

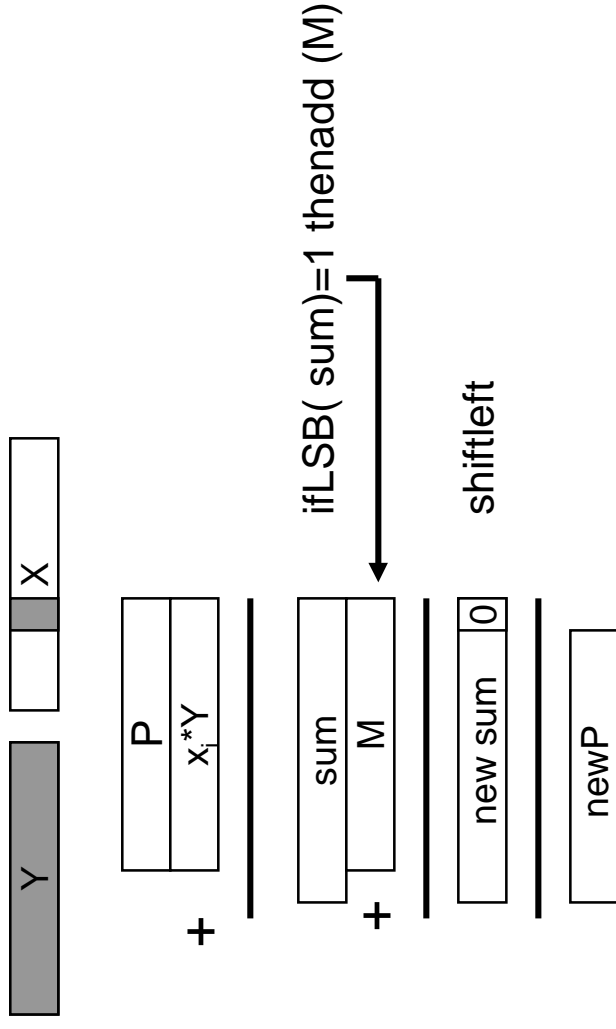
# Area-Time-Efficient Montgomery Modular Multiplication

Manfred Schimmler, Viktor Bunimov, Boris Tolg  
Institute for Computer Engineering and Communication Networks,  
Technical University of Braunschweig, Germany

# Modular Multiplication

Montgomery Multiplication:

- Same Idea as Interleaved Modular Multiplication :
- Reduce length of intermediate result
- Avoid Comparison with  $M$  by operating LSB-first



# Montgomery Multiplication

YX

$$\begin{array}{r}
 \underline{0111} * \underline{1011} \\
 0000 \\
 \underline{0111} \\
 0111 P_0=1? \\
 \underline{1101} \\
 10100 \text{ div}2 \\
 \underline{0111} \\
 10001 P_0=1? \\
 \underline{1101} \\
 11110 \text{ div}2 \\
 0000 \\
 \underline{1111} P_0=1? \\
 \underline{1101} \\
 11100 \text{ div}2 \\
 \underline{0111} \\
 10101 P_0=1? \\
 \underline{1101} \\
 100010 \text{ div}2 >M? \\
 \underline{-1101} \\
 \underline{X*Y*16^{-1} \bmod M = 0100}
 \end{array}$$

```

P = 0
For i = 0 to n-1 do
    P = P + x_i*Y
    if P_0=1 then P = P + M
    P = P div 2
If P > M then P = P - M
Result = P

```

# Montgomery Multiplication

Needs precomputing of  $16^2 \bmod M$   
and a second pass through the same  
algorithm

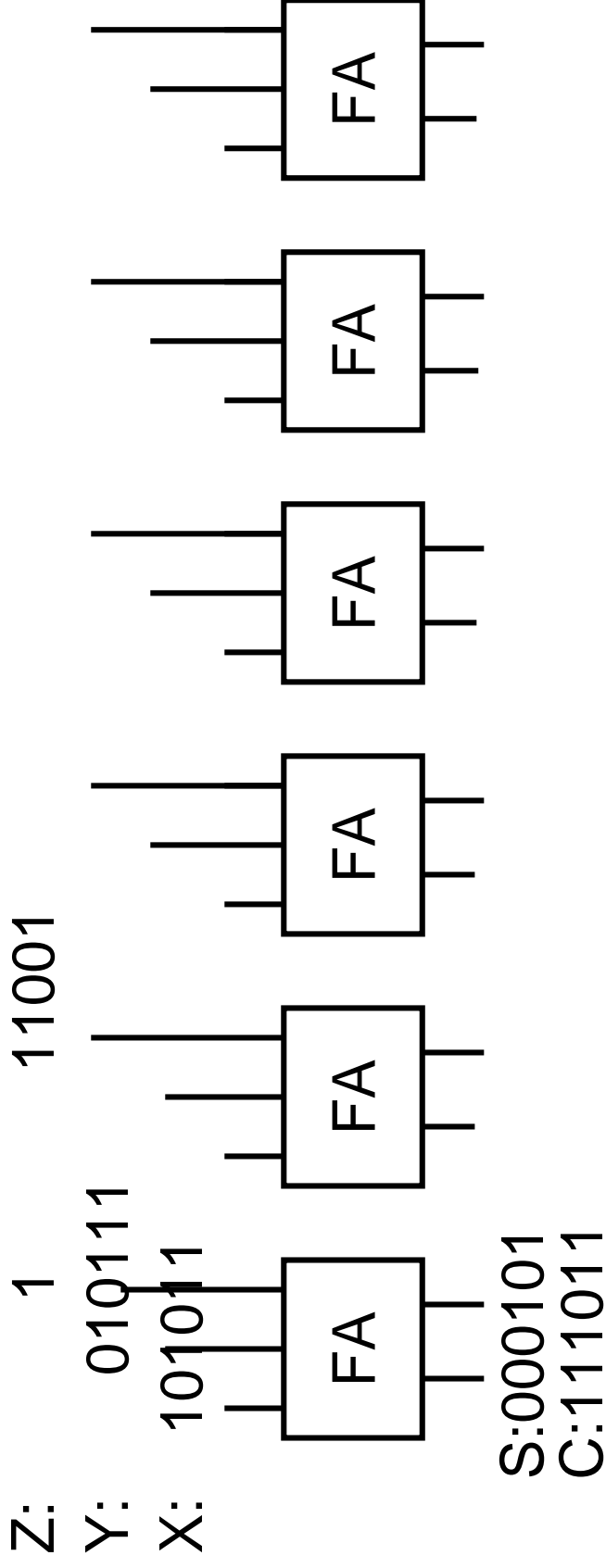
$$\text{Temp} = X * Y * 16^{-1} \bmod M = 0$$

$$\begin{aligned} P &= \text{Temp} * 16^2 * 16^{-1} \bmod M \\ &= X * Y \bmod M \end{aligned}$$

# Carry Save Addition

3 Operands X, Y, Z      2 Results S, C

$$S + 2 * C = X + Y + Z$$



# Carry Save Addition

3 Operands X, Y, Z      2 Results S, C

$$S + 2 \cdot C = X + Y + Z$$

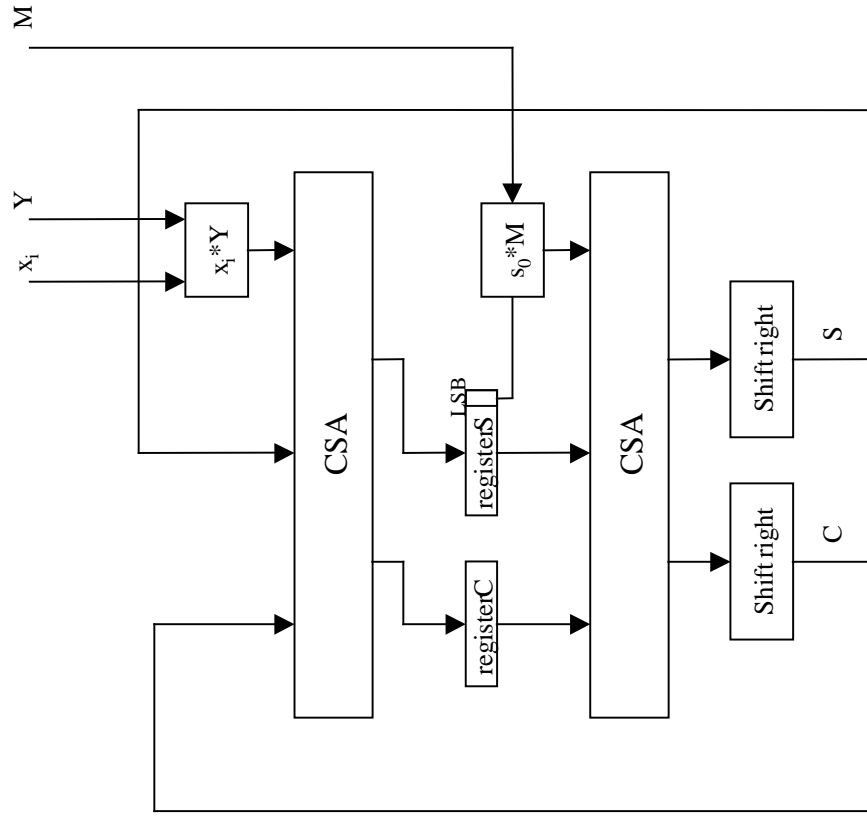
Advantage:      no carry propagation  
                 addition in  $O(1)$  time

Disadvantage:      difficult to compare results  
                 even difficult to compare to 0

# Algorithm for fast Montgomery Multiplication with using Carry Save Addition

- Inputs:  $X, Y, M$  with  $0 \leq X, Y < M$
- Output:  $P = (X \times Y \times (2^n)^{-1}) \bmod M$
- $n$ : number of bits in  $X$ ,
- $x_i$ :  $i$ th bit of  $X$
- $s_0$ : LSB of  $S$
  
- 1.  $S := 0; C := 0;$
- 2. **for**  $i := 0$  **to**  $k-1$  **do**
- 3.  $S, C := S + C + x_i * Y;$
- 4.  $S, C := S + C + s_0 * M;$
- 5.  $S := S \text{div} 2; C := C \text{div} 2;$
- 6.  $P := S + C$
- 7. **if**  $P \geq M$  **then**  $P := P - M;$

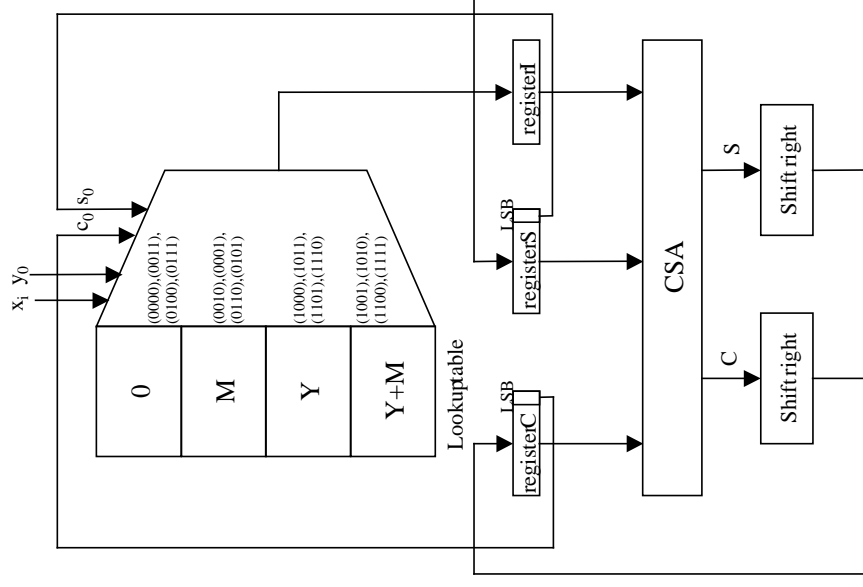
# Implementation of Algorithm for fast Montgomery Multiplication



# Algorithm for new Montgomery Multiplication with using Carry Save Addition

- Inputs:  $X, Y, M$  with  $0 \leq X, Y < M$
- Output:  $P = (X \times Y \times (2^n)^{-1}) \bmod M$
- $n$ : number of bits in  $X$ ,
- $x_i$ :  $i$ th bit of  $X$
- $s_0$ : LSB of  $S$ ,  $c_0$ : LSB of  $C$ ,  $y_0$ : LSB of  $Y$
- $R$ : precomputed value of  $Y + M$
  
- 1.  $S := 0$ ;  $C := 0$ ;
- 2. **for**  $i := 0$  **to**  $k-1$  **do**
- 3. **if**  $(s_0 = c_0)$  and not  $x_0$  **then**  $i := 0$ ;
- 4. **if**  $(s_0 \neq c_0)$  and not  $x_0$  **then**  $i := M$ ;
- 5. **if** not  $(s_0 \oplus c_0 \oplus y_0)$  and  $x_0$  **then**  $i := Y$ ;
- 6. **if**  $(s_0 \oplus c_0 \oplus y_0)$  and  $x_0$  **then**  $i := R$ ;
- 7.  $S, C := S + C + i$ ;
- 8.  $S := S \text{div} 2$ ;  $C := C \text{div} 2$ ;
- 9.  $P := S + C$
- 10. **if**  $P \geq M$  **then**  $P := P - M$ ;

# Implementation of Algorithm for new Montgomery Multiplication



# Complexity

Montgomery fast Version:

Area:  $2n$ -Bit- Adders

Time:  $2^*n$ loops ,each of time  $2$

AT:  $8^*n^2$

Montgomery new Version:

Area:  $1n$ -Bit- Adders

Time:  $n$  loops,each of time  $1$

AT:  $2^*n^2$

# Conclusion

The new Method is

- +time efficient
- +area efficient
- +energy efficient
- +simple

++superior to the method currently used !

## Future work

Improving the new method by higher -radix-techniques

Modular squaring

Modular exponentiation

Implementation of applications (like RSA)