

23-Feb-04 (1)

# CEG5010: A Massively Parallel RC4 Encryption Engine

23-Feb-04 (2)

## Overview

- Introduction
- Algorithm
- Architecture
- Implementation
- Results
- Conclusion

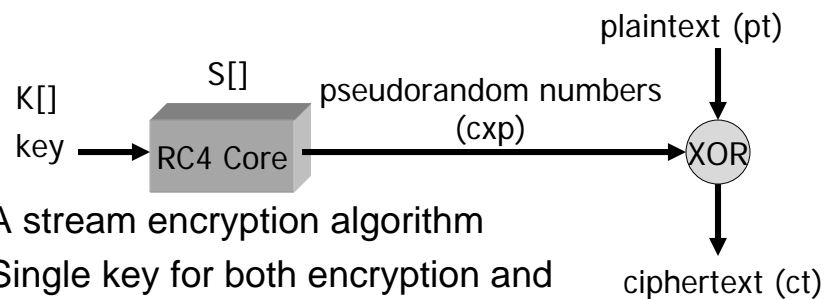
23-Feb-04 (3)

## FPGAs as cryptographic accelerators

- To date application to cryptanalysis limited: high cost and low density
- Recent improvements make FPGAs increasingly suitable for cryptanalysis
- High density (58k~4M system gates)
- High on-chip memory bandwidth
- Algorithm hardwired (more parallelism)
- Reconfigurable for new algorithms

23-Feb-04 (4)

## RC4 Algorithm – Overview



- A stream encryption algorithm
- Single key for both encryption and decryption
- A PRNG uses the key as seed
- Key size ranges from 40 to 256 bits
- S[] state memory, K[] is the key

23-Feb-04 (5)

## RC4 Algorithm – Key Scheduling

```

key_scheduling() {
    /* initialisation */
    for i = 0 to 255
        s[i] = i;

    /* scrambling */
    j = 0;
    for i = 0 to 255
    {
        j = j + K[i % k_size] + S[i];
        swap S[i] and S[j];
    }
}

```

23-Feb-04 (6)

## RC4 Algorithm – PRNG

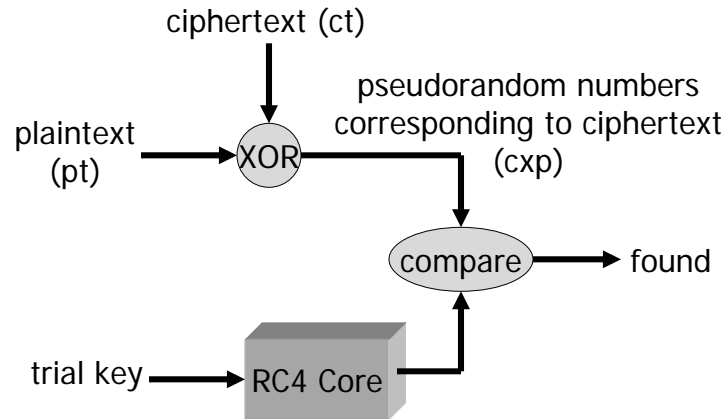
```

PRNG() {
    i = 0;
    j = 0;
    while not end of stream
    {
        i = (i + 1) % 256;
        j = (j + S[i]) % 256;
        swap S[i] and S[j];
        t = S[i] + S[j];
        ct[i] = pt[i] xor S[t];
    }
}

```

23-Feb-04 (7)

## Known Plain Text Attack



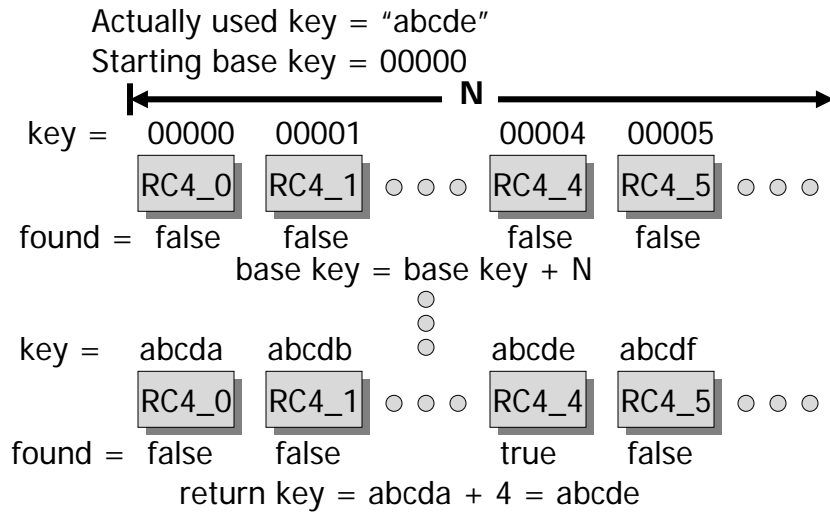
23-Feb-04 (8)

## Known Plain Text Attack

- Both plain text and cipher text are known
  - e.g. email has headers from which plain text can be determined
- Could be adapted for a cipher text only attack
  - e.g. look for 7-bit ascii characters in decrypted text

23-Feb-04 (9)

## Key Search



23-Feb-04 (10)

## Key Search

```

keysearch(){
  k = 0; cxp = pt xor ct;
  forever {
    for i = 0 to N-1 (in parallel){
      found = rc4(cxp, k + i)
      if (found(i))
        return k + i;
    }
    k = k + N;
  }
}

```

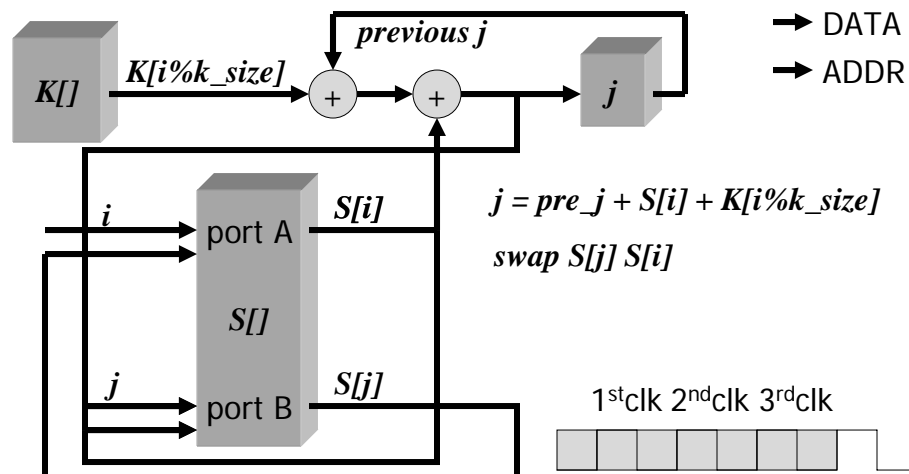
23-Feb-04 (11)

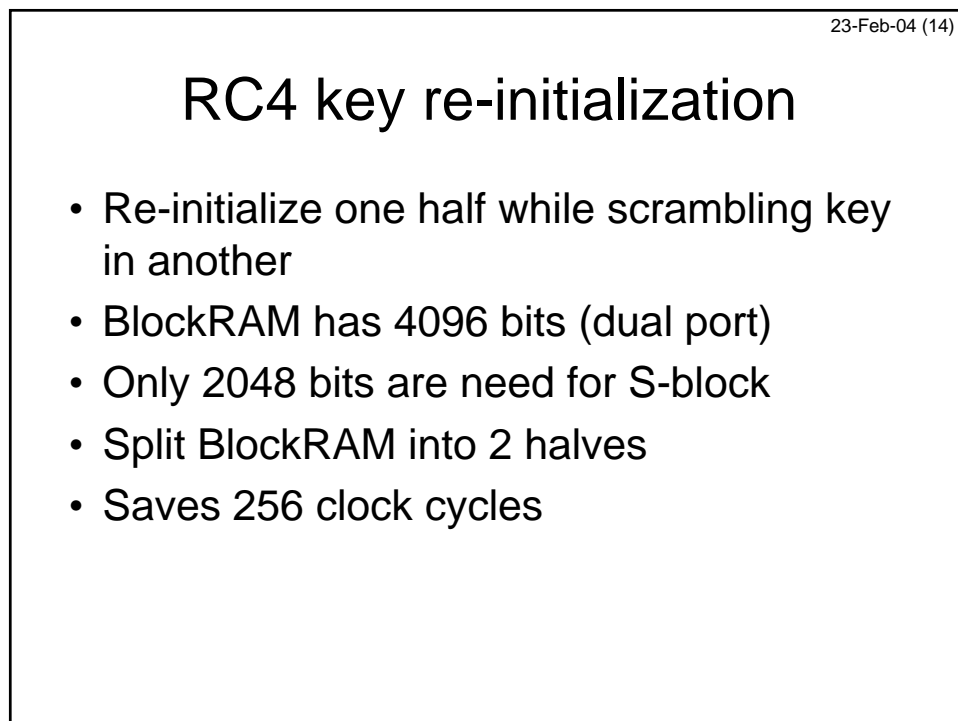
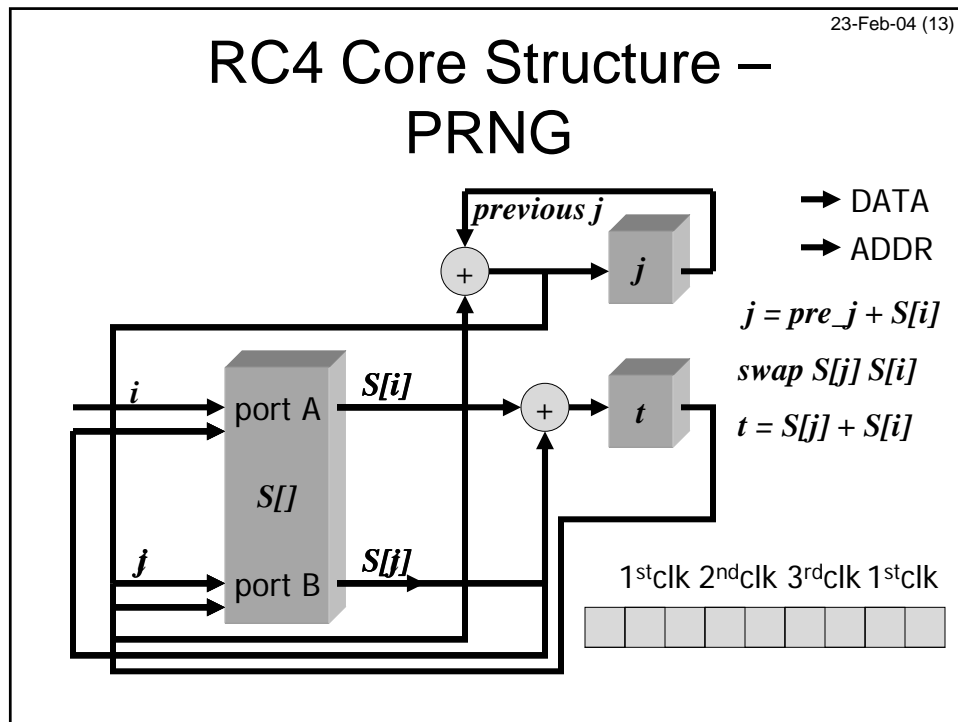
## RC4 Core Structure

- Key array,  $K[]$ , indexed by counter  $i$
- S-block,  $S[]$ , dual port Block RAM
- 3 x 8-bit adders, for
  - $j = j + S[i] + K[i]$
  - $t = S[i] + S[j]$
- 2 x 8-bit registers, for  $j$  and  $t$
- 3 clock cycles for each iteration in both phases
- S-block is re-initialized for next key in key scheduling

23-Feb-04 (12)

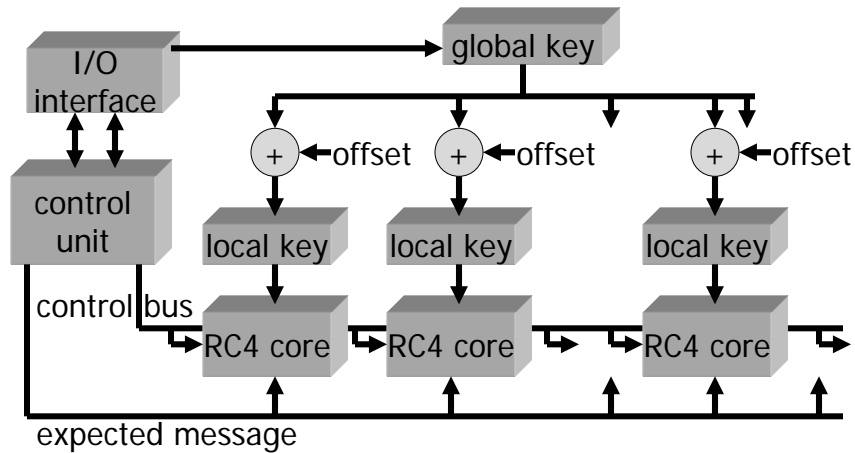
## RC4 Core Structure – Key Scheduling





23-Feb-04 (15)

## Parallel Key Search Datapath



23-Feb-04 (16)

## Parallel Key Search Datapath

- 96 RC4 cores on a single chip
- Single control unit
- Global  $i$ -counter, local  $j$ -counter
- Each core has an offset  $\{0 \dots 95\}$  and a copy of local key  
(local key = global key + offset)
- When core finds the key, core's offset and global key are sent to host

23-Feb-04 (17)

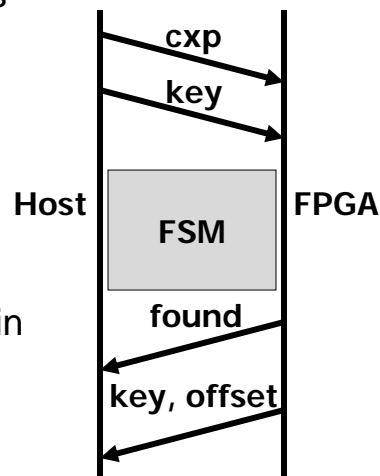
## Parallel Key Search Control

- Dispatch the global key to cores
- Start cores to decrypt the message
- All cores work synchronously
- If no valid key is found, update global key by adding 96 and loop
- If a valid key is found, stop all cores and report result to host

23-Feb-04 (18)

## Host Interface

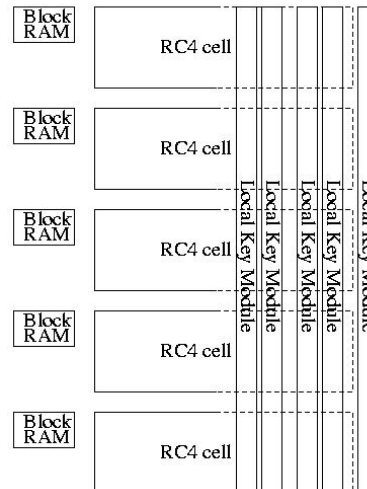
- Host sends expected results
- Host sends starting global key
- FSM starts, host waits
- Valid key found, FSM stops
- Host reads back the global key and core's offset
- Can start from any location in the key space



23-Feb-04 (19)

## Implementation – Core Design

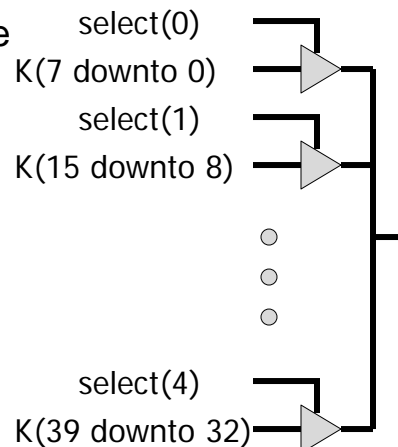
- Each core associated with a Block RAM
- A single core can fit into a 6x4 CLB array
- All core signals are locally routed
- For local key, 40-bit adder with carry chain use a column of 20 slices



23-Feb-04 (20)

## Implementation – Core Design

- Logic consumed by key byte selection
  - $K[i \% k\_size]$
- TBUFs are used instead
- Wired OR all outputs
- Eliminated 8 5-to-1 MUXs
  - Saves 8 slices for each core
  - Saves 768 slices in design



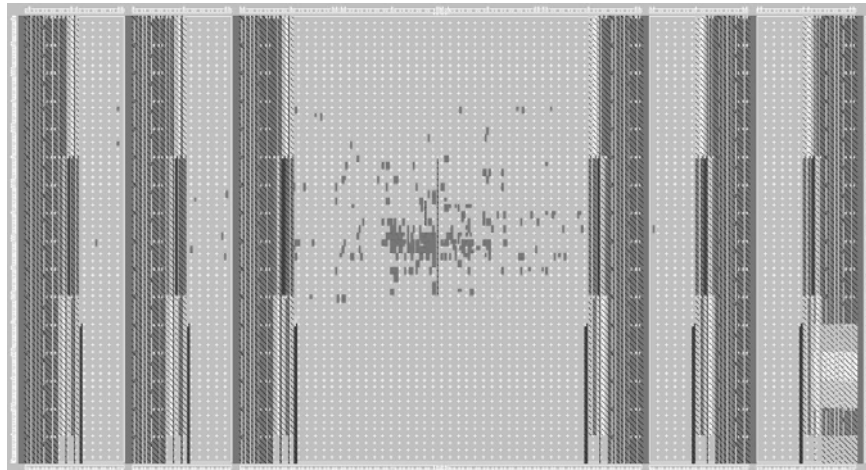
23-Feb-04 (21)

## Implementation – Parallel Design

- Structural design with reusable core module
- Uses RLOC attribute for placement of cores
- The control unit resides in the center area
- Buffers are added for large fanout signals
- Floor planning tools used to optimize the design

23-Feb-04 (22)

## Implementation – Parallel Design



23-Feb-04 (25)

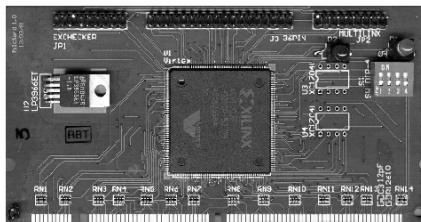
## Implementation – VHDL

- After stating constraints in VHDL, most parts of the design are fixed
- Floorplanner used to adjust the I/O and control units
- Hard macro generated from FPGA editor (dramatically reduces place and route time)

23-Feb-04 (26)

## Results – Platform

- Chip: Xilinx VirtexE (XCV1000E-6)
- Board: Pilchard (memory slot based card)



- Tools: Xilinx ISE4.1i
- Host: PIII 800MHz

• OS: Linux

23-Feb-04 (27)

## Results – Area Utilization

Resources	Use Count	Utilization Rate
DLLs	1 out of 4	25%
Block RAMs	96 out of 96	100%
SLICEs	5178 out of 12288	42%
TBUFs	4608 out of 12544	36%

23-Feb-04 (28)

## Results – Timing Spec.

- Individual core frequency: ~100MHz
- Max. design speed: ~80MHz
- Reported by Xilinx TRACE tools
- Key search engine frequency: 50MHz
- Interface frequency (system memory bus): 100MHz
- Critical path is: from key through two adders to Block RAM inputs

23-Feb-04 (29)

## Results – Performance

- 256 iterations for key scheduling
  - 8 iterations for PRNG
  - 3 clock cycles for each iteration
  - 20ns for each clock cycle
- ⇒ 15us for a single key (single core)
- ⇒ 6,000,000 keys/s (single FPGA)
- ⇒ 50hr for 40-bit key search (single FPGA)

23-Feb-04 (30)

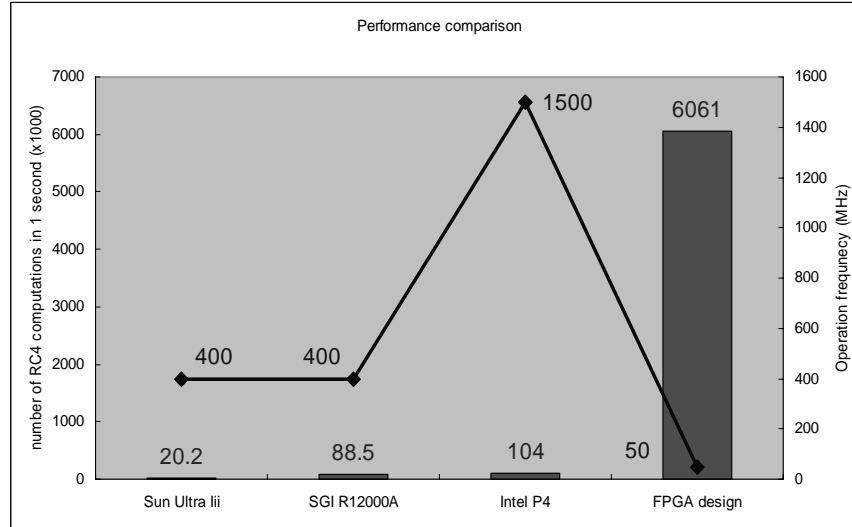
## Results – Performance

- Time to complete an encryption

Platform	Frequency (MHz)	Time (us)	Normalized time
Sun Ultra Ili	400	49.4	299
SGI R12000A	400	11.3	68.6
Intel P4	1500	9.62	58.3
This work	50	0.165	1

23-Feb-04 (31)

## Results – Performance



23-Feb-04 (32)

## Summary of approach

- Parallelism in RC4 core allows several operations to be completed in a single cycle
- On-chip resources were used to achieve a very low latency, high bandwidth memory interface
- Dual-ported memory for higher memory transfer efficiency
- Floorplanning was used to minimize interconnect delays
- A large number of the decryption cores were used in parallel

23-Feb-04 (33)

## Further Speedups

- Slightly faster part would allow us to operate at 100MHz (2x improvement)
  - Parts with more Block RAMs would enable us to increase number of cores
    - e.g. virtex EM XCV812E has 280 Block RAMs so we could fit 3x more cores (3x improvement)
- ⇒ Together would make a 6x improvement
- It may also be possible to utilize the unused slices to add more cores (currently only 42% of slices are used)

23-Feb-04 (34)

## Conclusion

- Described a high performance RC4 core which can perform an encryption in 15us
- Integrated 96 cores on a single FPGA
- 58x faster than 1.5GHz Pentium 4
- With improving density and on-chip memory, FPGAs are becoming increasingly suitable for cryptanalysis applications