Comparing Left-shift and Montgomery inverse implementations in ASIC

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
- MMI(a) = $a^{-1}\ 2^{2n} \bmod p$ $\qquad n = \lceil log_2 p \rceil$
  - $b_M = b\ 2^n \bmod p \rightarrow$ Montgomery domain (MD)
- Phase I – AMI(a) = $a^{-1}\ 2^k \bmod p$, $n \le k \le 2n$
  - AMI – Almost Montgomery Inverse
  - $k \approx 1.4n$ steps, of that $\approx 0.7n$ subtract+shift
- Phase II – MMI(a) = AMI(a) $2^{2n-k} \bmod p$

LSI – Left Shift Inverse

- Multiplicative Inverse modulo *p* (LSI)
- Modified Extended Euclidean Algorithm
- LSI(a) = a^{-1} mod p

$$n = \lceil log_2 p \rceil$$

  - New algorithm for Classical modular inverse (CHES 2002)
- Integer Domain
- Single phase (outputs inverse directly)
- Application: e.g. Elliptic Curve Cryptography, smart cards
- Not implemented in HW (ASIC) until now

Difference between MMI and LSI

- MMI uses MD, needs conversion – overhead
- MMI has two phases vs LSI one phase
- MMI 2. phase performs deferred halvings, this means more shifts and +/- operations
- LSI computes efficiently MI in ID avoiding conversion
- Left-shifting approach uses 2's complementary code
  - This allows to work with negative integers and easily choose operations +/- in computing MI
  - Control flow does not depend on tests based on subtraction of operands

Operational complexity comparison

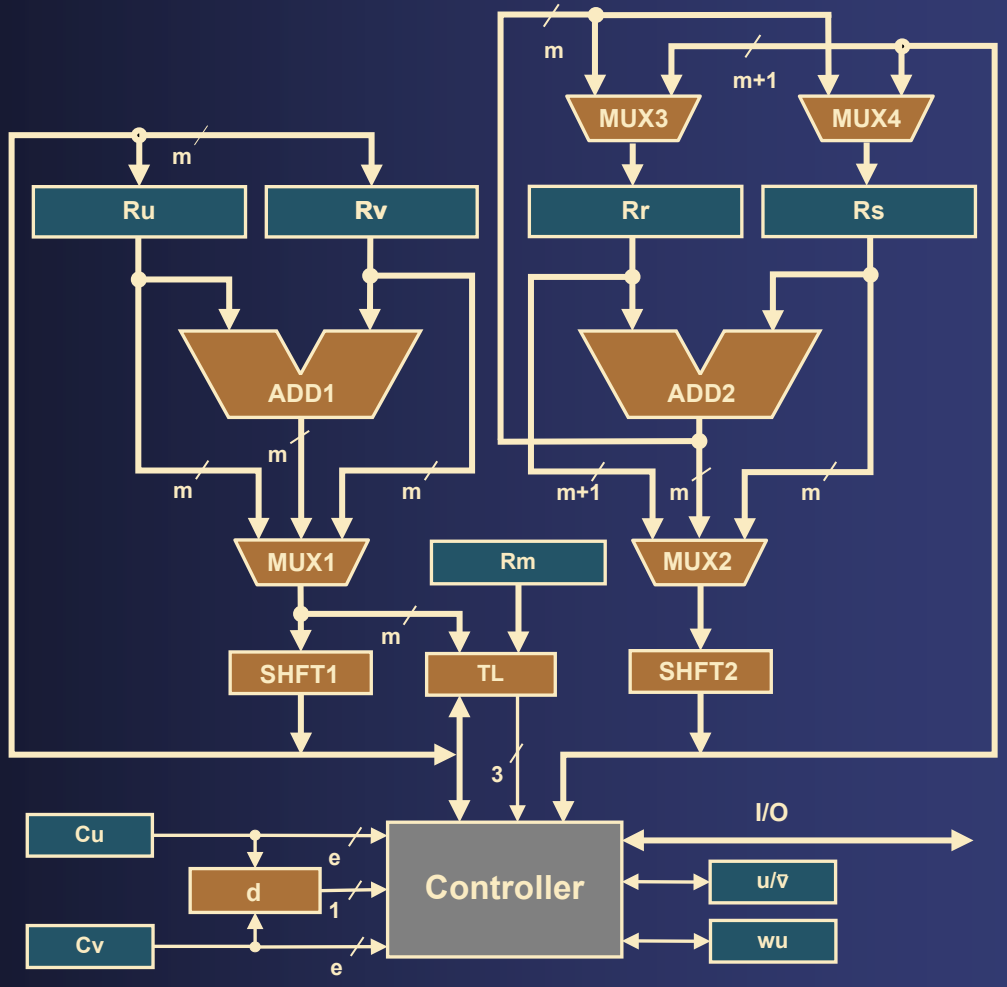
- LSI contains less operations than MMI
- MMI assuming operands aligned to LSB
- LSI assuming operands aligned to MSB
- Only data path complexity considered here

Operation count	MMI	LSI
sub+shift	0.7*n*	0.7*n*
shift only	0.7*n*	0.7*n*
test overhead	0.35*n*	**0**
second phase	0.6*n* Koç 0.4*n* Kaliski	**0**

Motivation for ASIC implementation

- Advantage of LSI – lower number of additions / subtractions
- Implementation in FPGA – dedicated adder structures do not allow to exploit this advantage
- Idea – ASIC offers better implementation possibilities
- ASIC platform allows a complex and weighted comparison
- First known ASIC implementation

A circuit implementation of LSI



Results – TSMC 0.18u technology, Synopsys DC

- **LSI is in average 35% faster than MMI**
- LSI area is in average 33% larger than MMI
- ⇒ LSI is well suited for integer domain calculation
- Is it possible to lower the area?

	MMI	**LSI**	**Speedup**
n	time (us)	time (us)	t_{MMI}/t_{LSI}(-)
128	0.76	0.56	1.36
160	0.98	0.75	1.31
192	1.17	0.88	1.33
256	1.66	1.20	1.38

LSI – Is it possible to lower area and maintain speed?

- Low number of additions wrt Montgomery – smaller adder → **multi-cycle addition**
  - \+ Smaller area
  - \+ Shorter critical path
  - – Number of cycles of +/- operations grows
- Multi-cycle addition is possible because the result is not needed for control of the same iteration, unlike MMI
- Shifting is cheap – **single cycle shifting**

Multi-cycle architecture details

- Adder length is half operand length $\rightarrow$ 2 cycle addition / subtraction
- Bus width halved, except for shifting
- Multiplexers from adder narrowed, but with more inputs
- Number of multiplexers increased
- Controller more complex

Results of multi-cycle architecture

- **Multi-cycle LSI (MLSI) area is 26% less than LSI**
- MLSI is 29% slower than LSI
  - Clock period is shorter, but the increase in number of cycles has a stronger effect on overall speed
  - Controller is more complex
  - Narrower buses but more multiplexor inputs
  - Adder area lower but adder delay only slightly lower (log)

	LSI		MLSI	
n	time (us)	area (mm^2)	time (us)	area (mm^2)
128	0.56	0.254	0.78	0.189
160	0.75	0.298	1.01	0.245
192	0.88	0.361	1.28	0.288
256	1.20	0.470	1.71	0.387

Conclusion

- Left-shift inversion (LSI) is a viable alternative Montgomery inversion (MMI) for the integer domain
- LSI is significantly faster, but has larger area
- We expected LSI to perform better considering its lower operational complexity
- Experiments with multi-cycle addition performed. Results are not satisfying so far, we will continue to explore good algorithmic properties, i.e. low number of +/- operations and shifts.
- Future work – scalable architecture, pipelining, asynchronous operation