Efficient Galois Field Arithmetic on SIMD Architectures

R. Bhaskar[1], P. K. Dubey[2], V. Kumar[3] A. Rudra[4] and A. Sharma[5]

1: *INRIA*, 2: *Intel Research*, 3: *Amazon.com* 4: *UT Austin*, 5: *Fiorano Software*

Galois Field Arithmetic

- $GF(q)$
  - Add, Subtract, Multiply, Divide.
  - Carry-less operations.
  - Constant word length.
  - No rounding Issues.
- $GF(2^k)$
  - k-bit operands.
  - An *irreducible* polynomial of degree k.
- Composite Fields.

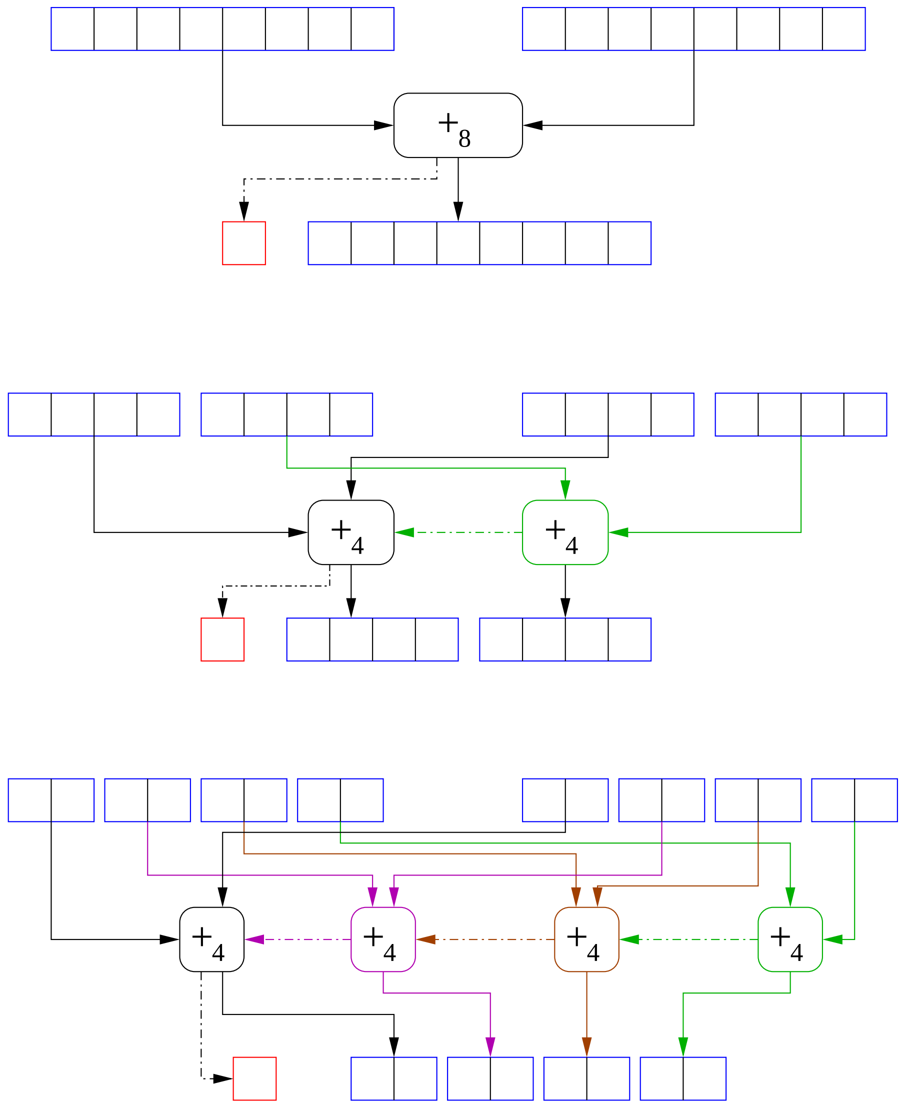
SIMD Architectures

- Data Parallelism
  - ◊ Image Processing.
  - ◊ Audio and Video Compression.
- Intel’s MMX, Sun’s VIS, AMD’s 3DNow!
- AltiVec
  - ◊ Extension to PowerPC.

Our Approach

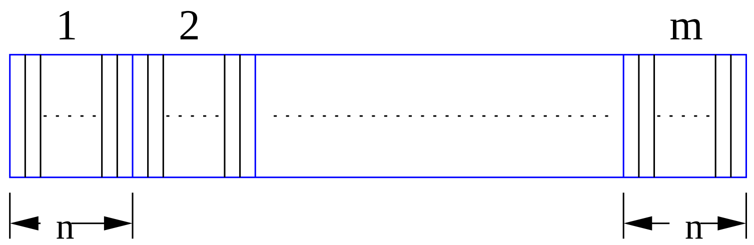
- Combines *data-slicing* and Composite Fields.
- *Algorithm* to generate “efficient” SIMD code.
- Susbstantial speedups
  - Reed-Solmon Error Correcting Codes.
  - Rijndael Encryption.
- Minimal Architectural Support.
  - Parallel TBL.
  - Bit-wise XOR/AND.
  - LOAD/STORE.

Adding Two 8-bit Integers



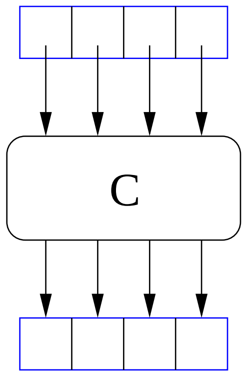
Composite Fields

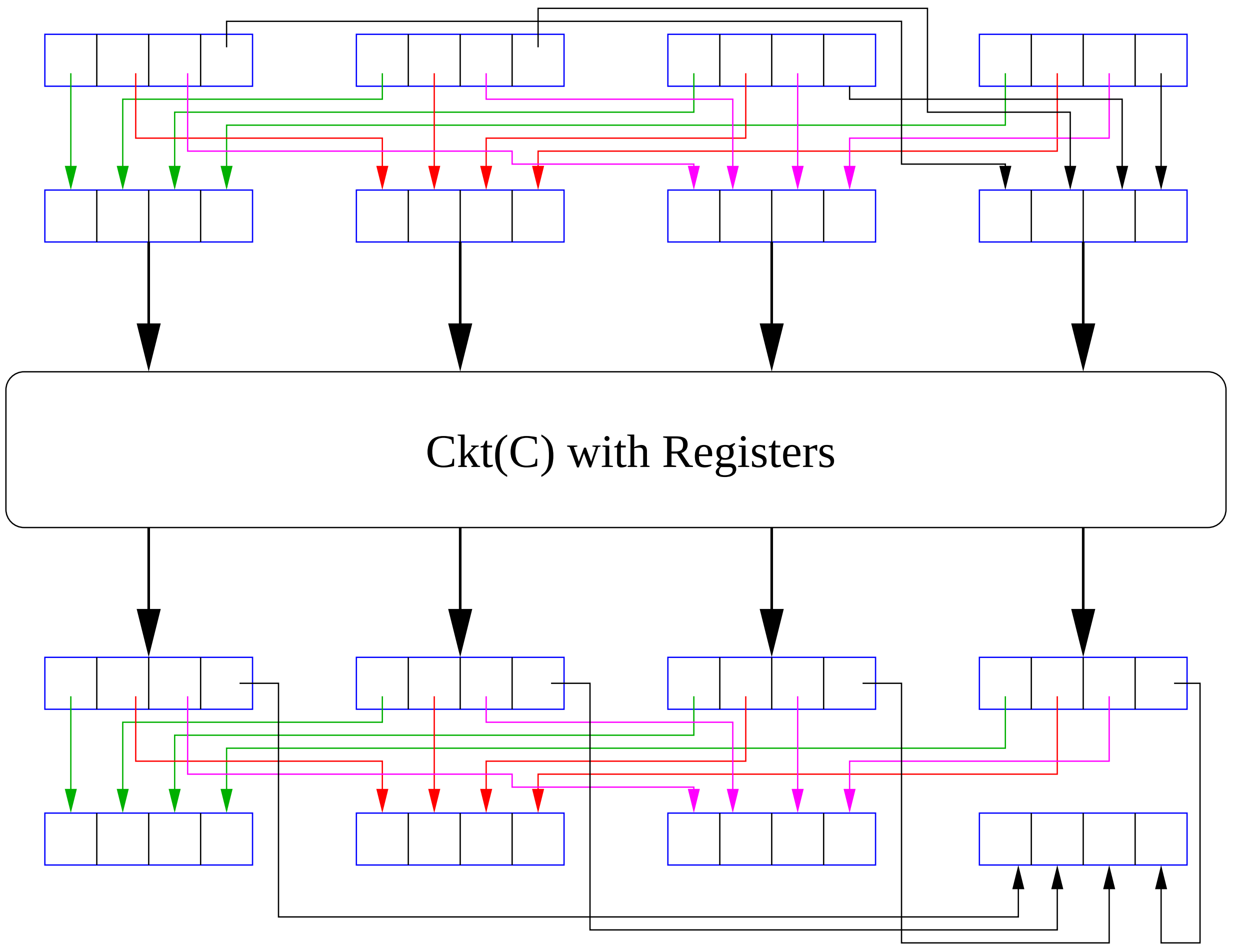
- $k = n \times m$



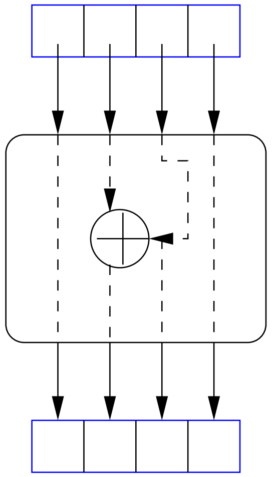
- "Alternate" parameters for $GF(2^k)$.
  - ◊ n, m such that $k = n \times m$.
  - ◊ Irreducible polynomial $Q(y)$ of degree n.
  - ◊ Irreducible polynomial $P(x)$ of degree m.

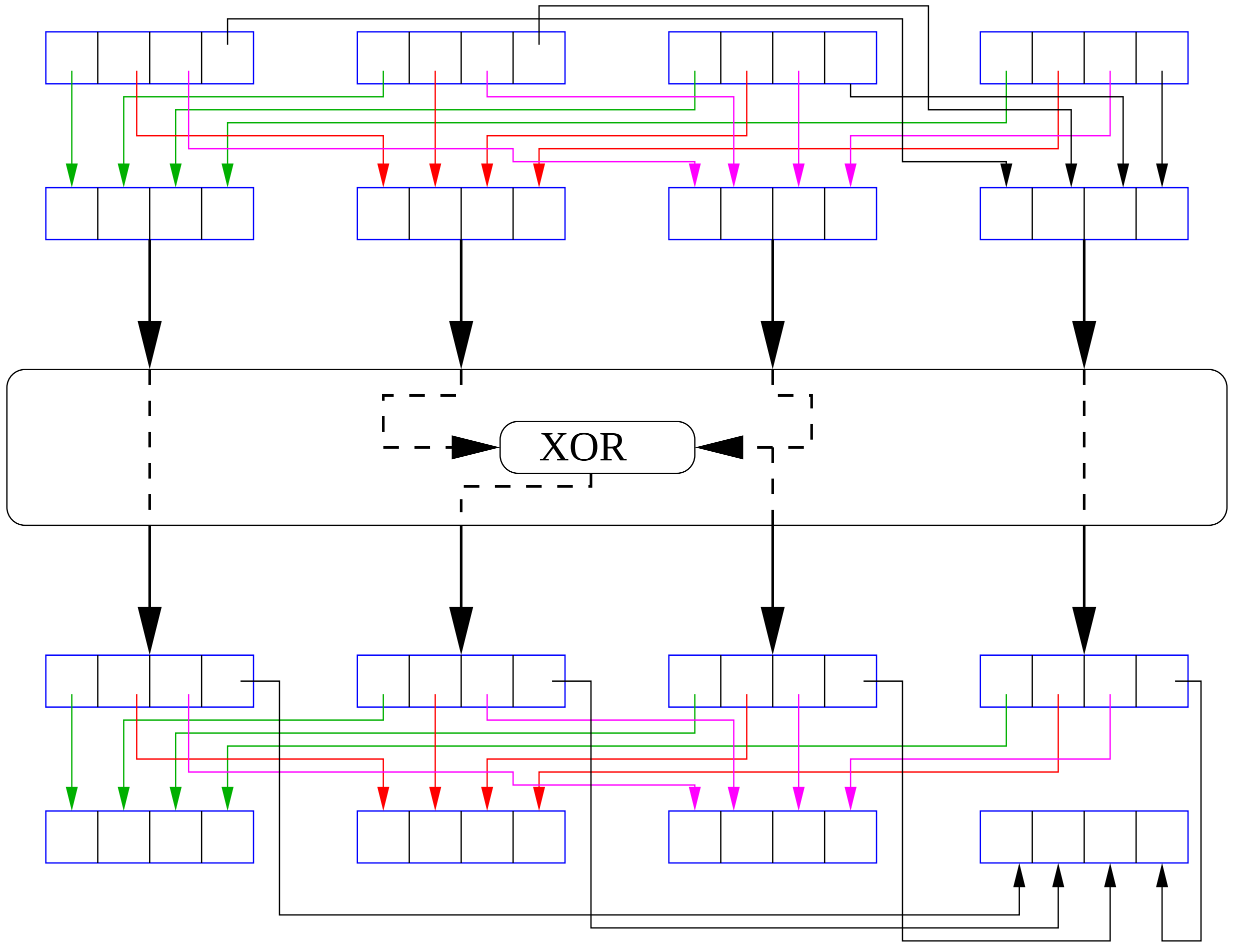
Bit Slicing





Why Bit Slicing Works





Combining Techniques

- Slicing allows for choice of composite fields.
- Gains
  - Cheaper operations in smaller fields.
  - Parallelism through data slicing.
- Overheads
  - Conversion between representations.
  - Data rearrangement in data-slicing.

The Algorithm

```
1. Set $d \leftarrow k, \mathcal{I} \leftarrow \emptyset, min_c \leftarrow \infty$.
2. If $k > W$ exit.
3. Repeat forever Steps 4 to 8:
4. While ($d$ does not divide $k$) set $d \leftarrow d - 1$.
5. If $d = 0$ exit.
6. $\mathcal{P} \leftarrow$ get_irr_polys_simple$(d - 1)$.
7. For all polynomials $P(x)$ in $\mathcal{P}$ do
     • $\mathcal{Q} \leftarrow$ get_irr_polys_cmplx $(\frac{k}{d} - 1, n, P(x))$.
     • For all polynomials $Q(y)$ in $\mathcal{Q}$ do
         ◊ $I_{P(x),Q(y),d} \leftarrow$ compile$(d, P(x), Q(y), \mathcal{S})$.
         ◊ if ($min_c >$ cost$(I_{P(x),Q(y),d})$) then
           $min_c \leftarrow$ cost$(I_{P(x),Q(y),d})$ and $\mathcal{I} \leftarrow I_{P(x),Q(y),d}$
8. Set $d \leftarrow d - 1$.
```

Reed-Solomon Performance

	RS(255,251)		RS(255,239)	
	Encoding	Decoding	Encoding	Decoding
Henrion (133 MHz Pentium)	45		NA	
4i2i.com (166 MHz Pentium)	12 (Encoding & Decoding)		2.72 (Encoding & Decoding)	
Byte-Sliced AltiVec Implementation (133 MHz PowerPC)	956	332	273	46

Throughput in Mbps

Rijndael Encryption Performance

	Cycles	Architecture	Comments
Worley et al	284	Pentium	Requires an 8KB table
	176	PA-RISC	
	124	IA-64	
Weiss et al.	210	Alpha 21264	
Wollinger et al.	228	TMS320C6x	
Aoki et al.	237	Pentium II	
Cryptomaniac	90	external co-processor	
AltiVec	230	1-way Altivec	Requires only a 256 byte table
	162	2-way Altivec	
Bit-sliced implementation	170	256b datapath width	Requires only EXOR, AND, L/S and 2KB table
	119	384b datapath width	
	100	512b datapath width	

Number of cycles for one block of encryption

Full Paper

http://www.cs.utexas.edu/users/atri/papers