Chaos and Reversibility in Simple Computations

Oleg Mazonka *mazonka@gmail.com*
Oct 2008

Abstract. This paper presents a simple computation – a multiplication of two two-bit numbers. The analysis of intermediate and final states is given. Also reversible functions are directly deduced.

Let x_1 and x_2 are 2 bits of two-bit number X and, y_2 and y_2 are of Y. Their multiplication Z can be expressed by a direct multiplication table

$$\begin{array}{cccc} & & x_2 & x_1 \\ & & y_2 & y_1 \\ \hline & u_2 & u_1 & z_1 \\ & v_2 & v_1 & \\ \hline z_4 & z_3 & z_2 & z_1 \end{array}$$

where z_i are 4 bits of the result Z=XY. The calculations are done in the following way

$$z_1 = x_1 y_1 \quad u_1 = x_2 y_1 \quad v_1 = x_1 y_2 \quad v_2 = x_2 y_2$$
$$z_2 = u_1 \wedge v_1 \quad u_2 = u_1 v_1 \quad z_3 = u_2 \wedge v_2 \quad z_4 = u_2 v_2$$

Let G be a function describing forbidden bit states. With each step of the calculation G changes to include the information of bit dependency. This can be expressed as

$$G_{i+1} \to G_i \mid x \wedge F$$

where | is Or, ^ is Xor (with Or having lower priority than Xor), and x calculated through $x = F$. For example, in $c = ab$ with initial G_0 the function G becomes $G = G_0 \mid c \wedge ab$. If $G_0 = \bar{a}$ i.e. $a = 1$, then $G = \bar{a} \mid c \wedge ab = \bar{a} \mid c \wedge b$, i.e. same as $c = b$. If $G_0 = a \wedge \bar{b}$ i.e. $a \neq b$, then $G = a \wedge \bar{b} \mid c \wedge ab = a \wedge \bar{b} \mid c$, i.e. c cannot be 1.

The function G of the final result of the computations can be expressed in only the resulting bits. This means that the system forgets some bits during the computation. When forgetting the bit a the function becomes

$$G_{i+1} \to G_i(a = 0)\, G_i(a = 1)$$

which means that the bit could be either 0 or 1.

In the above equation the function $G_i(a)$ has all the information about the bit a. Hence a can be expressed for example as $a = \overline{G_i(a = 1)}$ which means that if G gives 0 after substitution $a = 1$, then a is allowed to have value 0 and otherwise. Note that this calculation does require the future values of bit, but does not require the previous values, i.e. it is suitable for reversible computation.

The forward computation of multiplication of two-bit numbers is presented in the following table.

<table>
<tr><td>0</td><td>x_1, x_2, y_1, y_2</td><td>$G_0 = 0$</td><td>$G_0 = 0$</td></tr>
<tr><td>1</td><td>$z_1 = x_1 y_1$</td><td>G01=G00|z1^(x1*y1)</td><td>z1*!x1|z1*!y1|!z1*x1*y1</td></tr>
<tr><td>2</td><td>$u_1 = x_2 y_1$</td><td>G02=G01|u1^(x2*y1)</td><td>z1*!x1|z1*!y1|z1*u1*!x2|z1*!u1*x2|!z1*x1*y1|!z1*x1*u1|!z1*!x1*!y1*u1|!z1*!x1*y1*u1*!x2|!z1*!x1*y1*!u1*x2</td></tr>
<tr><td>3</td><td>$v_1 = x_1 y_2$</td><td>G03=G02|v1^(x1*y2)</td><td>z1*!x1|z1*!y1|z1*u1*!x2|z1*!u1*x2|z1*v1*!y2|z1*!v1*y2|!z1*x1*y1|!z1*x1*u1|!z1*x1*v1*!y2|!z1*x1*!v1*y2|!z1*!x1*!y1*u1|!z1*!x1*!y1*v1|!z1*!x1*y1*u1*!x2|!z1*!x1*y1*u1*v1|!z1*!x1*y1*!u1*x2|!z1*!x1*y1*!u1*v1</td></tr>
<tr><td>4</td><td>$v_2 = x_2 y_2$</td><td>G04=G03|v2^(x2*y2)</td><td>z1*!x1|z1*!y1|z1*u1*!x2|z1*!u1*x2|!z1*x1*y1|!z1*x1*u1|z1*u1*v1*!y2|z1*u1*v1*!v2|z1*u1*!v1*y2|z1*u1*!v1*v2|z1*!u1*v1*!y2|z1*!u1*v1*v2|z1*!u1*!v1*y2|z1*!u1*!v1*v2|!z1*!x1*!y1*u1|!z1*x1*x2*v1*!y2|!z1*x1*x2*v1*!v2|!z1*x1*x2*!v1*y2|!z1*x1*x2*!v1*v2|!z1*x1*!x2*v1*!y2|!z1*x1*!x2*v1*v2|!z1*x1*!x2*!v1*y2|!z1*x1*!x2*!v1*v2|!z1*!x1*y1*u1*!x2|!z1*!x1*y1*u1*v1|!z1*!x1*y1*!u1*x2|!z1*!x1*y1*!u1*v1|!z1*!x1*y1*!u1*v2|!z1*!x1*!y1*x2*v1|!z1*!x1*!y1*!x2*v1|!z1*!x1*!y1*!x2*v2|!z1*!x1*y1*u1*y2*!v2|!z1*!x1*y1*u1*!y2*v2|!z1*!x1*!y1*x2*y2*!v2|!z1*!x1*!y1*x2*!y2*v2</td></tr>
<tr><td>5</td><td>forget x_1</td><td>G05=G04(x1=0)*G04(x1=1)</td><td>z1*!y1|z1*u1*!x2|z1*!u1*x2|!z1*!y1*u1|z1*u1*v1*!y2|z1*u1*v1*!v2|z1*u1*!v1*y2|z1*u1*!v1*v2|z1*!u1*v1*!y2|z1*!u1*v1*v2|z1*!u1*!v1*y2|z1*!u1*!v1*v2|!z1*y1*u1*!x2|!z1*y1*u1*v1|!z1*y1*!u1*x2|!z1*y1*!u1*v1|!z1*y1*!u1*v2|!z1*y1*u1*y2*!v2|!z1*y1*u1*!y2*v2|!z1*!y1*x2*v1*!y2|!z1*!y1*x2*v1*!v2|!z1*!y1*!x2*v1*!y2|!z1*!y1*!x2*v2|!z1*!y1*x2*!v1*y2*!v2|!z1*!y1*x2*!v1*!y2*v2</td></tr>
<tr><td>6</td><td>forget x_2</td><td>G06=G05(x2=0)* G05(x2=1)</td><td>z1*!y1|!z1*!y1*u1|z1*u1*v1*!y2|z1*u1*v1*!v2|z1*u1*!v1*y2|z1*u1*!v1*v2|z1*!u1*v1*!y2|z1*!u1*v1*v2|z1*!u1*!v1*y2|z1*!u1*!v1*v2|!z1*y1*u1*v1|!z1*y1*!u1*v1|!z1*y1*!u1*v2|!z1*!y1*v1*!y2|!z1*y1*u1*y2*!v2|!z1*y1*u1*!y2*v2|!z1*!y1*!v1*!y2*v2</td></tr>
<tr><td>7</td><td>forget y_1</td><td>G07=G06(y1=0)*G06(y1=1)</td><td>!z1*u1*v1|z1*u1*v1*!y2|z1*u1*v1*!v2|z1*u1*!v1*y2|z1*u1*!v1*v2|z1*!u1*v1*!y2|z1*!u1*v1*v2|z1*!u1*!v1*y2|z1*!u1*!v1*v2|!z1*u1*y2*!v2|!z1*u1*!y2*v2|!z1*!u1*v1*!y2|!z1*!u1*!v1*!y2*v2</td></tr>
<tr><td>8</td><td>forget y_2</td><td>G08=G07(y2=0)* G07(y2=1)</td><td>z1*!u1*v2|!z1*u1*v1|z1*u1*v1*!v2|z1*u1*!v1*v2</td></tr>
<tr><td>9</td><td>$z_2 = u_1 \wedge v_1$</td><td>G09=G08|z2^(u1^v1)</td><td>!z1*u1*v1|!z1*u1*!z2|z1*u1*v1*!v2|z1*u1*v1*z2|z1*u1*!v1*v2|z1*u1*!v1*!z2|z1*!u1*v1*v2|z1*!u1*v1*!z2|z1*!u1*!v1*v2|z1*!u1*!v1*z2|!z1*!u1*v1*!z2|!z1*!u1*!v1*z2</td></tr>
<tr><td>10</td><td>$u_2 = u_1 v_1$</td><td>G10=G09|u2^(u1*v1)</td><td>!z1*u1*v1|!z1*u1*!z2|!z1*u1*u2|z1*u1*v1*!v2|z1*u1*v1*z2|z1*u1*v1*!u2|z1*u1*!v1*v2|z1*u1*!v1*!z2|z1*u1*!v1*u2|z1*!u1*v1*v2|z1*!u1*v1*!z2|z1*!u1*v1*u2|z1*!u1*!v1*v2|z1*!u1*!v1*z2|z1*!u1*!v1*u2|!z1*!u1*v1*!z2|!z1*!u1*v1*u2|!z1*!u1*!v1*z2|!z1*!u1*!v1*u2</td></tr>
<tr><td>11</td><td>$z_3 = u_2 \wedge v_2$</td><td>G11=G10|z3^(u2^v2)</td><td>!z1*u1*v1|z1*u1*v1*!v2|z1*u1*v1*z2|z1*u1*v1*!u2|z1*u1*v1*z3|z1*u1*!v1*v2|z1*u1*!v1*!z2|z1*u1*!v1*u2|z1*u1*!v1*z3|z1*!u1*v1*v2|z1*!u1*v1*!z2|z1*!u1*v1*u2|z1*!u1*v1*z3|z1*!u1*!v1*v2|z1*!u1*!v1*z2|z1*!u1*!v1*u2|z1*!u1*!v1*z3|!z1*u1*v2*!z2|!z1*u1*v2*u2|!z1*u1*v2*!z3|!z1*u1*!v2*!z2|!z1*u1*!v2*u2|!z1*u1*!v2*z3|!z1*!u1*v1*v2*!z2|!z1*!u1*v1*v2*u2|!z1*!u1*v1*v2*!z3|!z1*!u1*v1*!v2*!z2|!z1*!u1*v1*!v2*u2|!z1*!u1*v1*!v2*z3|!z1*!u1*!v1*v2*z2|!z1*!u1*!v1*v2*u2|!z1*!u1*!v1*v2*!z3|!z1*!u1*!v1*!v2*z2|!z1*!u1*!v1*!v2*u2|!z1*!u1*!v1*!v2*z3</td></tr>
<tr><td>12</td><td>$z_4 = u_2 v_2$</td><td>G12=G11|z4^(u2*v2)</td><td>!z1*u1*v1|z1*u1*v1*!v2|z1*u1*v1*z2|z1*u1*v1*!u2|z1*u1*v1*z3|z1*u1*v1*!z4|z1*u1*!v1*v2|z1*u1*!v1*!z2|z1*u1*!v1*u2|z1*u1*!v1*z3|z1*u1*!v1*z4|z1*!u1*v1*v2|z1*!u1*v1*!z2|z1*!u1*v1*u2|z1*!u1*v1*z3|z1*!u1*v1*z4|z1*!u1*!v1*v2|z1*!u1*!v1*z2|z1*!u1*!v1*u2|z1*!u1*!v1*z3|z1*!u1*!v1*z4|!z1*u1*v2*!z2|!z1*u1*v2*u2|!z1*u1*v2*!z3|!z1*u1*v2*z4|!z1*u1*!v2*!z2|!z1*u1*!v2*u2|!z1*u1*!v2*z3|!z1*u1*!v2*z4|!z1*!u1*v1*v2*!z2|!z1*!u1*v1*v2*u2|!z1*!u1*v1*v2*!z3|!z1*!u1*v1*v2*z4|!z1*!u1*v1*!v2*!z2|!z1*!u1*v1*!v2*u2|!z1*!u1*v1*!v2*z3|!z1*!u1*v1*!v2*z4|!z1*!u1*!v1*v2*z2|!z1*!u1*!v1*v2*u2|!z1*!u1*!v1*v2*!z3|!z1*!u1*!v1*v2*z4|!z1*!u1*!v1*!v2*z2|!z1*!u1*!v1*!v2*u2|!z1*!u1*!v1*!v2*z3|!z1*!u1*!v1*!v2*z4</td></tr>
<tr><td>13</td><td>forget u_1</td><td>G13=G12(u1=0)*G12(u1=1)</td><td>z1*!v1*v2|z1*!v1*u2|z1*!v1*z3|z1*!v1*z4|z1*v1*v2*z2|z1*v1*v2*!u2|z1*v1*v2*z3|z1*v1*v2*!z4|z1*v1*!v2*!z2|z1*v1*!v2*u2|z1*v1*!v2*z3|z1*v1*!v2*z4|!z1*v1*v2*!z2|!z1*v1*v2*u2|!z1*v1*v2*!z3|!z1*v1*v2*z4|!z1*v1*!v2*!z2|!z1*v1*!v2*u2|!z1*v1*!v2*z3|!z1*v1*!v2*z4|!z1*!v1*v2*u2|!z1*!v1*v2*!z3|!z1*!v1*v2*z4|!z1*!v1*!v2*u2|!z1*!v1*!v2*z3|!z1*!v1*!v2*z4</td></tr>
<tr><td>14</td><td>forget u_2</td><td>G14=G13(u2=0)*G13(u2=1)</td><td>z1*!v1*v2|z1*!v1*z3|z1*!v1*z4|z1*v1*v2*z2|z1*v1*v2*z3|z1*v1*v2*!z4|z1*v1*!v2*!z2|z1*v1*!v2*z3|z1*v1*!v2*z4|!z1*v1*v2*!z2|!z1*v1*v2*!z3|!z1*v1*v2*z4|!z1*v1*!v2*!z2|!z1*v1*!v2*z3|!z1*v1*!v2*z4|!z1*!v1*v2*!z3|!z1*!v1*v2*z4|!z1*!v1*!v2*z3|!z1*!v1*!v2*z4</td></tr>
<tr><td>15</td><td>forget v_1</td><td>G15=G14(v1=0)*G14(v1=1)</td><td>z1*v2*z2|z1*v2*z3|z1*v2*!z4|z1*!v2*z3|z1*!v2*z4|!z1*v2*!z3|!z1*v2*z4|!z1*!v2*z3|!z1*!v2*z4</td></tr>
<tr><td>16</td><td>forget v_2</td><td>G16=G15(v2=0)*G15(v2=1)</td><td>!z1*z4|z1*z2*z3|z1*z2*z4|z1*!z2*z3</td></tr>
</table>

The last step gives the G function expressed all in the result bits

$$G_{16} = \bar{z}_1 z_4 \mid z_1 (z_3 \mid z_2 z_4)$$

which has 5 atomic steps in calculation (if written in the same manner). This value is the measure of chaos of the given calculation process.

The interpretation of that measure can be given as following. Imagine a space defined by the initial set of variables and a set of constrains. Constrains define the allowed states of the system. And the function which describes all these states is G . This function is expressed in some language and hence has some measurable effort required evaluating it. Next let the system evolve in the way that other variables can be added increasing the space dimension, removed, and constrains also can be changed. If the area of the allowable states smooth (has symmetries) within the space, the representation of G should not be complex. But when the area become wrinkled (less symmetric), the information to describe it increases. Hence the representation of G becomes more complex and bigger. This is where the system becomes less descriptive – chaotic.

If the function G is known at each step of the process, the process can be unwound back in time. Let V_2 be a function depending on the final bit set. As it is said above, it is possible to obtain the value of v_2 expressed via the final bits:

$$V_2 = \overline{G_{15}(v_2 = 1)}$$

Once V_2 is found one can continue up the steps with all the rest variables:

$V_1 = \overline{G_{14}(v_2 = V_2, v_1 = 1)}$

$U_2 = \overline{G_{13}(v_2 = V_2, v_1 = V_1, u_2 = 1)}$

and so on.

V2	!z1*z3*!z4|z1*!z2*!z3*z4
V1	!z1*z2*!z4|z1*z2*!z3*!z4|z1*!z2*!z3*z4
U2	z1*!z2*!z3*z4
U1	z1*!z2*!z3*z4
Y2	!z1|z2*!z3*!z4|!z2*!z3*z4
Y1	z1|z2*z4|!z2*!z3|!z2*z4
X2	!z1*z3*!z4|z1*!z2*!z3*z4
X1	z1|z2*!z4

The above arguments show that there is a simple direct algorithm for obtaining the reverse computations and measuring chaos by finding the most compact representation for the G function. Unfortunately this works easy enough only for simple models. If one increases, for example, the number of bits of the multiplied numbers, the function G grows so large that it is becomes impractical. I was able to do this up to 2 12-bit numbers. A possible way to avoid this problem is to find a correspondence between symmetry of the calculation and G function. Then the information about the symmetry can be applied to the representation of G making it smaller.

One interesting conclusion from this can be made about density of the space. If starting with a set of variables without any constrain ($G = 0$), the bidirectional process of adding (computing) a new variable and immediately removing another (forgetting), would keep G equal to 0; thus, making the process linearly reversible. [Bidirectional here is that number of initial states is equal to a number of final states.] In practice it might be feasible just to keep a number of total variables close to the initial having the function not exactly equal to 0 but keeping it enough small. In particular one can claim that if such a multiplication algorithm can be found, then the factorization problem is solved.