The Use of Finite Field GF(256) in the Performance Primitives Intel® IPP

Software & Service Group/
VCSD/CIP/IPP

Sergey Kirillov

Oct 7, 2008

Agenda

- Short IPP review
- GF(256) operations being in focus
- Methods for implementation operations over GF(256)
- IPP implementation of GF(256) operations
- Some of performance results

IPP. Short Review

Main metrics:

- **16** domains and **10000** functions
- high degree of **optimization**
- running on **wide range** Intel® **CPUs**
- **popular OS** (Windows, Linux, Mac OS)
- a lot of **samples** of usage
- developed **Q&A** infrastructure
- a lot of **customers** (40K licenses)

Factors of success:

- understanding of IA architectures
- architecture designer’s support, including Instruction Set enhancement

Technical feature:

most of IPP functions are targeted on single and double floating point and integer data types

IPP customers:

Samples of Applications over GF(2^8)

- protected version of AES
- RS decoder

	AES	RS decoder
main operation	multiplicative inversion $Z = X^{-1}$	polynomial value $P(X) = p_0+p_1.X+\ldots+p_{n-1}.X^{n-1}$
basic operation	GF(2^8) multiplications $Z = X.Y$, $Z = X.C$	GF(2^8) multiplication $Z = X.Y$

Operation Specific – vector Mode

AES:

$$\left.\begin{array}{r}\text{SubBytes()}\\ \text{ShiftRows()}\\ \text{MixColumns()}\\ \text{AddRoundKey()}\end{array}\right\} F_i(b_j),\ j=0,..,15$$

RS decoder:

$$\left.\begin{array}{r}\text{syndromes error}\\ \text{locations error}\\ \text{values}\end{array}\right\} P_i(x_j),\ x_j \text{ – fixed values}$$

Additional Code Designer’s Problem:

no instruction level support for GF multiplication

Implementation of GF(2^8) Operations

1) **Table Lookup**. Any necessary table (**mulTbl**, **logTbl**, **alogTbl**, **invTbl** and etc.) can be pre-computed in advance and used in application.

multiplication:

z=x·y ⇔ mulTBL[x][y] or

z=x·y ⇔ alogTbl[(logTbl[x]+logTbl[y]) mod 255]

inversion:

$z=x^{-1}$ ⇔ invTbl[x]

2) **Direct Computation**. Streaming SIMD Extension2 (SSE2) offers enough for direct GF(2^8) multiplication. The **performance is comparable** with Lookup Method. But using SSE2-based multiplication for **inversion** looks **inefficient**.

elementary operation $r = r \cdot t + a * b_i$

```
GFMUL_BIT r,a,b,mod,tmp
   PXTIME r,mod,tmp
   PMASK  tmp,b
   paddb  b,b
   pand   tmp,a
   pxor   r,tmp
```

```
PXTIME r,mod,tmp
   PMASK tmp,r
   paddb r,r
   pand  tmp,mod
   pxor  r,tmp
```

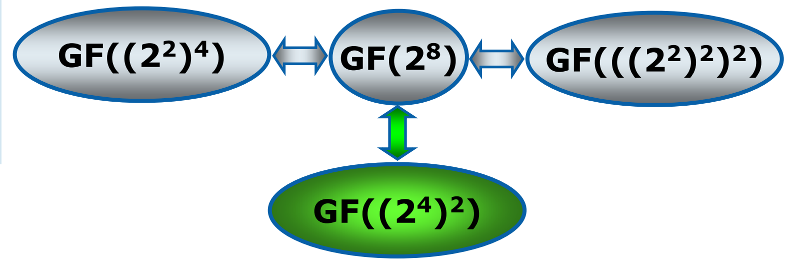
```
PMASK mask,data
   pxor    mask,mask
   pcmpgtb mask,data
```

3) **Composite Fields**. It's known that GF(2^{nm}) **is isomorphic** to GF($(2^n)^m$) but their arithmetic **complexity differ** and depends on choice of **n** and **m** and on constructing polynomials.

GF(2^8) Operations. Composite Field Approach

8=2*2*2=2*4=4*2

Our choice:	H: GF(2^8) ⇔ GF($(2^4)^2$)
Ground field:	GF(2^4), $q()=u^4+u+1$
Extension:	GF($(2^4)^2$), $p=t^2+t+L$



multiplication $z=x\cdot y$

$z_0=z_0\cdot y_0+x_1\cdot y_1\cdot L$
$z_1=x_0\cdot y_1+x_1\cdot(y_0+y_1)$

inversion $z=x^{-1}$

$z_0=(x_0+x_1)\cdot d^{-1}$
$z_1=x_1\cdot d^{-1}$
$d=x_0\cdot(x_0+x_1)+L\cdot x_1^{\,2}$

arbitrary L

{9,B,D,E}

where $H(x)=x_0+x_1t$, $H(y)=y_0+y_1t$, $H(z)=z_0+z_1t$
and x_i, y_i, z_i belongs GF(2^4)

Composite Field are frequently used by HW designers

Implementation of GF(2^4) Operations

Intel Core 2 Duo → SSSE3 → PSHUFB A,B ;; [$a_{b0}, a_{b1}, .., a_{b15}$]

```
PLOOKUP dst,src,TBL
        movdqa   dst, TBL
        pshufb   dst, src
```

multiplication

```
PLOOKUP   xmm2, xmm0, logTbl
PLOOKUP   xmm3, xmm1, logTbl
paddb     xmm2, xmm3
PLOOKUP   xmm3, xmm2, alogTbl
```

logTbl

C0	0	1	4	2	8	5	A	3	E	9	7	6	D	B	C

alogTbl

1	2	4	8	3	6	C	B	5	A	7	E	F	D	9	1

multiplication inversion

```
LOOKUP   xmm1, xmm0, invTbl
```

invTbl

0	1	9	E	D	B	7	6	F	2	C	5	A	4	3	8

const. multiplication

```
LOOKUP   xmm1, xmm0, mul9
```

(*9) mul9

0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
...	...	...	...	...	...	...	...	...	...	...	...	...	..		
0	9	1	8	2	B	3	A	4	D	5	C	6	F	7	E
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...

square

```
LOOKUP   xmm1, xmm0, sqrTbl
```

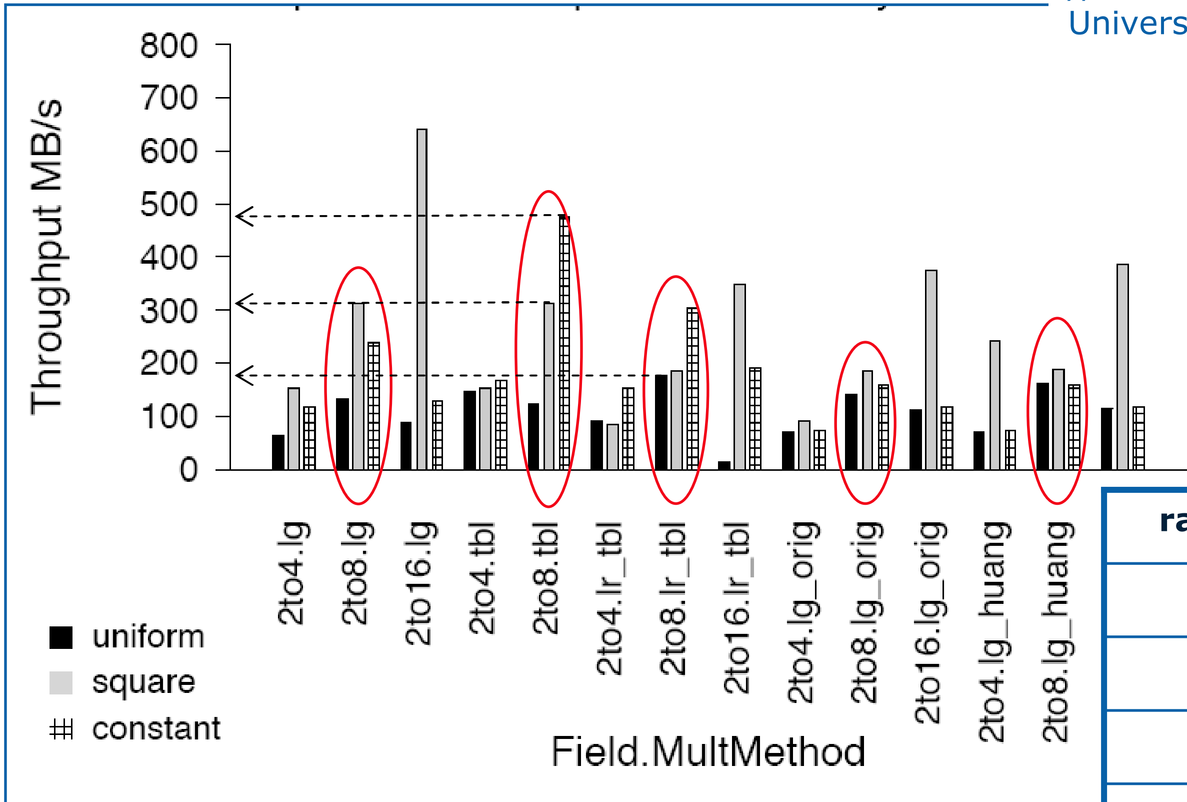
sqrTbl

0	1	4	5	3	2	7	6	C	D	8	9	F	E	B	A

Basic Operations Performance. IA

Analysis and Construction of Galois Fields for Efficient Storage Reliability. Tech. Report UCSC-SSRC-07-09

// Storage Systems Research Center & University of California, Aug, 2007



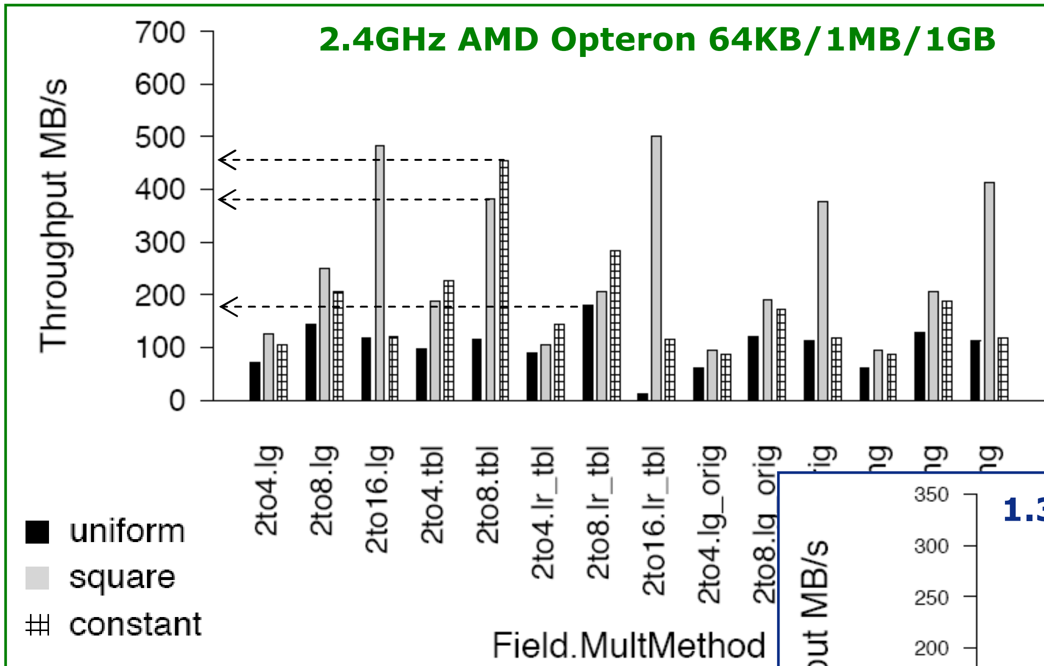
composite GF($(2^4)^2$)

rate (MB/s)	c/op	operation
1104	**1.81**	**uniform**
3333	**0.60**	**square**
4000	**0.50**	**constant**
1104	1.81	inversion
2040	**0.98**	**conversion**

~5 - ~10x faster

2GHz **Intel® Core™ 2 Duo** 32KB/2MB/2GB

Basic Operations Performance. AMD, PPC

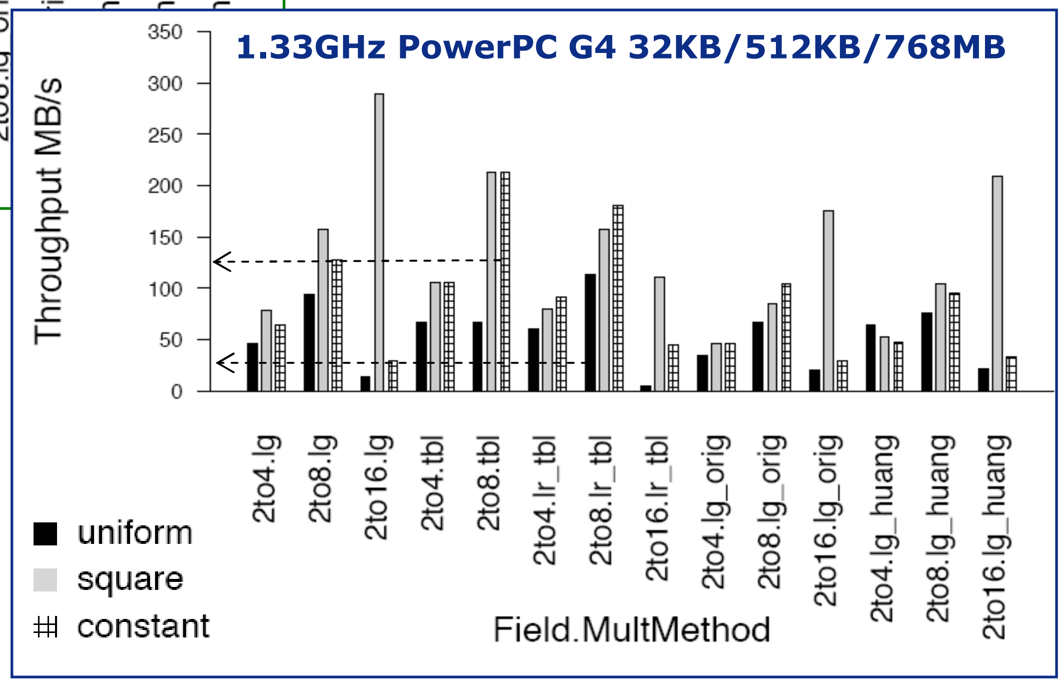


2009, SSE5, AMD:

pperm dst, src1,src2, ctr

// for (i=0; i<15; i++)

// dst[i] = ctr[i].op (src1|src2)ctr[i].sel



PPC G4:

vperm dst, src1,src2, ctr

// for (i=0; i<15; i++)

// dst[i] = (src1|src2)ctr[i].sel

AES & RS decoder Performance. Intel ® Core™ 2

AES (cycles/byte)

implementation	encode	decode
fast, but unprotected	17	17
protected: V1	**50**	**65**
protected: V2	**40**	**40**
protected: V3	**64**	**78**
protected: Composite Field	**28**	**28**

Ability for cache observation

V1: once per round

V2: once per few rounds

V3: multiple times per round

RS decoder (cycles/block)

2GHz **Intel® Core™ 2 Duo** 32KB/2MB/2GB

stage	IPP (tbl)	IPP (cf)	IPP(cf)	DSP StarCore™ SC140
syndromes	**164277**	**11646**	**5877**	**~6000**
error locator	**5580**	**2214**	**2772**	**~4000**
error locations	**74187**	**7416**	**4464**	**~4000**
error values	**10026**	**1314**	**729**	**~600**
Total	**254070**	**22590**	**15174**	**14600**

RS(255,223) **RS(255,239)**

Summary

• Considered the Lookup Table, Direct Computation and Composite Fields based implementations of $GF(2^8)$ operations.

• Compared performance of the implementations and made sure that the Composite Field approach is the best on Intel® Architecture as well as on PowerPC and AMD (non existed yet) microprocessors.

• Considered implementation of protected version of AES cipher and RS decoder based on Composite Field approach. Demonstrated their performance advantage over traditional approaches.

intel®

Leap ahead™