

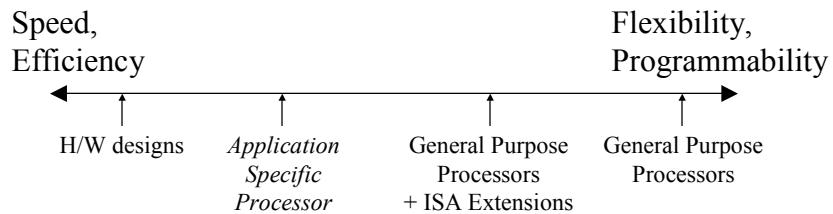
CryptoManiac

Slides borrowed with permission from
Todd Austin and Lisa Wu
University of Michigan
Advanced Computer Architecture Laboratory

Architecture's Diminishing Return

- Staples of value we strive for
 - High Speed
 - Low Power
 - Low Cost
- Tricks of the trade
 - Faster clock rates, via pipelining
 - Higher instruction throughput, via ILP extraction
- Strong evidence of diminishing return, PIII vs. P4
 - 22% less P4 inst throughput (0.35 vs. 0.45 SPECInt/MHz)
- Less return $\Rightarrow$ less value $\Rightarrow$ 

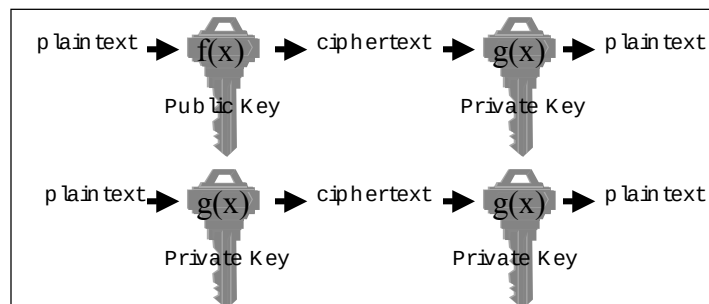
A Powerful Solution: Eschew Generality



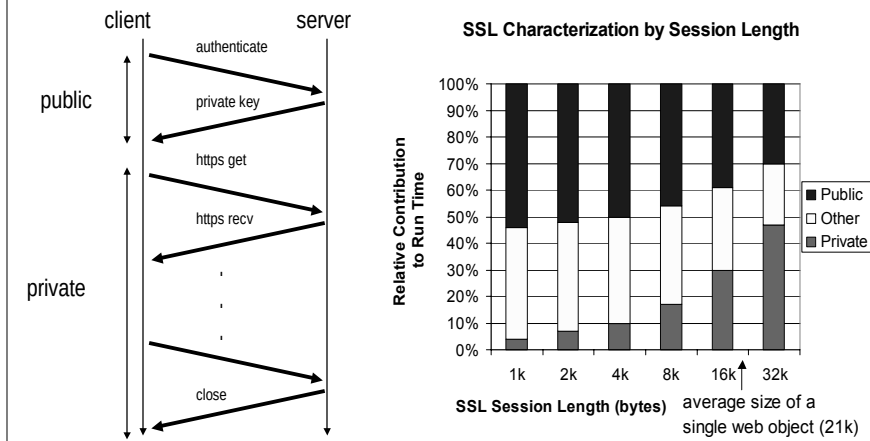
- ĩ Specialization limits the scope of a device's operation
 - ĩ Produces stronger properties and invariants
 - ĩ Results in higher return optimizations
 - ĩ Programmability preserves the flexibility regarded by GPPs
- ĩ A natural fit for embedded designs
 - ĩ Where application domains are more likely restrictive
 - ĩ Where cost and power are 1st order concerns

Cryptography

- ĩ Definitions:
 - ĩ *encryption* vs. *decryption*
 - ĩ *public-key cipher* vs. *secret-key cipher*
- ĩ *Public-secret key ciphers* are the most commonly used



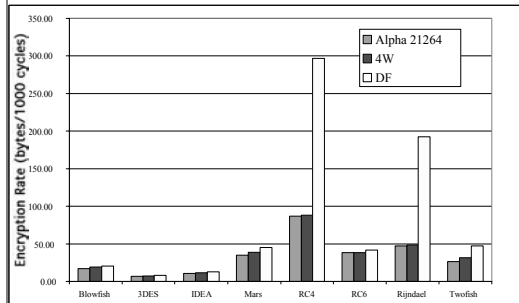
SSL Session Breakdown Focus: Secret-Key Ciphers



Benchmark Suite

Cipher	Key Size	Blk Size	Rnds/Blk	Author	Application
3DES	112	64	48	CryptSoft	SSL, SSH
Blowfish	128	64	16	CryptSoft	Norton Utilities
IDEA	128	64	8	Ascom	PGP, SSH
Mars	128	128	16	IBM	AES Candidate
RC4	128	8	1	CryptSoft	SSL
RC6	128	128	18	RSA Security	AES Candidate
Rijndael	128	128	10	Rijmen	AES Standard
Twofish	128	128	16	Counterpane	AES Candidate

Cipher Throughput Analysis



Alpha 21264 vs. 4W

- All except Mars and Twofish were within 10% of the actual machine tests

- Mars 11%, Twofish 15%

Alpha 21264 vs. DF

- Blowfish, IDEA, and RC6 are running within 20% of DF performance

- Mars 29%, Twofish 76%

- RC4 and Rijndael are outliers

Characteristics of Cipher Kernels

- Diffusion (goal of cryptography)
 - Goal is to randomly impress upon each group of output bits some information from each of the input bits
 - Process needs to be reversible
 - Should result in a random perturbation of each output bit with a probability > 50%
- Cipher kernel loops run about 16 times on each block of data, mixing the data more and more each round
- Cipher kernels have very little/to no parallelism
 - Usually a very long recurrence

Breakdown of Cipher Operations

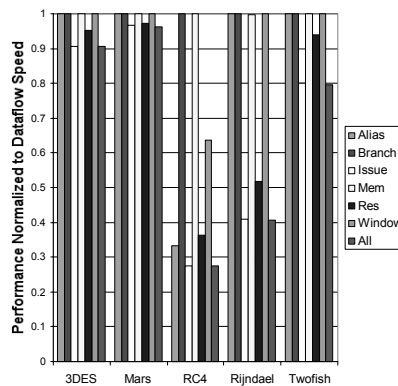
- Rotates
 - Rotate the bits in a register
- Modular Addition
- Modular Multiplication ($2^N + 1$ prime modulus operations)
- Substitutions
 - Table-based substitutions
 - SBOX is a table of values indexed with plaintext (a byte) that produces the result of the key-parameterized function
- General Permutations
 - XBOX is map N bits onto N bits with any arbitrary exchange of individual bits

Blowfish Cipher Kernel

```
for (ii=0; ii < BF_ROUNDS; ii++) {  
    register BF_LONG tmp;  
    r ^= p[ii+1];  
    r ^= (((s[(int)(l >> 24L)] +  
            sbbox[0x0100 + ((int)(l >> 16L) & 0xff)]) ^  
            sbbox[0x0200 + ((int)(l >> 8L) & 0xff)]) +  
            sbbox[0x0300 + ((int)(l) & 0xff)]) & 0xffffffffL;  
    tmp = r; r = l; l = tmp;  
}  
r ^= p[BF_ROUNDS+1];
```

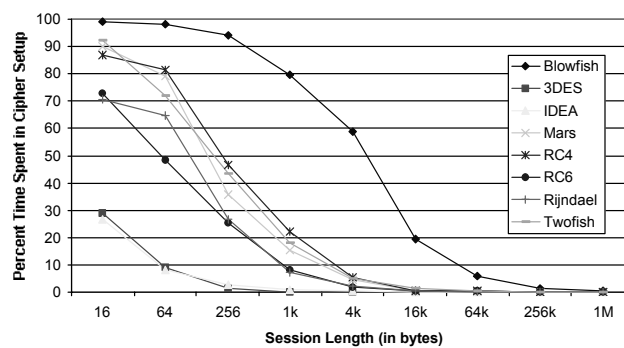
Cipher Bottleneck Analysis

Analysis of Bottlenecks in Cipher Kernels



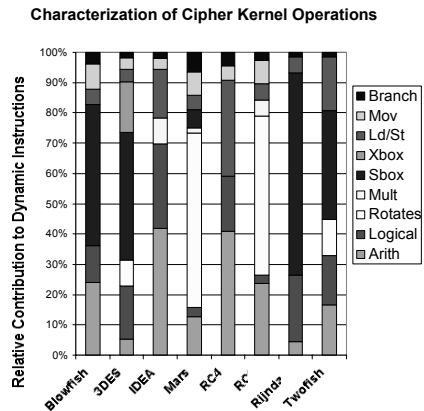
- ĩ Alias - impact of stalling loads in the pipeline until all earlier store addresses have been resolved
- ĩ Branch - effects of mispredictions
- ĩ Issue - impact of reducing issue width
- ĩ Mem - impact of introducing a realistic memory system
- ĩ Res - impact of limited functional unit resources
- ĩ Window - impact of a limited-size instruction window

Cipher Relative Run Time Cost Focus: Kernel Loop



- ĩ 3DES and IDEA are small even for 16 byte sessions
- ĩ Mars, RC4, RC6, Rijndael, and Twofish drop well below 10% for 4k+ byte sessions
- ĩ Blowfish is outlier, drops below 10% only for 64k+ byte sessions

Cipher Kernel Characterization



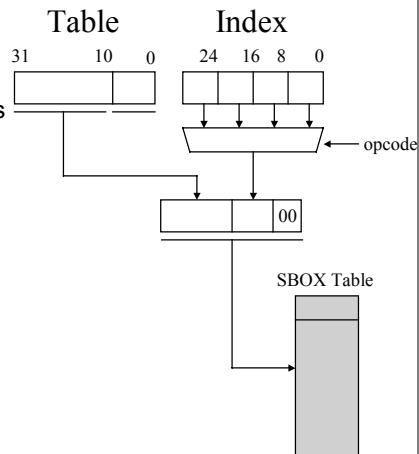
- SBOX - substitutions
- XBOX - permutations
- IDEA, Mars, RC4, and RC6 rely on arithmetic computations; benefit from more resources (multiplies) and from faster operations (rotates)
- Blowfish, 3DES, Rijndael and Twofish rely on substitutions; benefit from increased memory bandwidth and accesses

Architectural Extensions

- All instructions are limited to two register input operands and one register output
 - ROL and ROR (rotates) for 64 and 32-bit data types
 - ROLX and RORX support a constant rotate of a register input, followed by an XOR with another register input
 - MULMOD computes the modular multiplication of two register values modulo the value 0x10001
 - SBOX speeds the accessing of substitution tables with 256-entry tables and 32-bit contents
 - SBOXSYNC synchronize the SBOX table with memory
 - XBOX implements a portion of a full 64-bit permutation

Crypto-Specific ISA

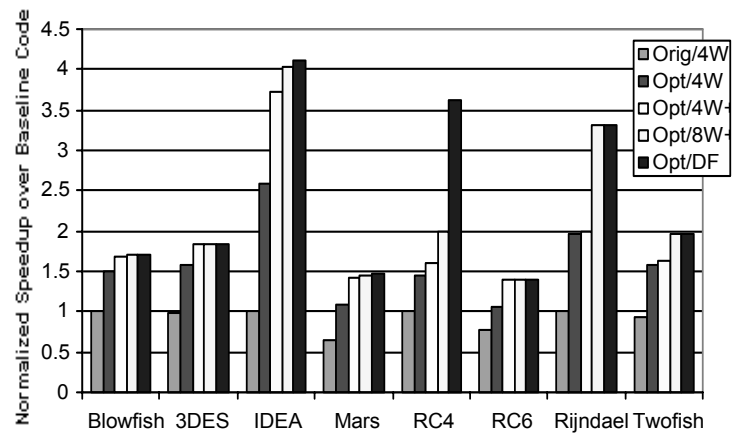
- frequent SBOX substitutions
 - $X = \text{sbox}[y \gg c] \& 0\text{xff}$
 - $X = \text{sbox}[m[j^c][1]]$
- SBOX instruction eliminates address generation
 - All SBOX tables are aligned to a 1k byte boundary
 - Address generation becomes zero-latency bit concatenation
- Stores to SBOX storage are not visible by later SBOXis until
 - An SBOXSYNC is executed
 - An alias bit is set
- SBOX instruction
 - Incorporates byte extract
 - Speeds address generation
- Original 4-cycle operation becomes a 1-cycle CryptoManiac instruction



Crypto-Specific ISA (cont.)

- Ciphers often mix logical/arithmetic operation
 - Excellent diffusion properties plus resistance to attacks
- ISA supports instruction combining
 - Logical + ALU op, ALU op + Logical
 - Eliminates dangling XORis
- Reduces kernel loop critical paths by nearly 25%
 - Small (< 5%) increase in clock cycle time

Performance of ISA Extensions



CryptoManic ISA

```

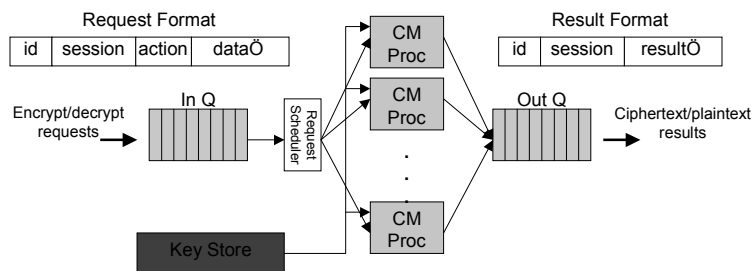
i bundle := <inst><inst><inst><inst>
i inst := <operation pair><dest><operand 1><operand 2><operand 3>
i operation pair := <short><tiny>|<tiny><short>|<tiny><tiny>|<long><nop>
i tiny := <xor> | <and> | <inc> | <signext> | <nop>
i short := <add> | <sub> | <rot> | <sbox> | <nop>
i long := <mul> | <mulmod>
    
```

<i>Instruction</i>	<i>Semantics</i>
Add-Xor r4, r1, r2, r3	$r4 \leftarrow (r1 + r2) \oplus r3$
And-Rot r4, r1, r2, r3	$r4 \leftarrow (r1 \& r2) \ll r3$
And-Xor r4, r1, r2, r3	$r4 \leftarrow (r1 \& r2) \oplus r3$

The CryptoManiac Processor

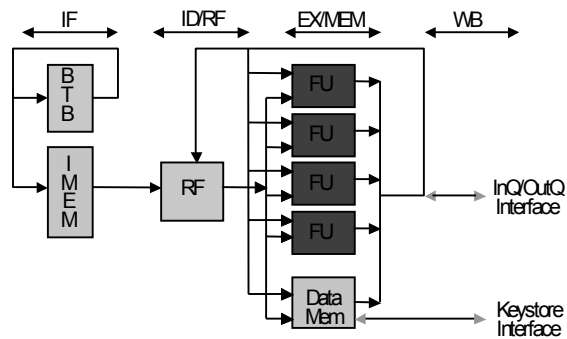
- ï A 4-wide 32-bit VLIW machine with no cache and a simple branch predictor
- ï Supports a triadic (three input operands) ISA that permits combining of most cryptographic operation pairs for better clock cycle utilization
- ï Can be combined into chip multiprocessor configurations for improved performance on workloads with inter-session and inter-packet parallelism

A Case Study: CryptoManiac



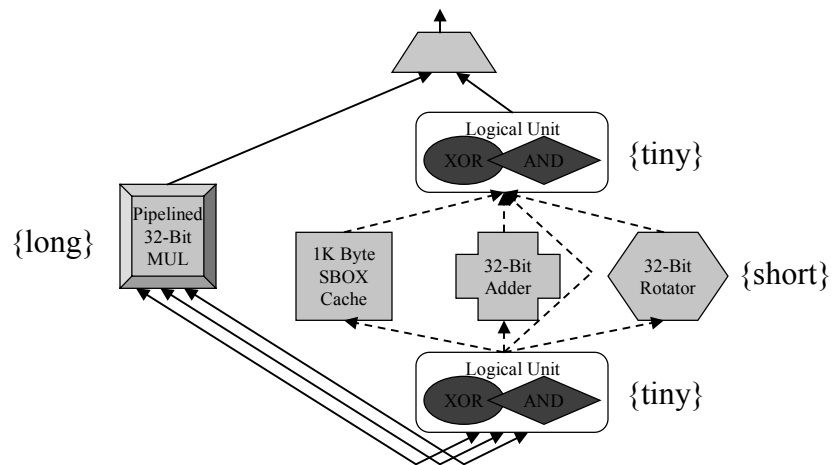
- ï Efficient crypto-processor for private-key ciphers
 - ï Chip-multiprocessor design extract inter-session parallelism
- ï A highly specialized and efficient design
 - ï Crypto-specific microarchitecture, ISA, compiler, and circuits

Crypto-Specific Microarchitecture



- Simple 4-wide 32-bit statically scheduled VLIW
 - No caches needed, small instruction and data RAMs
 - 16-entry BTB predicts branches
- Resulting design is small and efficient

Crypto-Specific Functional Unit

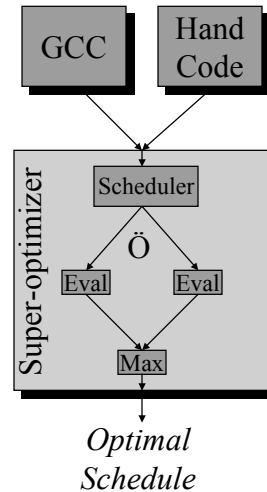


Timing and Area Results

Timing and Area Estimates for Various CryptoManiac Configurations				
	4W Comb	3W Comb	2W Comb	4W NoComb
Timing Estimate	2.78 ns	2.66 ns	2.54 ns	2.76 ns
Area Estimate	1.39mm x 1.39mm	1.33mm x 1.33mm	1.26mm x 1.26mm	1.3mm x 1.3mm
Power Estimate	606.37 mW	593.51 mW	568.50 mW	586.86 mW
Synthesis Constraint	3 ns	3 ns	3 ns	3 ns
Critical Path	byps-lgc-add-lgc	byps-lgc-add-lgc	byps-lgc-add-lgc	add

Crypto-Specific Compiler

- ï Crypto-kernels
 - ï Small code size
 - ï Deterministic dependencies
 - ï Deterministic control
 - ï Deterministic latency
 - ï Little loop-level parallelism
- ï Super-optimizer identifies *optimal* schedule
 - ï Generates all schedules
 - ï Chooses best given constraints and goals
- ï Focuses uArch evaluation



Blowfish Cipher Kernel

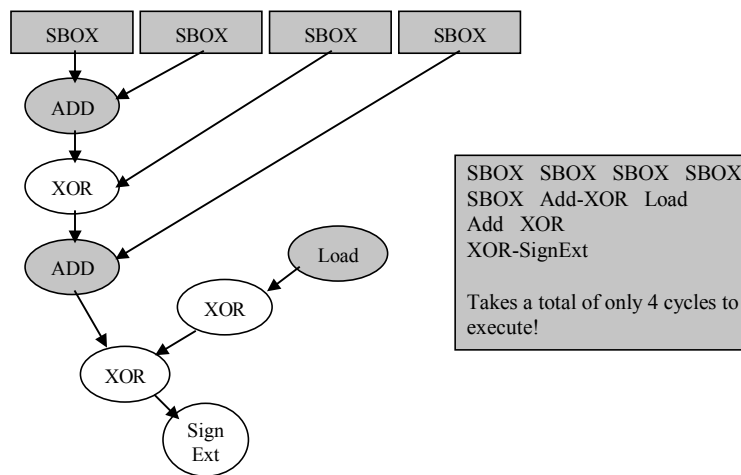
```

for (ii=0; ii < BF_ROUNDS; ii++) {
    register BF_LONG tmp;
    r ^= p[ii+1];
    r ^= (((s[(int)(l >> 24L)] +
        sbbox[0x0100 + ((int)(l >> 16L) & 0xff)]) ^
        sbbox[0x0200 + ((int)(l >> 8L) & 0xff)]) +
        sbbox[0x0300 + ((int)(l) & 0xff)]) & 0xffffffffL;

    tmp = r; r = l; l = tmp;
}
r ^= p[BF_ROUNDS+1];

```

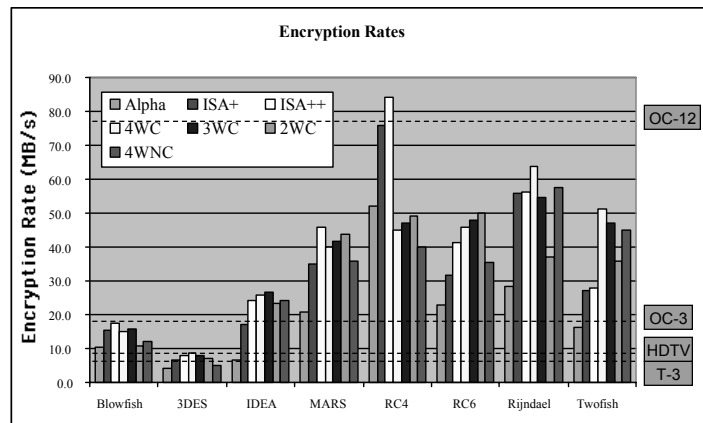
Scheduling Example: Blowfish



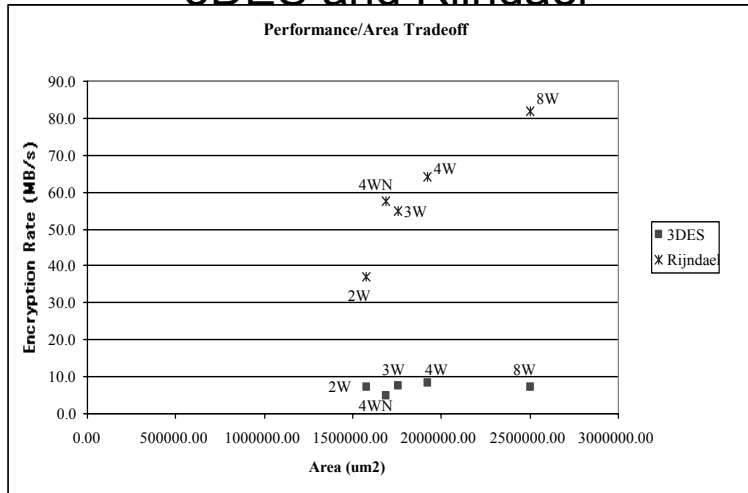
Simulation Methodology

- Use SimpleScalar as baseline processor
- Compiled original algorithms on the Alpha
- Broke simulation for the algorithms into 2 sections
 1. Startup and shutdown code
 2. Cipher Kernel
- Converted the Alpha assembly code of the Cipher kernels into their own ISA
- CryptoManiac results could be captured by
 - SimpleScalar results running algorithm WITHOUT the Cipher kernel
 - + Simulating the Cipher Kernel in a special interpreter
- Or could be captured by
 - Modifying SimpleScalar to switch to Cipher Interpreter when a special instruction is fetched, and switch back when finished.

Encryption Performance



Special Case Studies: 3DES and Rijndael



Conclusion

- ï Two hardware/software-design techniques to improve the performance of secret-key cipher algorithms
 - ï Add instruction support for fast substitutions, general permutations, rotates, and modular arithmetic
 - ï SBOX eliminates address generation
 - ï Overall speedup of 59% over baseline machine w/ rotates
 - ï Design an efficient 4-wide VLIW cryptographic co-processor called the CryptoManiac
 - ï Instruction combining - efficient utilization of clock cycle
 - ï Rijndael runs 2.25 times faster with 1/100th area and power of a 600MHz Alpha processor