. Comparing with the first multiplier system presented in section 3.1, although the shift-addition is carried out in two steps, which takes two rows in the configuration, the algorithms of these two systems are just identical. Thus the correctness of this system follows directly from Theorem 4. □

## 3.3 Discussion

**Lemma 6.** *(Multiplier Assembly Time Lemma). For all $A \geq 1, B \geq 2$, the assembly time of the final configuration $F$ produced by $S_\times$ or $S_{\times 2}$ on $S$ that encodes $A$ and $B$ and pads $A$ with $n_B$ 0-tiles is $\Theta(n_A + n_B)$.*

*Proof.* For both of the two systems, the length of the final configuration is $n_A + n_B$, the height is $\Theta(n_B)$. According to Lemma 2.3 in[13], the assembly time of the final configuration $F$ produced by $S_\times$ or $S_{\times 2}$ on $S$ that encodes $A$ and $B$ and pads $A$ with $n_B$ 0-tiles is $\Theta(n_A + n_B)$. □

We use $l$ and $h$ to represent the length and the height of the final configuration.

For the first system, we use 24 distinct tile types. The final configuration is of height $n_B$. We reduce the tileset-size of the system to 16 in our second system. However the height of the final configuration is increased. The value of $h$ depends on the number of 1-bits in $B$. Suppose there are $x$ 1-bits in $B$, i.e. we have $n_B - x$ 0-bits in $B$. Each 1-bit(except the MSB 1-bit) takes two rows in the configuration while each 0-bit takes one row, thus $h = 1 + (n_B - x) + 2(x - 1) = n_B + x - 1$. We have $1 \leq x \leq n_B$,thus $n_B \leq h \leq 2n_B - 1$.

Along with Brun's scheme[13], we make comparison of these three systems in Table 1. We could see that the tileset-size of our second system is the least; however the height of the final configuration might be increased, depending on the number of 1-bits in $B$. The height of the final configuration of our first system is the least; the tileset-size is also less than Bruns. All of these three systems share the same assembly time, i.e. $\Theta(n_A + n_B)$.

While consider performing multiplication, one chould make a choice between these systems according to the actual demand. Intuitively, our first system could be choosed if one hopes to construct the seed configuration in a simpler way, while the second system is beneficial for those who want to perform multiplication with tile types as less as possible.

**Table 1.** Comparing the efficiency of multiplier systems

| Scheme | version 1 | version 2 | Brun |
|---|---|---|---|
| Tileset-size | 24 | 16 | 28 |
| Assembly time | $\Theta(n_A + n_B)$ | $\Theta(n_A + n_B)$ | $\Theta(n_A + n_B)$ |
| $l$ | $n_A + n_B + 1$ | $n_A + n_B + 1$ | $n_A + n_B + 1$ |
| $h$ | $n_B$ | $n_B \leq h \leq 2n_B - 1$ | $n_B + 1$ |

## 4 Conclusion

In this paper we present two multiplication systems which compute $A * B$ for given $A$ and $B$ in constant time. Our first system is constructed using subsystems $S_S$ and $S_{S+}$, which requiring 24 computational tile types; Further improvement is achieved in our second system: using subsystems $S_S$, $S_+$ and a different seed configuration, we show that multiplier system could be implemented using only 16 tile types. As a result, we achieved reduced tileset-size compared with that of Brun's 28-tile multiplier system.

**Acknowledgements.** This work was supported by the National Natural Science Foundation of China (No. 61272440, No. 61073149), Research Fund for the Doctoral Program of Higher Education of China (20090073110027), State Key Laboratory of ASIC & System (11KF0020), Key Lab of Information Network Security, Ministry of Public Security (C11603) and Scholarship Award for Excellent Doctoral Student granted by Ministry of Education.

## References

1. Adleman, L.: Molecular computation of solutions to combinatorial problems. Science 266(5187), 1021–1024 (1994)
2. Lipton, R.: Using dna to solve np-complete problems. Science 268(4) (1995)
3. Liu, Q., Wang, L., Frutos, A., Condon, A., Corn, R., Smith, L., et al.: Dna computing on surfaces. Nature 403(6766), 175–179 (2000)
4. Ouyang, Q., Kaplan, P., Liu, S., Libchaber, A.: Dna solution of the maximal clique problem. Science 278(5337), 446–449 (1997)
5. Rothemund, P., Winfree, E.: The program-size complexity of self-assembled squares. In: Proceedings of the Thirty-Second Annual ACM Symposium on Theory of Computing, pp. 459–468. ACM (2000)
6. Winfree, E.: Algorithmic Self-Assembly of DNA. PhD thesis, California Institute of Technology (1998)
7. Winfree, E., Liu, F., Wenzler, L., Seeman, N., et al.: Design and self-assembly of two-dimensional dna crystals. Nature 394(6693), 539–544 (1998)
8. Barish, R., Rothemund, P., Winfree, E.: Two computational primitives for algorithmic self-assembly: Copying and counting. Nano Letters 5(12), 2586–2592 (2005)
9. Rothemund, P., Papadakis, N., Winfree, E.: Algorithmic self-assembly of dna sierpinski triangles. PLoS Biology 2(12), e424 (2004)
10. Brun, Y.: Solving np-complete problems in the tile assembly model. Theoretical Computer Science 395(1), 31–46 (2008)
11. Brun, Y.: Improving efficiency of 3-SAT-solving tile systems. In: Sakakibara, Y., Mi, Y. (eds.) DNA 16 2010. LNCS, vol. 6518, pp. 1–12. Springer, Heidelberg (2011)
12. Lagoudakis, M., LaBean, T.: 2-d dna self-assembly for satisfiability. In: DNA Based Computers V, vol. 54, pp. 141–154 (2000)
13. Brun, Y.: Arithmetic computation in the tile assembly model: Addition and multiplication. Theoretical Computer Science 378(1), 17–31 (2007)