"Implementation Trade-offs of Triple-DES in the SRC Reconfigurable Computing Environment"

Osman Devrim Fidanci [1], Hatim Diab [1], Tarek El-Ghazawi [1], Kris Gaj [2] and Nikitas Alexandridis [1]

1: George Washington University

2: George Mason University

Outline

- Why Reconfigurable Computing ?
- SRC-6E General Purpose Reconfigurable Computer
  - Hardware Architecture
  - Programming Model
- Triple DES Encryption Algorithm
  - Single DES Algorithm
  - Triple DES with 2 Keys
- Implementation Trades-offs of Triple-DES in SRC
- Conclusions

Why Reconfigurable Computing ?

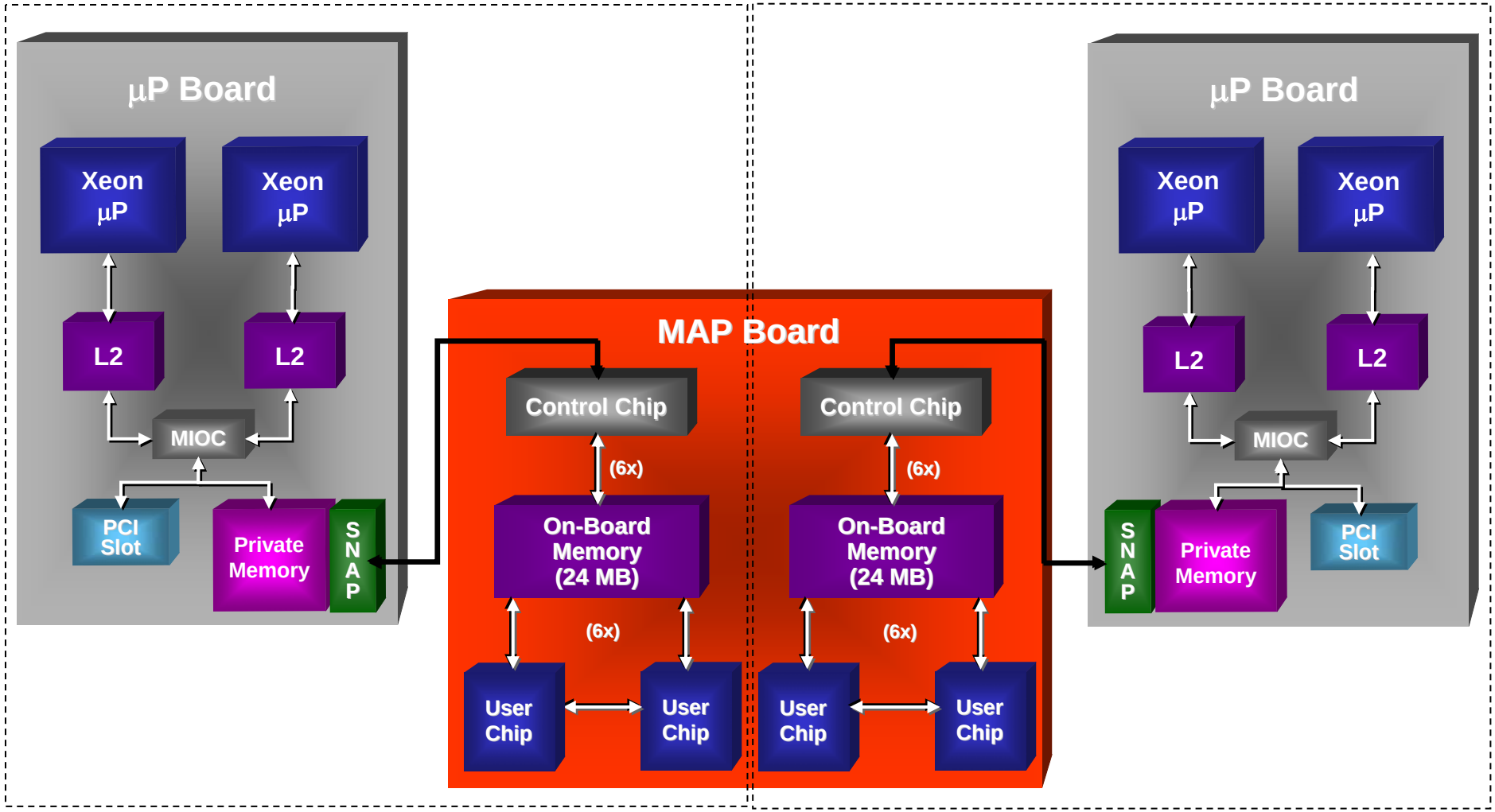
Performance

- Direct instantiation of hardware results in better efficiency
- Reduces I/O bandwidth requirements by elimination of Load/Store paradigm

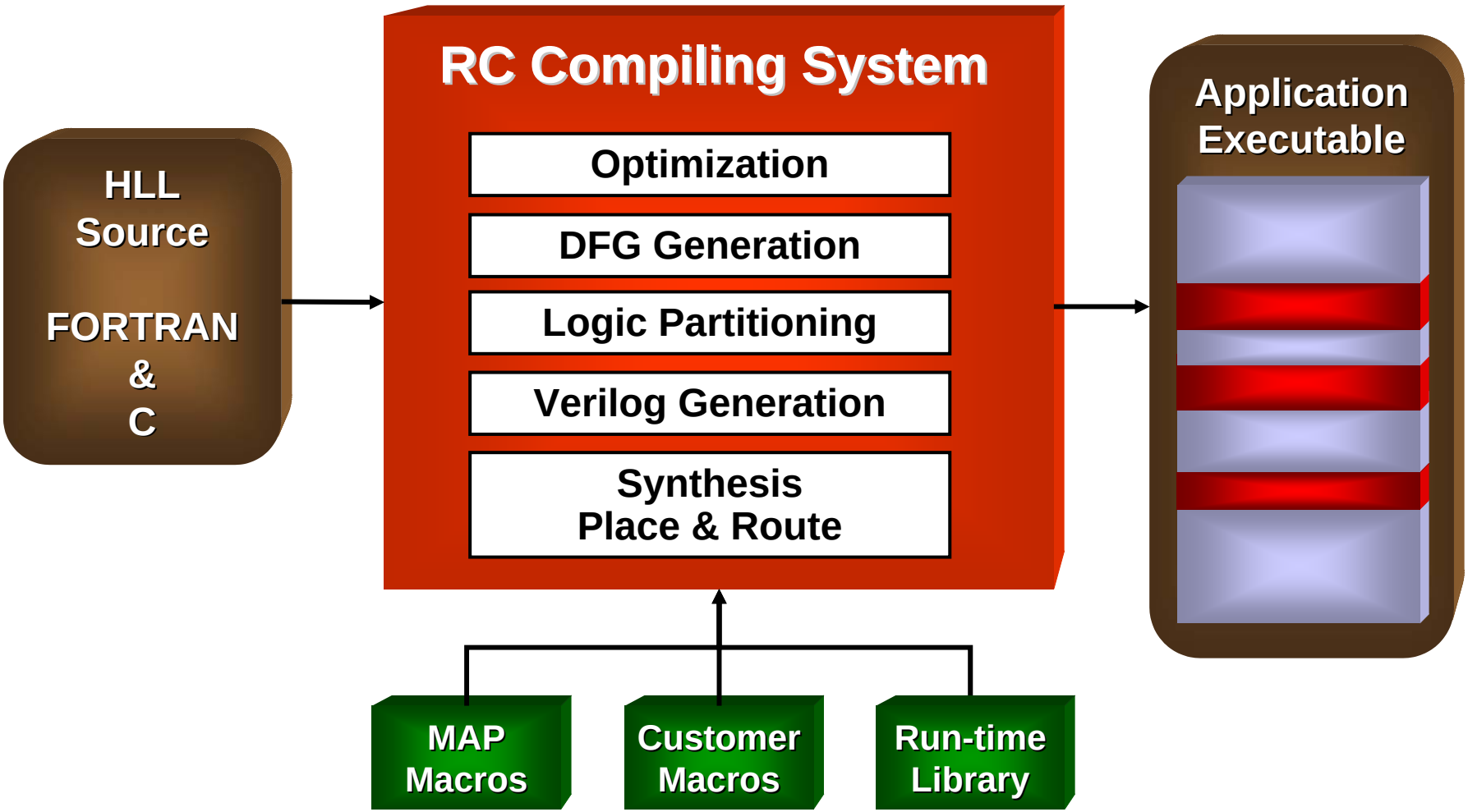
Scalability

- Basic technology improving much faster than Moore's law
- Takes advantage of parallelism found in many programs

SRC-6E Hardware Architecture



SRC MAP Compiler Architecture



Advantages of SRC Hardware

- Introduction of the SNAP card
  - Data feed to the FPGA though Memory bus with 800 MB/s peak (theoretical)
  - Eliminating the PCI bottleneck in traditional FPGA cards

SRC-6E System SW

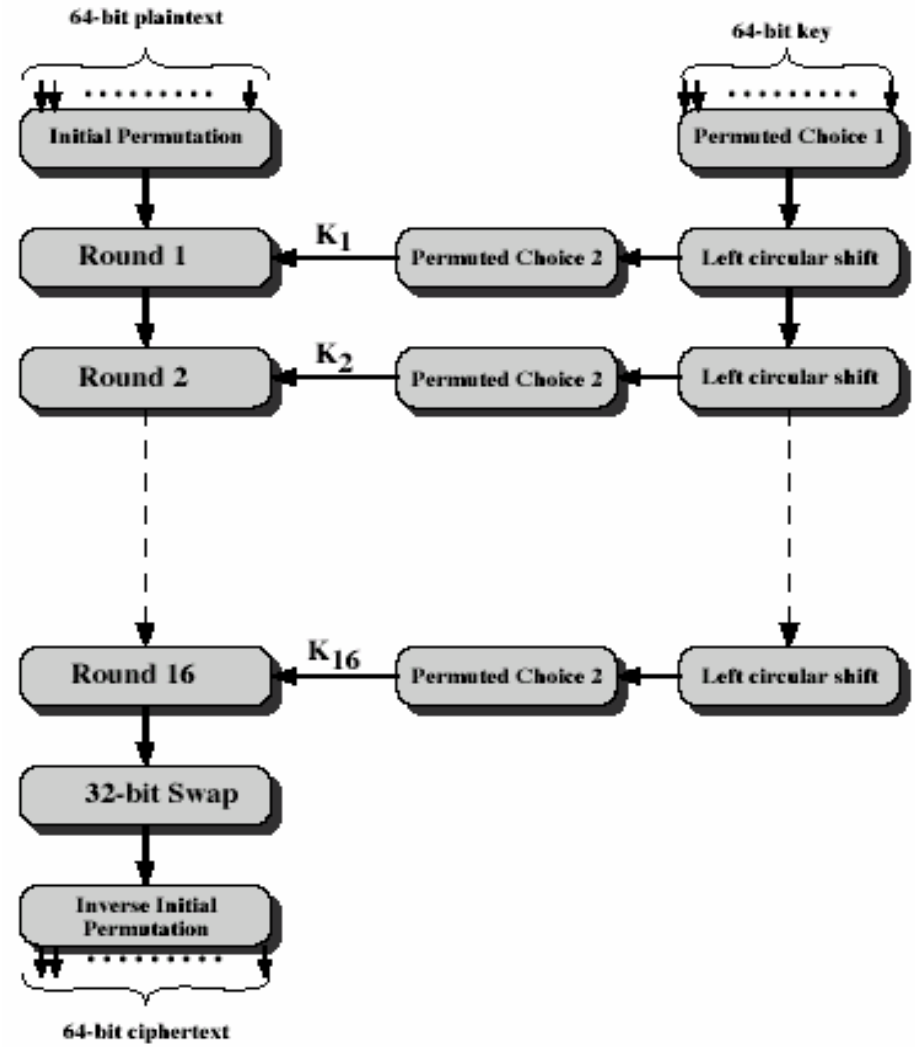
- **System**
  - Linux
    - Red Hat 7.2
    - Driver and Library additions to support SNAP and MAP
- **Compilers**
  - Microprocessor and MAP
- **Tools**
  - WINE
  - FPGA
    - Synplicity Synplify Pro
    - Xilinx Integrated Software Environment

Macro Development for SRC

- User macros can be defined in Verilog or VHDL
- Three types of macros defined in SRC platform:
  - *Functional*
  - *Stateful*
  - *External*
- System clock frequency is 100MHz. So, User macros should be optimized for this speed.

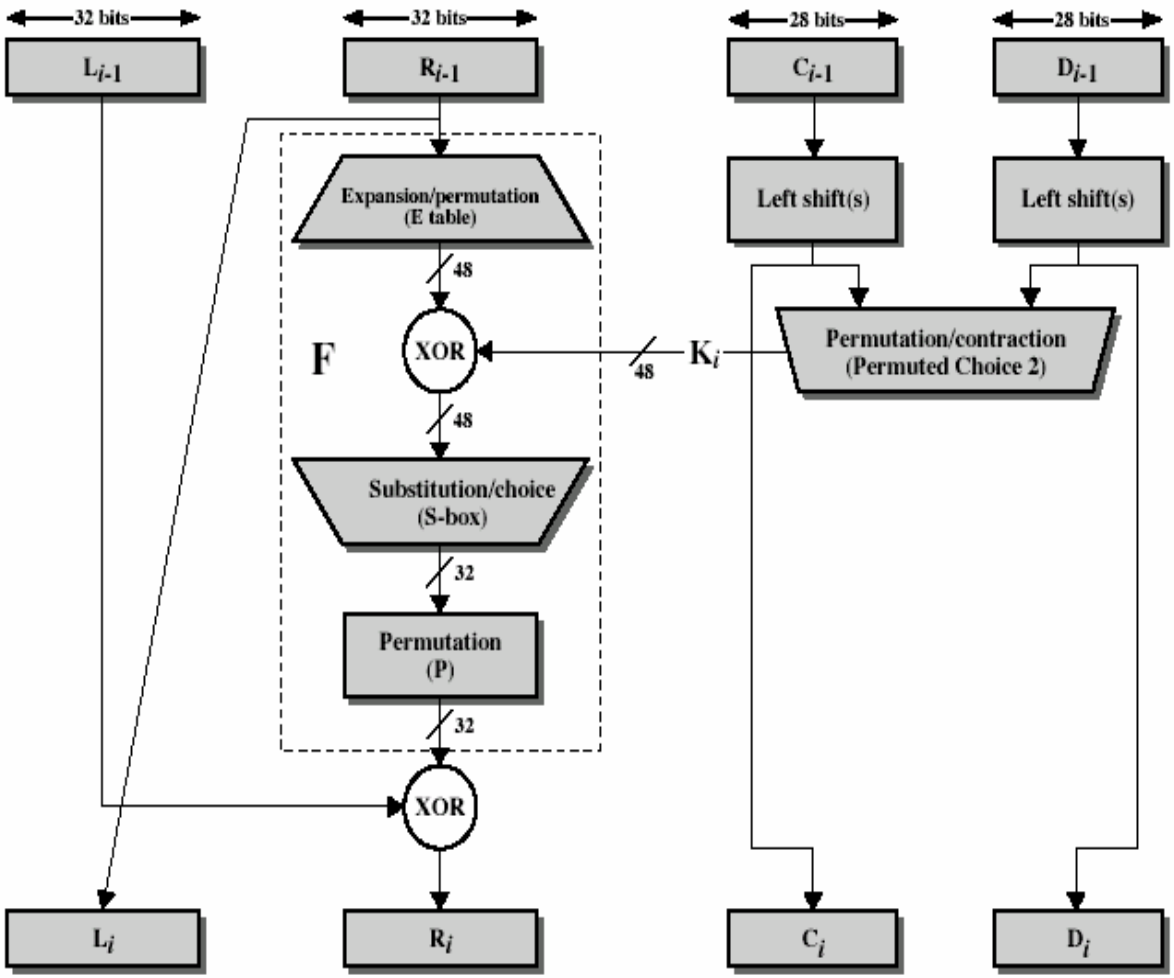
  Otherwise FIFO must be employed

General Depiction of DES Encryption Algorithm



- ✓ 64-bit inputs (plaintext and keys)
- ✓ 64-bit output (ciphertext)
- ✓ 16-round operation plus initial permutations
- ✓ Pipelined, 17-clock cycles latency
- ✓ Output generated at every clock cycle

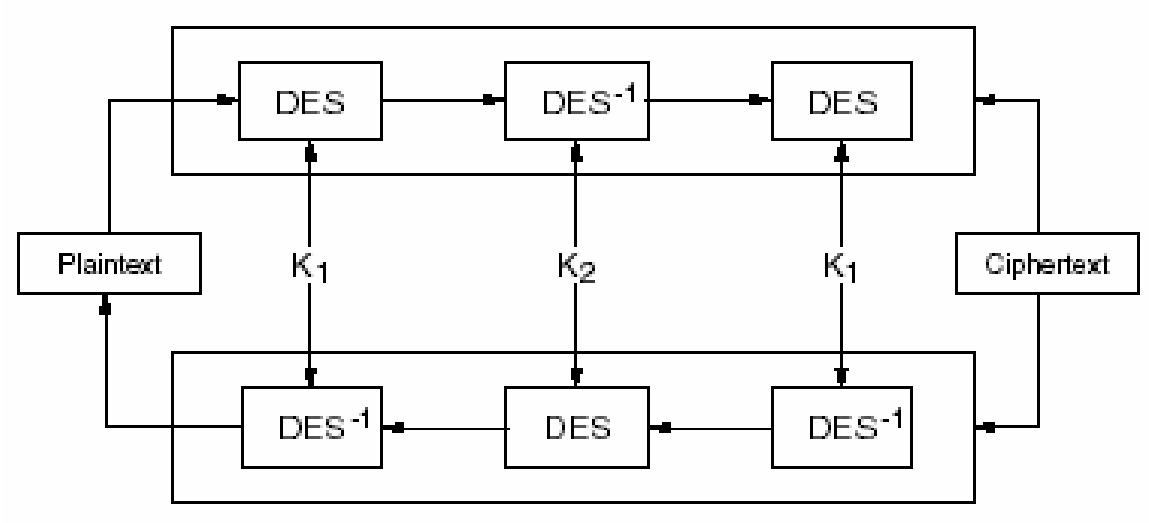
Single Round of DES Algorithm



- Expansion/permutation operation (E-table)
- Logical XOR operations
- Substitution/choice operation (S-box)
- Permutation (P)

Triple DES with 2 Keys

Encipher



Decipher

- "Triple-DES with two keys" scheme is used for backward compatibility with DES (by setting two keys identical, provides single DES encryption/decryption).
- Triple DES requires 51-clock cycle to get ciphertext

Different Implementations of Triple DES in SRC

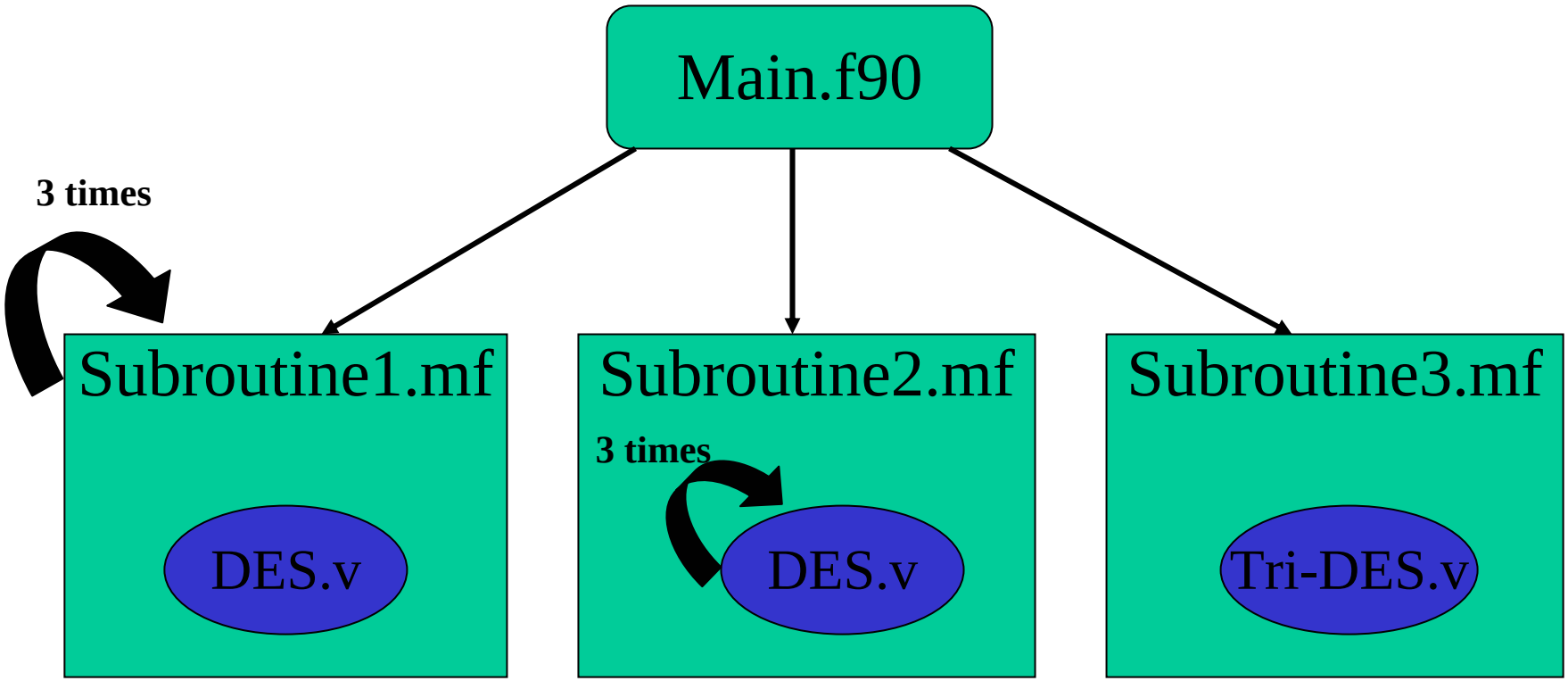
- **Case 1**

  Effect: DES macro is called three times from Fortran main file
- **Case 2**

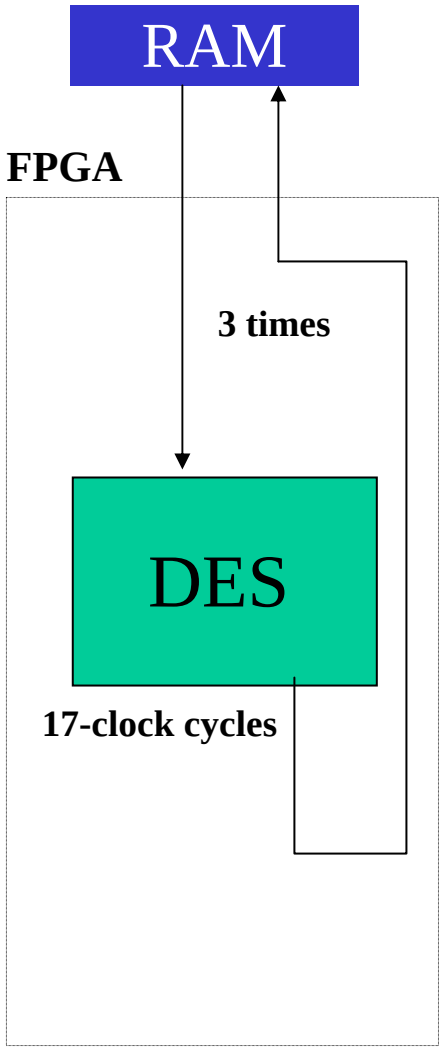
  Effect: DES macro is called three times from Fortran subroutine file
- **Case 3**

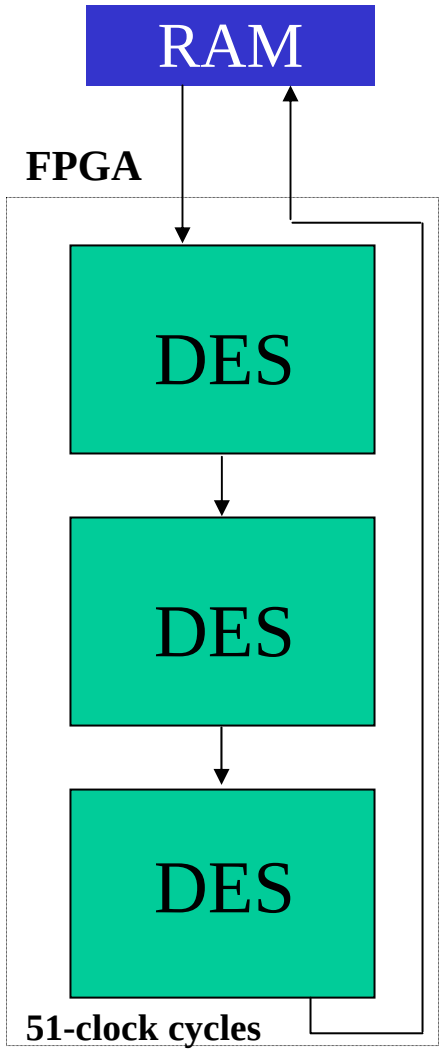
  Effect: Triple-DES macro is called once from Fortran main and subroutine files

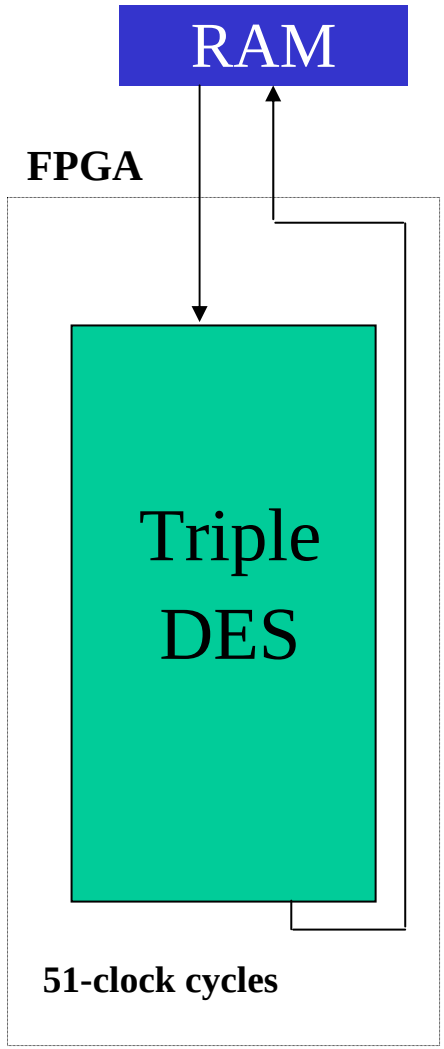
Calling DES/Tri-DES macro from Fortran HLL



Implied Architectures







Case 1 Case 2 Case 3

Processing Time Measurement

- The timer macro is simply a counter running at 100 MHz
- There are two calls:

  - start_timer: zeros out the counter

  - read_timer: gets the counter's current value

Implementation Trades-Offs of Triple DES in SRC

	Experiment(1)	Experiment(2)	Experiment(3)
Maximum Clock Speed (MHz)	102.3	100.8	101.8
CLB Slices (Tot. equiv. Gate count)	6,177 (163,835)	13,269 (382,927)	11,786 (359,635)
Total Processing Time (91 blocks of data) (ns)	4440	1820	1820

Timing Estimations

Total execution time =

Transfer time **+** **Total processing time**

μP memory ↔ on-board memory

Measured experimentally

Timing Estimations – cont.

Total processing time =

$$\left(\begin{array}{l} \text{Load time} \\ \text{first block} \end{array} + \#\text{pipeline\_stages} + (N-1) + \begin{array}{l} \text{Store time} \\ \text{last block} \end{array} \right) \times t$$

where

N – number of data blocks being processed

t – number of subroutine calls

Load time + Store time = Load/Store time = T_{LS} (unknown)

Timing Estimations – cont.

Case 1	Case 2	Case 3
#_pipeline stages: 17 #_data blocks = N: 91 #_subroutine_calls = t: 3 Load/Store time: T_{LS} Clock period: 10ns Estimated # of clock cycles for processing 91 blocks: **$[(17 + 90) + T_{LS}] \times 3 = 444$** **Load/Store time:** **$T_{LS} = 41$**	#_pipeline stages: 51 #_data blocks = N: 91 #_subroutine_calls = t: 1 Load/Store time: T_{LS} Clock period: 10ns Estimated # of clock cycles for processing 91 blocks: **$[(51 + 90) + T_{LS}] = 182$** **Load/Store time:** **$T_{LS} = 41$**	#_pipeline stages: 51 #_data blocks = N: 91 #_subroutine_calls = t: 1 Load/Store time: T_{LS} Clock period: 10ns Estimated # of clock cycles for processing 91 blocks: **$[(51 + 90) + T_{LS}] = 182$** **Load/Store time:** **$T_{LS} = 41$**

I/O Overhead (T_{LS}/Total time)

	Case 1	Case 2	Case 3
91 data blocks	27.7 %	22.5 %	22.5 %
501 data blocks	7.3 %	6.9 %	6.9 %
1001 data blocks	3.8 %	3.7 %	3.7 %
10001 data blocks	0.4 %	0.4 %	0.4 %

Ratio of processing times

$$\frac{\text{Case 1 Total processing time}}{\text{Case 2\&3 Total processing time}} =$$

$$= \frac{(17 + (N\text{-}1) + 41) \times 3}{(51 + (N\text{-}1) + 41)} =$$

N	91	501	1,001	10,001
ratio	2.44	2.83	2.91	2.99

Discussion

Case 2, i.e., calling **DES** macro **three times from** the Fortran **subroutine**, and

Case 3, i.e., calling **3DES** macro **once** from the Fortran subroutine

almost equivalent

- the same execution time
- area 13% larger for Case 2 because of the default interface between single DES modules

Discussion – cont.

Case 1, i.e., calling **DES** macro **three times from** the Fortran **main file**, and

Cases 2 and 3

very different

- approximately two times smaller area for Case 1
- over two times longer execution time caused by the larger number of iterations, larger I/O overhead (communication between FPGA and on-board memory) and smaller utilization of the pipeline

Conclusions

- SRC Programming Model enables flexible choice of the hardware architecture used to implement required function
- Implied architecture depends on
  - function (granularity) of the hardware description language macro
  - placement of macro calls in a high level language program

Conclusions – cont.

- Common features of all implemented architectures
  - deep pipelining
  - operational clock frequency 100 MHz
- Overhead associated with the run-time communication between FPGA (User Chip) and on-board memory can be made negligible for processing of large amounts of data