Proof of the Latency Optimization

This short document is an appendix to the paper [1]. This paper presents several optimizations for Montgomery modular multipliers in GF(P) based on *hyper-threading* to avoid "bubbles" in DSP slices pipeline: several *logical multipliers* (LMs) compute independent multiplications at the same time in one *physical* hyper-threaded modular multiplier (HTMM).

Input : $A = \sum_{i=0}^{s-1} a_i\, 2^{iw}$, $B = \sum_{j=0}^{s-1} b_j\, 2^{jw}$, $P = \sum_{j=0}^{s-1} p_j\, 2^{jw}$ such that $0 \leq A, B < 2P$
Requires: $4P < 2^m$ and $P' = -P^{-1} \bmod 2^w$
Output : $T \equiv (A\, B\, 2^{-m}) \bmod P$, $0 \leq T < 2P$
begin
  $t_{0\ldots s-1} \leftarrow 0$
1 **for** $i = 0$ **to** $s-1$ **do**
2 **for** $j = 0$ **to** $s-1$ **do**
3 $v_j \leftarrow t_0 + a_i \times b_0$
4 $(d, u_j) \leftarrow v_j + d$
5 $q_i \leftarrow (v_0 \times P') \bmod 2^w$
6 **for** $j = 0$ **to** $s-1$ **do**
7 $(c, t_{j-1}) \leftarrow u_j + q_i \times p_j + c$
8 **return** $T = t_{-1}^{(\text{next})}\, 2^{(s-1)w} + \sum_{j=0}^{s-2} t_j\, 2^{jw}$

Algorithm 1: MMM algorithm based on [2] and [3], modified to integrate latency optimization from [1].

To reduce the latency of one MMM in HTMM, we proposed to overlap the computations of independent MMMs in successive LMs. This optimization is described in Algorithm 1 and illustrated in Figure 1. In the original algorithm provided in [1] (top part in Figure 1), computations of $w \times m$ bits partial products ($a_i \times B$ and $q_i \times P$) in inner loops (index j) require 4 clock cycles, plus one extra clock for carry propagation (words of index $j = s = 4$).

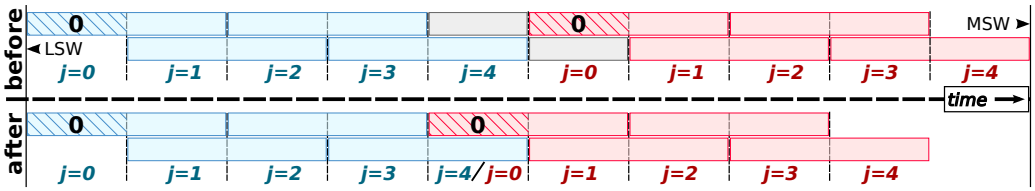

Fig. 1. Schedule example for two 128-bit MMMs computed in two successive LMs (colors), *before* and *after* latency reduction (figure from paper [1]). Rectangles are w-bit words computed at iterations of index j.

In the optimized version (bottom part in Figure 1), the result t_{s-1} of the first LM at iteration $j = 4$ and the result t_{-1} of second LM at iteration $j = 0$ share the same w-bit word. This optimization is possible as the low significant word t_{-1} of the result is always zero and discarded in MMM. We use this word during each outer loop iteration to store intermediate values for the next LM. Here, $^{(\text{curr})}$, $^{(\text{prev})}$ and $^{(\text{next})}$ denote the values computed respectively in the current, previous and next LM.

We will now show that sharing one w-bit word between successive LMs to store two different intermediate values throughout intermediate computations does not modify the expected results. We need to verify that $t_{-1}^{(\text{next})} = t_{s-1}^{(\text{curr})} < 2^w$ thanks to the properties of the MMM algorithm, and that overlapping MSW t_{s-1} of current LM with LSW t_{-1} of next LM does not create "hidden" carries between results of successive independent MMMs.

In the modified Algorithm 1, carries d and c are propagated between successive LMs. As in the original algorithm from [1], we here define $u_s = d$ after the last iteration of first inner loop (in lines 2 to 4 in Algorithm 1), and $t_{s-1} = u_s + c$ after the last iteration of the second inner loop (in lines 6 and 7 in Algorithm 1).

To compute T_i in the current LM, we need the value $t_{-1}^{(\text{next})}$, computed at iteration $j = 0$ of second inner loop (line 7 in Algorithm 1) in next LM:

$$t_{-1}^{(\text{next})} = \left(u_0^{(\text{next})} + q_i^{(\text{next})} \times p_0 + c^{(\text{curr})}\right) \bmod 2^w. \tag{1}$$

Due to the propagation of carry words u_s between successive LMs, we have

$$u_0^{(\text{next})} = \left(T_{i-1}^{(\text{next})} + a_i^{(\text{next})} \times b_0^{(\text{next})} + u_s^{(\text{curr})}\right) \bmod 2^w. \tag{2}$$

However, "reduction quotients" q_i are computed before propagation of u_s carry words, and then

$$q_i^{(\text{next})} = \left((T_{i-1}^{(\text{next})} + a_i^{(\text{next})} \times b_0^{(\text{next})} \times P' \right) \bmod 2^w. \tag{3}$$

Putting together equations (1), (2) and (3), we obtain the equation

$$\begin{aligned} t_{-1}^{(\text{next})} &= (T_{i-1}^{(\text{next})} + a_i^{(\text{next})} \times b_0^{(\text{next})} + u_s^{(\text{curr})} + q_i^{(\text{next})} \times p_0 + c^{(\text{curr})}) \bmod 2^w \\ &= \left((T_{i-1}^{(\text{next})} + a_i^{(\text{next})} \times b_0^{(\text{next})} + q_i^{(\text{next})} \times p_0) + (u_s^{(\text{curr})} + c^{(\text{curr})}) \right) \bmod 2^w. \end{aligned} \tag{4}$$

Due to Montgomery algorithm, the LSW is equal to 0:

$$T_{i-1} + a_i \times b_0 + q_i \times p_0 \equiv 0 \bmod 2^w.$$

Then, from equation (4), the value $t_{-1}^{(\text{next})}$ is now

$$t_{-1}^{(\text{next})} = \left(u_s^{(\text{curr})} + c^{(\text{curr})} \right) \bmod 2^w.$$

Value $u_s^{(\text{curr})}$ is computed as follows:

$$u_s^{(\text{curr})} = \left\lfloor \frac{T_{i-1}^{(\text{curr})} + a_i^{(\text{curr})} \times B^{(\text{curr})} + u_s^{(\text{prev})}}{2^m} \right\rfloor.$$

From properties of MMM in Algorithm 1, we have

$$\begin{cases} 4P < 2^m \\ 0 \leq B, T_{i-1} < 2P < 2^{m-1} \\ 0 \leq a_i < 2^w \end{cases} \tag{5}$$

Then the value of $u_s^{(\text{curr})}$ is bounded by

$$0 \leq u_s^{(\text{curr})} < 2^{w-1}. \tag{6}$$

Similarly, the value of c is

$$c^{(\text{curr})} = \left\lfloor \frac{\sum_{j=0}^{s-1} \left(u_j^{(\text{curr})} 2^{jw} \right) + u_s^{(\text{prev})} + q_i^{(\text{curr})} \times P + c^{(\text{prev})}}{2^m} \right\rfloor.$$

From boundaries in (5) we have

$$0 \leq c^{(\text{curr})} \leq 2^{w-2}. \tag{7}$$

We now use boundaries of $u_s^{(\text{curr})}$ and $c^{(\text{curr})}$ in equations (6) and (7) to bound the value of $t_{-1}^{(\text{next})}$:

$$0 \leq t_{-1}^{(\text{next})} < 2^w.$$

We then have

$$\begin{aligned} t_{-1}^{(\text{next})} &= \left(u_s^{(\text{curr})} + c^{(\text{curr})} \right) \bmod 2^w \\ &= u_s^{(\text{curr})} + c^{(\text{curr})} \\ &= t_{s-1}^{(\text{curr})} \end{aligned}$$

Then, from Algorithm 1, the result of MMM is

$$\begin{aligned} T^{(\text{curr})} &= t_{-1}^{(\text{next})} 2^{(s-1)w} + \sum_{j=0}^{s-2} t_j^{(\text{curr})} 2^{jw} \\ &= \sum_{j=0}^{s-1} t_j^{(\text{curr})} 2^{jw} \end{aligned}$$

and is lesser than 2^m, as $0 \leq t_j < 2^w$, $\forall j \in [0, s-1]$. Results for independent MMMs in successive LMs then do not overlap, and are the same as in original algorithm. □

REFERENCES

[1] G. Gallin and A. Tisserand, "Generation of hyper-threaded GF(P) multipliers for flexible curve based cryptography on FPGA," *submitted article*.

[2] P. L. Montgomery, "Modular multiplication without trial division," *Math. of Comp.*, vol. 44, pp. 519–521, Apr. 1985.

[3] C. D. Walter, "Montgomery exponentiation needs no final subtractions," *Electronics Letters*, vol. 35, pp. 1831–1832, Oct. 1999.