# Comparison of Different Implementations of Montgomery Modular Inverse in Hardware

Jiří Buček, Róbert Lórencz

{bucekj | lorencz}@fel.cvut.cz

Czech Technical University in Prague, FEE

Department of Computer Science and Engineering

# MMI – Montgomery Modular Inverse

- Montgomery Multiplicative Inverse modulo $p$ (MMI)
- Often computed with Extended Euclidean Algorithm
- $\mathrm{MMI}(a) = a^{-1}\, 2^{2n} \bmod p$ $\qquad n = \lceil \log_2 p \rceil$
  - $b_{\mathrm{M}} = b\, 2^n \bmod p \rightarrow$ Montgomery domain (MD)
- Phase I – $\mathrm{AMI}(a) = a^{-1}\, 2^k \bmod p$, $n \le k \le 2n$
  - AMI – Almost Montgomery Inverse
  - $k \approx 1.4n$
- Phase II – $\mathrm{MMI}(a) = \mathrm{AMI}(a)\, 2^{2n-k} \bmod p$
- Applications: Cryptography
  - EC: $n =$ 160, 192…

# MMI – Algorithm Description

**Phase I**

$u \leftarrow p$, $v \leftarrow a$, $r \leftarrow 0$, $s \leftarrow 1$, $k \leftarrow 0$
while (1)
    if ($u_{LSB} == 0$) then $u \leftarrow u/2$, $s \leftarrow 2s$
    else if ($v_{LSB} == 0$) then $v \leftarrow v/2$, $r \leftarrow 2r$
    else  $x = u - v$
        if ($x == 0$) then break
        if ($u > v$) then $u \leftarrow (u - v)/2$, $r \leftarrow r + s$, $s \leftarrow 2s$
        else $v \leftarrow (v - u)/2$, $s \leftarrow r + s$, $r \leftarrow 2r$
    $k \leftarrow k + 1$

**Phase II**

while ($k \neq 2n$)
    $x = 2s – p$
    if ($x \geq 0$) then $s \leftarrow x$
    else $s \leftarrow 2s$
    $k \leftarrow k + 1$
return $s$

# MMI – Algorithm Overview (Phase I, AMI)

- Traditional

if $(u_{LSB} == 0)$ then $u \leftarrow u/2$, $s \leftarrow 2s$

else if $(v_{LSB} == 0)$ then $u \leftarrow u/2$, $r \leftarrow 2r$

2 steps

else if $(u > v)$ then $u \leftarrow (u - v)/2$, $r \leftarrow r + s$, …

else $v \leftarrow (v - u)/2$, …

- Subtraction-Free

if $(u_{LSB} == 0)$ then $u \leftarrow u/2$, $s \leftarrow 2s$

else if $(v_{LSB} == 0)$ then $u \leftarrow u/2$, $s \leftarrow 2s$

1 step

else $x = u + v$

if $(C(x) == 0)$ then $u \leftarrow x/2$, $r \leftarrow r + s$, …

else $v \leftarrow x/2$, …

# Traditional AMI – with Subtractions

- Keep always $u$ and $v$ positive $\Rightarrow$
- Need to compute $u - v$ (≈ $0.7n$)
  - Sometimes also $v - u$ (≈ $0.35n$)

1. At the same time: 2 subtractors $\Rightarrow$
   - Extra space
2. Sequentially: 1 subtractor $\Rightarrow$
   - Extra time $\Rightarrow$ slowdown > 1.25 (cycles)

- In each case, 1 more adder for $r + s$ (slave part)
- Together: 2 adders (2ADD) or 3 adders (3ADD)

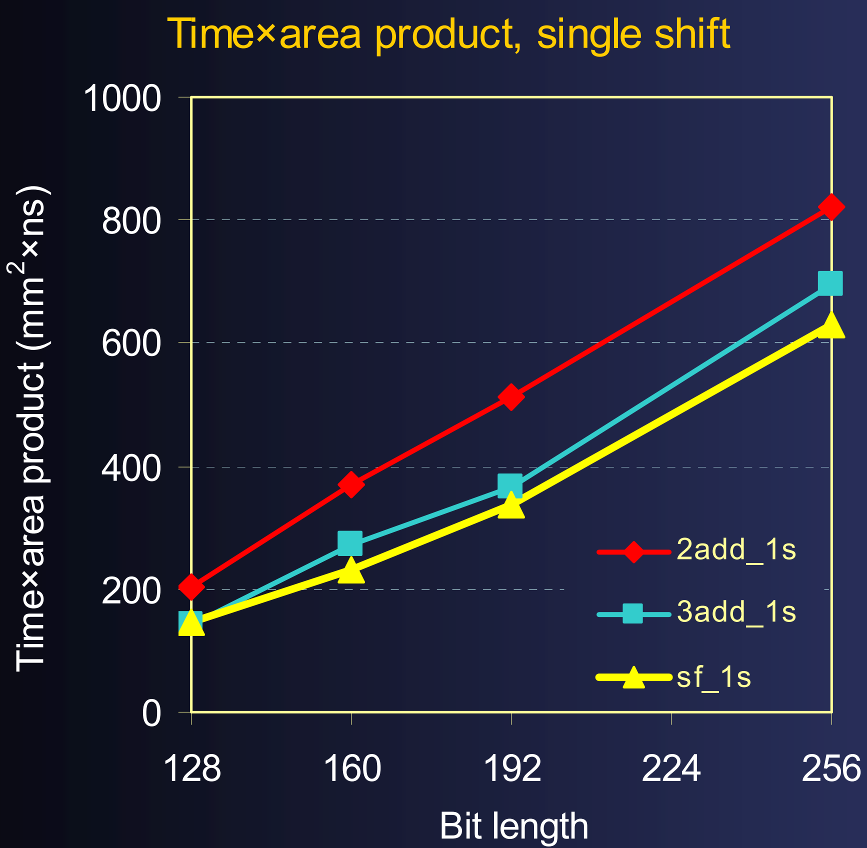
# Subtraction-free AMI (SF AMI)

- Always $u$ is negative & $v$ positive ( 2's complement)
- After evaluation $x = u + v$
  - if (carry-out = 0) $\Rightarrow u \leftarrow x/2$; else $v \leftarrow x/2$
- No need for subtractors, 1 adder, no more bus bits
- No slowdown $\rightarrow$ no redundant decision step
- Again: $r + s$ computed using an adder in slave part
- Together – SF contains 2 adders
- Both traditional and SF AMI – Possible to exploit 2 shifts when LSB and LSB – 1 are “00”

# Phase II – multiplication by $2^{2n-k}$ mod $p$

- Interleaved multiplication and reduction
- Multiplication by 2 (1 shift) $\Rightarrow$ ≈ 0.6$n$ steps
  - Reduction by $p$ or 0 $\Rightarrow$ 1 subtraction (test)
    - Suitable for 2ADD, 3ADD, SF (needs $-p$)
- Multiplication by 4 (2 shifts) $\Rightarrow$ ≈ 0.3$n$ steps
  - Reduction by 3$p$, 2$p$, $p$ or 0 $\Rightarrow$ 3 subtractions
    - Reduced value $s$: $s - 3p$, $s - 2p$, $s - p$
    - Suitable for 3ADD – however –
  - Prediction of Reduced Value (PRV): 3p/2p/p/0 $\Rightarrow$ only 2 subtractions
    - Reduced value $s$: ($s - 3p$, $s - 2p$) or ($s - 2p$, $s - p$)
    - Suitable for SF (needs $-p$, $-2p$, $-3p$)

# Results – ASIC single shifting



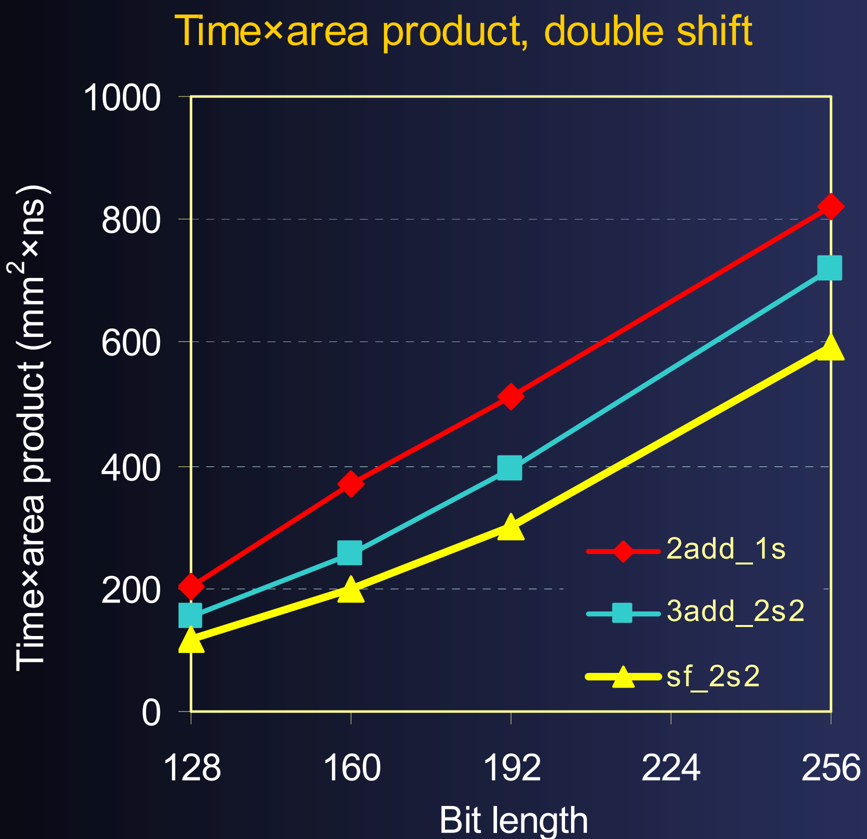

| Relative computation time | | | |
|---|---|---|---|
| n (bits) | 2add_1s | 3add_1s | sf_1s |
| 128 | 1,00 | 0,73 | 0,74 |
| 160 | 1,00 | 0,72 | 0,76 |
| 192 | 1,00 | 0,72 | 0,72 |
| 256 | 1,00 | 0,74 | 0,84 |

Subtraction-free (SF) Inverse:

- SF is equally fast as 3ADD
- SF area ~12% smaller than 3ADD
- SF & 3ADD ~1.3×faster than 2ADD

- One shift per clock cycle (2ADD has 0,35*n* wasted cycles)
- Implemented in VHDL, using Synplify ASIC, TSMC 0.18u

# Results – ASIC double shifting




| Relative computation time | | | |
|---|---|---|---|
| n (bits) | 2add_1s | 3add_2s2 | sf_2s2 |
| 128 | 1,00 | 0,49 | 0,50 |
| 160 | 1,00 | 0,52 | 0,51 |
| 192 | 1,00 | 0,52 | 0,48 |
| 256 | 1,00 | 0,57 | 0,56 |

Subtraction-free (SF) Inverse:

- SF is equally fast as 3ADD
- SF area ~20% smaller than 3ADD
- SF & 3ADD 2×faster than 2ADD

- Up to 2 shifts per clock in Phase I, 2 shifts per clock in Phase II
- Still max. 1 shift per clock in 2ADD

# Conclusion

SF-MMI with 2 shifts vs. other MMI:

- Equally small, but faster than 2ADD-MMI
- Equally fast, but smaller than 3ADD-MMI
- Feasible implementation in ASIC
- Improvement in time x area product
  - ASIC in average 20% over 3ADD with 2 shifts
- Suitable for no long word cryptography applications such as ECC

# Thank you for your attention

# Appendix – 3ADD MMI algorithm

**Phase I**

$u \leftarrow p$, $v \leftarrow a$, $r \leftarrow 0$, $s \leftarrow 1$, $k \leftarrow 0$

while (1)

    if ($u_{LSB} == 0$) then $u \leftarrow u/2$, $s \leftarrow 2s$

    else if ($v_{LSB} == 0$) then $v \leftarrow v/2$, $r \leftarrow 2r$

    else $x = u - v$

        if ($x == 0$) then break

        if ($C(x) == 1$) then $u \leftarrow x/2$, $r \leftarrow r + s$, $s \leftarrow 2s$

        else $v \leftarrow (v - u)/2$, $s \leftarrow r + s$, $r \leftarrow 2r$

    $k \leftarrow k + 1$

**Phase II**

while ($k \neq 2n$)

    $x = 2s - p$

    if ($x > 0$) then $s \leftarrow x$

    else $s \leftarrow 2s$

    $k \leftarrow k + 1$

return $s$

# Appendix – SF MMI algorithm

**Phase I**

$u \leftarrow -p$, $v \leftarrow a$, $r \leftarrow 0$, s $\leftarrow 1$, $k \leftarrow 0$
while (1)
    if ($u_{\mathrm{LSB}}$ == 0) then $u \leftarrow u/2$, $s \leftarrow 2s$
    else if ($v_{\mathrm{LSB}}$ == 0) then $v \leftarrow v/2$, $r \leftarrow 2r$
    else $x = u + v$
        if ($x$ == 0) then break
        if (C($x$) == 0) then $u \leftarrow x/2$, $r \leftarrow r + s$, $s \leftarrow 2s$
        else $v \leftarrow x/2$, $r \leftarrow r + s$, $r \leftarrow 2r$
    $k \leftarrow k + 1$

**Phase II**

while (k $\neq 2n$)
    $x = 2s + (-p)$
    if (C($x$) == 1) then $s \leftarrow x$
    else $s \leftarrow 2s$
    $k \leftarrow k + 1$
return $s$

# Appendix – SF MMI with 2 bit shifting

**Phase I**

$u \leftarrow -p,\ v \leftarrow a,\ r \leftarrow 0,\ s \leftarrow 1,\ k \leftarrow 0$

while (1)

if ($u_{1,0}$ == "00") then $u \leftarrow u/4,\ s \leftarrow 4s,\ k \leftarrow k + 2$

else if ($u_0$ == 0) then $u \leftarrow u/2,\ s \leftarrow 2s,\ k \leftarrow k + 1$

else if ($v_{1,0}$ == "00") then $v \leftarrow v/4,\ r \leftarrow 4r,\ k \leftarrow k + 2$

else if ($v_0$ == 0) then $v \leftarrow v/2,\ r \leftarrow 2r,\ k \leftarrow k + 1$

else $x = u + v,\ z = r + s$

if ($x$ == 0) then break

if (C($x$) == 1) then

if ($x_1$ == 0) then

$u \leftarrow x/4,\ r \leftarrow z,\ s \leftarrow 4s,\ k \leftarrow k + 2$

else $u \leftarrow x/2,\ r \leftarrow z,\ s \leftarrow 2s,\ k \leftarrow k + 1$

else if ($x_1$ == 0) then

$v \leftarrow x/4,\ s \leftarrow z,\ r \leftarrow 4r,\ k \leftarrow k + 2$

else $v \leftarrow x/2,\ s \leftarrow z,\ r \leftarrow 2r,\ k \leftarrow k + 1$

# Appendix – SF MMI with 2 bit shifting

**Phase II**

if ($k_0$ == 1) then
    $x = 2s + (-p)$
    if ($C(x) == 1$) then $s \leftarrow x$
    else $s \leftarrow 2s$
    $k \leftarrow k + 1$
while ($k \neq 2n$)
    if ($PRV_0(s, p) == 0$) then $x = 4s$
    else $x = 4s + (-2p)$
    if ($PRV_1(s, p) == 0$) then $z = 4s + (-p)$
    else $z = 4s + (-3p)$
    if ($z < 0$) then $s \leftarrow x$
    else if($x < 0$) then $s \leftarrow z$
    else if($PRV_0(s, p) == 0$) then $s \leftarrow z$
    else if($PRV_1(s, p) == 0$) then $s \leftarrow x$
    else $s \leftarrow z$
    $k \leftarrow k + 2$
return $s$