Efficient Permutation Instructions for Fast Software Cryptography

Aditya Prasad

2/10/02

Introduction

- Want to perform permutations in software
- Current ISAs do not provide efficient bit-level software permutations

Motivations

- Facilitates more widespread use of
  - Secure information processing
  - Faster multimedia processing
- Current processors are word-oriented, so bit-level permutations are hard.

Secure Information Processing

- Authentication of users and host machines
- Confidentiality of messages sent over public networks
- Assurance that messages, programs, and data have not changed in transit

Secure Information Processing, cont'd

- Access control
- Provisions to ensure
  - privacy
  - anonymity
  - availability of essential services

Question

- "What general-purpose operations should this programmable processor incorporate so that it can execute cryptographic functions without significant performance degradation?"

Symmetric-key cryptography

- Break message into blocks and use
  - Confusion, to obscure relationship between plaintext and ciphertext
  - Diffusion, to spread redundancy of plaintext over ciphertext
- Used by DES (Data Encryption Standard), needs to be sped up

Quick Multimedia Processing

- Required for fast processing of multimedia instructions
- Many ISA extensions do not provide subword permutation instructions

New Permutation Instructions

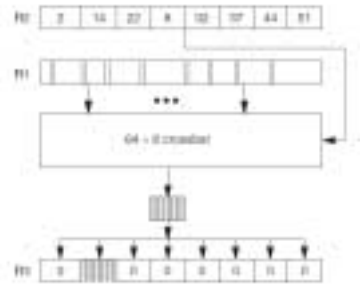
- Permuting n 1-bit elements, multi-bit elements in an n-bit word
- Previously, arbitrary n-bit permutations took O(n) time.
- Created four new methods: PPERM, GRP, CROSS, OMFLIP

Some math

- Number of n-bit permutations
  - $n! = O(n^n)$
  - $n! = \Omega(2^n)$
- Bits needed to specify one
  - $\lg(n!) = \theta[\, n \lg(n) \,]$
- Repetition allowed
  - $\lg(n^n) = n \lg(n)$

PPERM

- Explicitly specifies position from source for each bit in destination
- PPERM, x, Rs, Rc, Rd
  - x specifies contiguous bits in Rd
  - Rs contains bits to be permuted
  - Rc contains config. bits
  - Rd gets permuted result

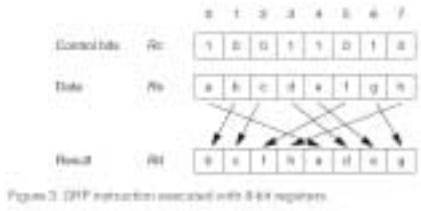


PPERM

- n bits in destination
- each requires lg(n) bits to determine source
- total n lg(n) bits
- Specify k bits per instruction
  - Need n/k = lg(n) instructions to do an n-bit permutation. For n = 64, k = 8 need 8 instructions

GRP

- GRP Rs, Rc, Rd
  - Rs is a source reg
  - Rc is a source reg
  - Rd is destination reg
- Sort the bits in Rs into left and right groups, according to bits in Rc

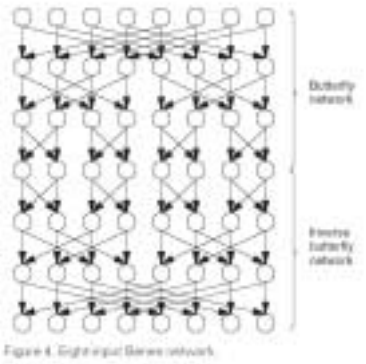


GRP

- n-bit registers
- Can do any n-bit perm. with lg(n) GRP instructions
  - Paper claims proof by construction
    - [Z. Shi and R. Lee, 2000]

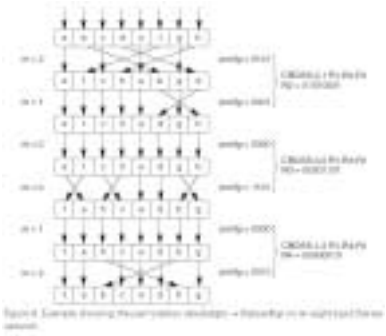
CROSS

- Based on the Benes network
  - Connecting two butterfly networks of the same size back-to-back
  - An n-input Benes network can be broken into 2 lg(n) stages, with lg(n) distinct stages
  - At each stage, every input has two outputs to the next stage



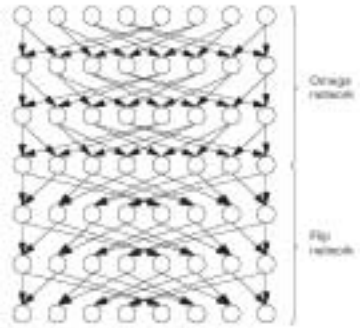
CROSS

- CROSS, m1, m2, Rs, Rc, Rd
  - Rs is source register
  - Rd is destination register
  - Rc is the configuration register
  - m1, m2 specify the basic ops
- Any perm. requires lg(n) CROSS instructions



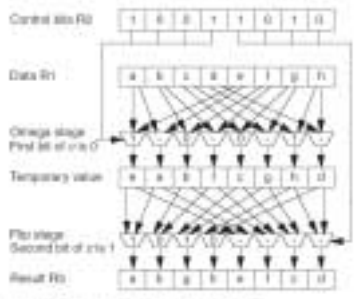
OMFLIP

- CROSS requires unit to have whole Benes network in hardware
- Instead, use Omega-flip network: an omega network followed by a flip network



OMFLIP

- OMFLIP, c, Rs, Rc, Rd
  - Rs is a source register
  - Rc is a source register
  - Rd is a source register
  - For each bit in c, 0 indicates omega operation, 1 indicates flip operation
- Requires lg(n) instructions, with less hardware



Hardware reqs

- PPERM requires a 64x8 crossbar network, and specialized shifter
- GRP requires a hierarchical gathering network with lg(n) stage bits.
- CROSS instruction requires a full Benes network
- OMFLIP requires the smallest area

[illegible]

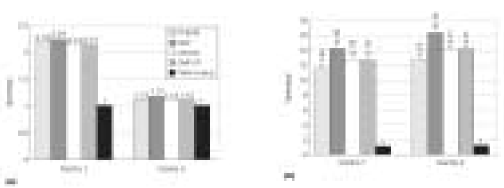
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]

[illegible]

[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]

Table 3. Memory storage requirement in 64-bit processors.

Characteristic	PPERM	GRP	CROSS or OMFLIP	Table lookup
[illegible]	64	40	40	[illegible]



[illegible]

[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]