Progress Report

Kan Shi

October 2013

1 Design of Digit-parallel Online Adder

2 Design and Optimization of A Digit-parallel Online Multiplier

2.1 Online Multipilication Algorithm

Typically the online multiplication is performed in a recursive manner. The algorithm to generate 1-digit product z_j is given by Algorithm 1, where j is the number of iteration, r denotes the radix and N represents the word-length of input operands.

Algorithm 1 Online Multiplication

Initialization: $X[-\delta] = Y[-\delta] = P[-\delta] = 0$

Recurrence: $for\ \ j = -\delta,\ \ -\delta+1,\ \ \cdots,\ 1,\ 2,\ \cdots,\ N-1\ \ do$

$$\begin{array}{rcl} H[j] & = & r^{-\delta}\left(x_{j+\delta}\cdot Y[j+1] + y_{j+\delta}\cdot X[j]\right) \\ W[j] & = & P[j] + H[j] \\ z_j & = & sel(\widehat{W}[j]) \\ P[j+1] & = & r\left(W[j] - z_j\right) \end{array} \tag{1}$$

The operands and product are signed number in the range $(-1, 1)$, and they can be represented with signed digits from the set $\{-a, \cdots, a\}$. For instance in a radix-2 online multiplier, the most common digit set is $\{-1, 0, 1\}$. The operands and product at iteration j can be represent by (2).

$$X[j] = \sum_{i=0}^{j+\delta-1} x_i r^{-i-1},\ \ Y[j] = \sum_{i=0}^{j+\delta-1} y_i r^{-i-1},\ \ Z[j] = \sum_{i=0}^{j-1} z_i r^{-i-1} \tag{2}$$

In addition, $sel(\widehat{W}[j])$ is the function to select the product digit at iteration j. The selection is accomplished by using a discretization algorithm [xxx] based on the knowledge of $\widehat{W}[j]$, which is an estimate of the residue $W[j]$. In the following of this article, $\widehat{W}[j]$ is obtained by truncating $W[j]$ to its first few MSDs.

Note that in this algorithm, the first non-zero product is obtained when $j = 0$, while the MSD of the N-digit input operand is applied when $j = -\delta$. In order to acquire the most significant N-digit of product, x_j and y_j are assigned to be 0 when $j = N-\delta, \cdots, N-1$. Similarly if $2N$-digit is required for product, x_j and y_j should be assigned as 0 when $j =$

$N-\delta, \cdots, 2N-1$. In this article, we will focus on the situation where the multiplication stops when the first N-digit MSDs of products are obtained. Because normally the remaining N-digit LSDs will be truncated in real computations such as $a \times b + c$ where $a,\ b,\ c$ are N-digit numbers.

2.2 Radix-2 Online Multiplier

Currently we consider the online multiplication algorithm in radix-2 as it is the most commonly used situation in computer arithmetic. In a radix-2 online multiplier, the design choices are summarized in Table 1 [xxx].

Table 1: Design Choice for Radix-2 Online Multiplication.

r	Digit Set	$\widehat{W}[j]$	δ
2	$\{\bar{1}, 0, 1\}$	$w_{-2}w_{-1}.w_0$	3

Hence (1) can be modified as given by (3).

$$\begin{array}{rcc} H[j] & = & 2^{-3}(x_{j+3} \cdot Y[j+1] + y_{j+3} \cdot X[j]) \\ W[j] & = & P[j] + H[j] \\ z_j & = & sel(\widehat{W}[j]) \\ P[j+1] & = & 2(W[j] - z_j) \end{array} \tag{3}$$

The selection function is given by (4) [xxx].

$$z_j = sel(\widehat{W}[j]) = \begin{cases} 1 & \text{if } \widehat{W}[j] \geqslant \frac{1}{2} \\ 0 & \text{if } -\frac{1}{2} \leqslant \widehat{W}[j] < \frac{1}{2} \\ \bar{1} & \text{if } \widehat{W}[j] < -\frac{1}{2} \end{cases} \tag{4}$$

2.3 Implementation of Radix-2 Online Multiplication Algorithm

The online multiplication algorithm can be implemented in either a digit-serial fashion or a digit-parallel fashion, of which the general structure is demonstrated in Fig.xxx(a) and Fig.xxx(b) respectively. In the digit-serial implementation, the input operands are given digit by digit. Correspondingly the product is generated one digit per cycle. Whereas in the digit-parallel architecture, N-digit operands are given simultaneously, and the N-digit MSDs of products can be sampled at the same time. In comparison to the digit-serial structure where a general architecture is required in order to process inputs for all j, digit-parallel structure offers the opportunities to optimize the logic of each stage with respect to the given inputs. For example when $j = N-\delta, \cdots, N-1$, we have $x_j = y_j = 0$ as described earlier. In this case $H[j] = 0$ and $W[j] = P[j]$ in (1) and (3), that is, the signed-digit by vector multiplication can be removed from this stages and hence the area is reduced. Therefore in the following of this section, we describe the details of implementation and optimization for the digit-parallel online multiplication.

2.3.1 Signed-digit by Vector Multiplication Module

As can be seen in (3), two signed-digit by vector multiplication (SDVM) modules are required to generate $H[j]$. Implementing SDVM for $r > 2$ is difficult, since it is basically a normal

multiplier and the computation time will depend on the precision of operands. This will introduce a large overhead of delay and area. However for $r = 2$ the structure becomes straight forward, because x_j and y_j can only be ± 1 or 0. As an example for $y_{j+3} \cdot X[j]$ we have (5).

$$y_{j+3} \cdot X[j] = \begin{cases} X[j] & \text{if } y_{j+3} = 1 \\ \overline{X}[j] + 1 & \text{if } y_{j+3} = \bar{1} \\ 0 & \text{otherwise} \end{cases} \tag{5}$$

2.3.2 Generation of Residue $W[j]$

We can see from (3) that addition is required to generate $W[j]$. Instead of using conventional adder, such as the ripple carry adder, the carry-save adder is employed so that the addition time is independent of the precision. Correspondingly $W[j]$ and $P[j]$ are represented in the carry-save form. Hence we have (6) which can be implemented using a carry-save adder with 4 inputs and 2 outputs as shown in Fig.xxx.

$$(WC[j],\ WS[j]) = PC[j] + PS[j] + 2^{-3}x_{j+3}Y[j+1] + 2^{-3}y_{j+3}X[j] \tag{6}$$

The word-length of the carry-save adder is determined by the word-length of inputs. For a N-digit online multiplier, a $(N+5)$-bit carry-save adder is required for (6) because of the 3-bit shift and 2-bit integer is needed for $\widehat{W}[j]$.

2.3.3 Selection Function

The selection function can be directly implemented using combinational logic based on (4), where z_j is represented in the signed-digit form coding as $(z_+,\ z_-)$. The expression for z_+ and z_- are given by (7) where $w_{-2}w_{-1}.w_0$ is used to represent $\widehat{W}[j]$ as shown in Table 1.

$$z_+ = \overline{w}_{-2}(w_{-1} + w_0), \quad z_- = w_{-2}(\overline{w}_{-1} + \overline{w}_0) \tag{7}$$

2.3.4 Generation of Propagation Output $P[j+1]$

According to (3) the fractional bits of $P[j+1]$ are identical to $2W[j]$ as z_j is an integer number. Moreover w_{-2} will be discarded after shift operation to keep the word-length of $P[j+1]$ to be $N+5$ bits. Therefore only the first integer bit of $P[j+1]$ should be calculated as given by (8), while the remaining bits are directly wired to the corresponding bits in $2W[j]$. Note that $\oplus$ in (8) denotes the exclusive OR operation.

$$p_{-2} = 2(w_{-1} \ \oplus \ (z_+ \ \oplus \ z_-)) \tag{8}$$

2.3.5 Optimization for Area Reduction

We have discussed earlier that the SDVM module is not required in the last δ stages. Besides in the first stage, we have $W[j] = 2^{-3}x_{j+3}Y[j+1]$ meaning that only one SDVM is needed. In addition the carry-select adder is not necessary for the first stage since $P[-\delta] = 0$. For the first δ stages, we also have $z_j = 0$ and therefore $P[j+1] = 2W[j]$. This means the selection function can be removed for those stages. Moreover word-length optimization can be performed for $W[j]$ and $P[j]$ in all stages instead of keeping $N+5$ bits. The overall area saving after optimization can be $xxx\%$.

3 Probabilistic Model of Overclocking Error in Online Multiplier

3.1 Annihilation of the Propagation Chain

While the delay in a digit-parallel online multiplier might be derived from many sources such as the computation delay to generate outputs, the overall delay will eventually be determined by the longest propagation delay between stages with increasing operand word-lengths. Let the propagation delay between two adjacent stages in a N-digit olMult be dented by μ, hence the delay of the longest chain which propagates from the MSD to the LSD is given by d_w as shown in (9).

$$d_w = (N + \delta - 1) \cdot \mu \tag{9}$$

However, we note that the chain is annihilated at a certain stage if the propagation inputs of this stage change while the propagation outputs keep stable. This will shrink the value of d_w, and there are two possible cases specifically. As an example for the first case, assume at time t ($t > \mu$) the value of propagation inputs and outputs of a stage S_i to be $Pin(t)$ and $Pout(t)$ respectively. After delay μ, the input value changes to $Pin(t + \mu)$ and stabilizes thereafter, while the output of this stage remains $Pout(t)$. Hence for the next stage S_{i+1}, the input of which varies from $Pout(t - \mu)$ to $Pout(t)$, as illustrated in Figure.xxx. Under this situation, the chain delay to S_{i+1} is reduced by μ because of the annihilation.

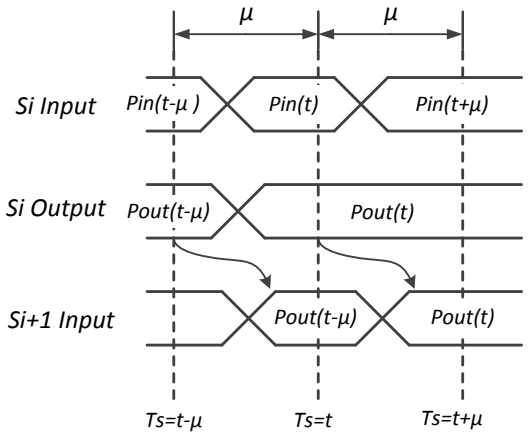

Figure 1: Timing wave.

For the second case, the current chain will be completely annihilated and a new chain will be generated at a given stage if $Pout(t) = Pout(0)$ for $t = \mu,\ 2\mu,\ \cdots$. Therefore the worst case delay is given by (10) where d_p denotes the delay of the p^{th} propagation chain.

$$d'_w = max(d_1,\ d_2,\ \cdots) \tag{10}$$

In the following of this section, detailed analysis for both cases will be described and the worst-case delay of the olMult will be discussed.

3.2 Worst-case Analysis in Online Multiplier

3.2.1 Two Observations

From (3) two observations can be made under the assumption that all signals are reset to 0 initially.

Observation 1. *The two integer bits of* $W[j]$ *and* $2P[j+1]$ *are either* 00 *or* 11.

This observation can be justified by contradiction. All the combinations of $\widehat{W[j]}$ and the corresponding z_j, the most significant 3 bits of $P[j+1]$ and the 2 integer bits of $2P[j+1]$ are listed in Table 2. For $j=-3$, we have (11) according to (3).

$$\begin{cases} W[-3] &= 2^{-3}(y_1 \cdot X[-3]) \\ H[-2] &= W[-3] \end{cases} \tag{11}$$

Hence $\widehat{W[-3]}$ is either 11.1 or 00.0, with the corresponding $2P[-2]$ being 00 or 11 which is the propagation input for the next stage. For $j > -3$, the 3 MSBs of $H[j]$ in (3) is either 00.0 or 11.1 due to the shift of binary point. Also as seen in Table.xxx, if the two integer bits of $\widehat{W[j]}$ are identical, the integer bits of $2P[j+1]$ will be the same. Therefore the propagation of $2P[j+1]$ will not generate diverse bits in the integer part of $W[j]$.

Table 2: All the combinations of $\widehat{W[j]}$ and the corresponding z_j, $P[j+1]$ and $2P[j+1]$.

$\widehat{W[j]}$	z_j	$P[j+1]$ (3 MSBs)	$2P[j+1]$ (2 MSBs)
00.0	0	00.0	00
00.1	1	11.1	11
01.0	1	00.0	00
01.1	1	00.1	01
10.0	$\bar{1}$	11.0	10
10.1	$\bar{1}$	11.1	11
11.0	$\bar{1}$	00.0	00
11.1	0	11.1	11

Observation 2. $P[j+1]$ *will not change if only the integer bits of* $W[j]$ *change to a new value.*

According to Observation 1, there are only four possible cases for the change of $W[j]$ as summarized in Table.xxx. This observation describes the situation where a propagation chain annihilates but not necessarily generating a new chain, because $2P[j+1]$ may not maintain its initial value at all times.

3.2.2 Annihilation of the Propagation Chain in An olMult

As stated in Section 3.1, the annihilation of propagation chain would leads to a reduction of the propagation delay. It is worthwhile exploring the conditions of occurrence of Observation 2. For instance in a 3-digit olMult, there are 6 stages in total, as shown in Fig.xxx. If $x_{-3} \neq 0$ and/or $y_{-3} \neq 0$, no more than 6 MSBs of $P[-2]$ will change with respect to the initial value at

Table 3: Ob2.

$\widehat{W[j]}$	z_j	$P[j+1]$ (3 MSBs)	$2P[j+1]$ (2 MSBs)
$00.0 \to 11.0$	$0 \to \bar{1}$	00.0	00
$00.1 \to 11.1$	$1 \to \bar{0}$	11.1	11
$11.0 \to 00.0$	$\bar{1} \to 0$	00.0	00
$11.1 \to 00.1$	$0 \to 1$	11.1	11

$t = 0$. This is because in (11) $y_1 \cdot X[-3]$ contains only 1 fractional bit and 2 integer bits. Hence maximally 5 MSBs of $2P_{[-2]}(0)$ will be updated and thereafter stabilized. It is worth noting that this is irrelevant to the word-length of the olMult. Additionally as assumed previously in the timing model, at $t = 0$ the computation of $H[j]$ ($j \in [-2, 2]$) in (3) is finished. This results in $2P_{[j]}(0)$, which will be updated sequentially with a delay of μ, triggered by the propagation of $2P_{[-2]}(0)$. During each propagation, the number of the altered MSBs will be reduced by 1, because of the shifting of the binary point in $2P[j]$. For example, $2P_{[-1]}(t)$ stabilizes when $t = \mu$, and the altered MSBs for $2P_{[-1]}(0) \to 2P_{[-1]}(\mu)$ are 4 bits. This process repeats until S_1, of which the changed MSBs for $2P_{[1]}(3\mu) \to 2P_{[1]}(4\mu)$ are 2 bits. According to Observation 2, the chain triggered by $2P_{[-2]}(0)$ is annihilated at S_1. Therefore the propagation delay from S_{-3} to S_2 is 4μ.

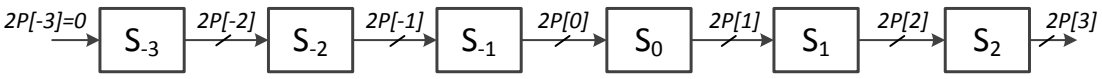

Figure 2: Propagation chain in 3-digit olMult

It is worth noting that in this example, the chain still propagates through S_2 without generating a new chain. Because it is hardly to achieve $2P_{[2]}(t) \equiv 2P_{[2]}(0)$ with $t = \mu,\ 2\mu, \cdots$, due to the propagation of $2P_{[j]}(t)$ where $j \in [-2, 1]$. The maximum number of the changed MSBs for $2P_{[2]}(3\mu) \to 2P_{[2]}(4\mu)$ is 3, because this is actually triggered by $2P_{[-1]}(0)$, which varies 6 MSBs in comparison to its initial value. Likewise there will be maximally 4 MSBs changing for $2P_{[3]}(4\mu) \to 2P_{[3]}(5\mu)$.

The above changing process happens in a 3-digit olMult when $P[-2] \neq 0$, which is only determined by the inputs of the first stage regardless of the other stages. Actually in other situations, the overall propagation delay will not be increased. For example for $t = 0,\ \mu,\ \cdots$ if $P_{[-2]}(t) = 0$ while $P_{[-1]}(t) \neq 0$, the chain starts from $2P_{[-1]}(0)$. Based on a similar analysis, the chain will finally annihilate at S_2, of which we have 2 MSBs changing for $2P_{[2]}(3\mu) \to 2P_{[2]}(4\mu)$, yielding a delay of 4μ. This value shrinks to 3μ if $P_{[-2]}(t) = P_{[-1]}(t) = 0$ while $P_{[0]}(t) \neq 0$ for $t = 0,\ \mu,\ \cdots$.

3.2.3 State Transition Graph in An olMult

For an olMult with a greater word-length, the aforementioned process is still valid. Annihilation happens at S_i when there are only 2 MSBs of $2P_{[i]}(t)$ update before it is stabilized. Otherwise the propagation continues regardless of the inputs $x_{j+4},\ y_{j+4},\ X[j]$ and $Y[j+1]$ of S_j. The propagation process from S_1 can be illustrated using a state transition graph. One possible

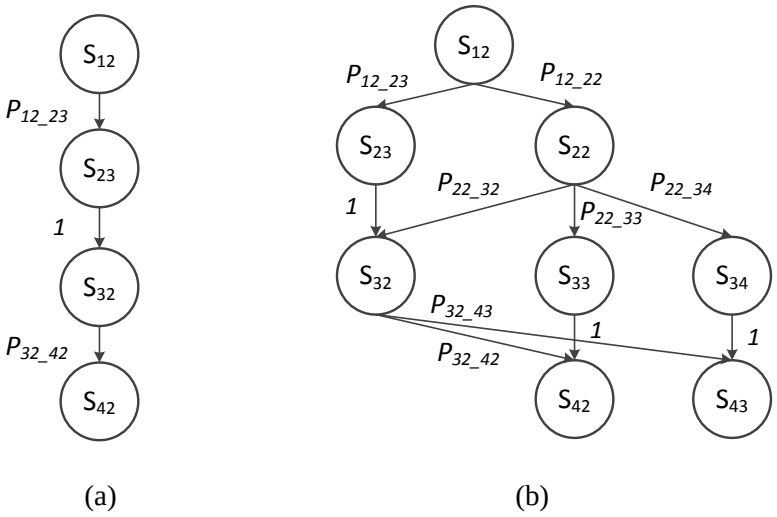

Figure 3: State transition graph in a 5-digit olMult when $2P_{[-2]}(0) \neq 0$. (a) One possible transition path from $Stage1$ to $Stage4$ with transition probabilities labelled on the edges. (b) All possible transition paths starting from S_1.

transition path from S_1 in a 5-digit olMult is shown in Fig.xxx, where a state S_{mn} denotes the propagation input of $Stage\ m$ changes n MSBs prior to stabilization. The transition probabilities are labelled on the edges. For instance P_{12_23} denotes the conditional probability $P(S_{23} \mid S_{12})$. Note that $P_{23\_32} = 1$ because annihilation only happens when $n = 2$ in S_{mn}. In this case, the probability of this transition path is given by (12), where $P(S_{12})$ is determined by the value of x_1 and y_1 because S_{12} will happen when $2P_{[-2]}(0) \neq 0$.

$$\begin{aligned} P_{trans} &= P(S_{42},\ S_{32},\ S_{23},\ S_{12}) \\ &= P(S_{42} \mid S_{32},\ S_{23},\ S_{12})P(S_{32} \mid S_{23},\ S_{12})P(S_{23} \mid S_{12})P(S_{12}) \\ &= P_{32\_42} \cdot 1 \cdot P_{12\_23} \cdot P(S_{12}) \end{aligned} \tag{12}$$

The delay of this transition path can also be determined by counting the number of states S_{m2}. First, there is no annihilation prior to S_{12}. Second, S_{42} is excluded because it is not necessary to wait for the propagation output of this stage to generate before sampling z_4. Hence there are totally 2 annihilation states S_{12} and S_{32} in this path. According to (9) the worst case delay without any annihilation is $d_w = 7\mu$. Delay of a specific transition path is given by (13). Hence the delay of this path is 5μ.

$$d = d_w - \mu \cdot AnnihilationNo. \tag{13}$$

Additionally as discussed earlier, there might be other possible states when annihilation happens. The complete transition graph of a 5-digit olMult is shown in Figure 3.2.3.

(extra description about why Fig.xxx looks like this)

3.2.4 Conclusion

To summarize in a 5-digit olMult, the worst case delay is 5μ, instead of 7μ as predicted in (9). This example can be generalized to an olMult with any word-length based on the similar

analysis. Figure xxx illustrates the critical path delay with respect to given word-lengths under two cases: one is predicted from (9), while the other is obtained through the worst case analysis as described above. Between the two lines we can always observe a gap, which increases with the growth of the operand word-length. This finding indicates that instead of operating at a frequency reported by the timing analysis tool, we can safely increase the operating frequency to a large extend (e.g. 38.9% speedup for a 16-digit olMult) without timing violations, because in an olMult there is no chain propagating from the first stage to the final stage. Note that Figure.xxx is observed based on the aforementioned simple timing models. The experimental results based on real timing information on an FPGA will be given in Section.xxx.

In the remaining of this section, we investigate the situation where the operating frequency is further increased such that timing violation happens. Probabilistic models of both magnitude and probability of the overclocking error will be detailed.

3.3 Probability of Overclocking Error in Online Multiplier

3.4 Magnitude of Overclocking Error in Online Multiplier

4 Probabilistic Model of Truncation Error in Online Operators