ECE 545
Lecture 12

Design of Controllers
Part 2

George Mason University

1

Required reading

- P. Chu, *RTL Hardware Design using VHDL*

Chapter 10, Finite State Machine: Principle & Practice

2

Class Exercise 3
CIPHER – Lightweight Architecture

3

Pseudocode

Split input I into four words, I3, I2, I1, I0, of the size of w bits each

```
A = I3; B = I2; C = I1; D=I0
B = B + S[0]
D = D + S[1]
      for i = 1 to r do
      {
            T = (B*(2B + 1)) <<< k
            U = (D*(2D + 1)) <<< k
            A = ((A ⊕ T) <<< U) + S[2i]
            C = ((C ⊕ U) <<< T) + S[2i + 1]
            (A, B, C, D)  =  (B, C, D, A)
      }
A = A + S[2r + 2]
C = C + S[2r + 3]
O = (A, B, C, D)
```

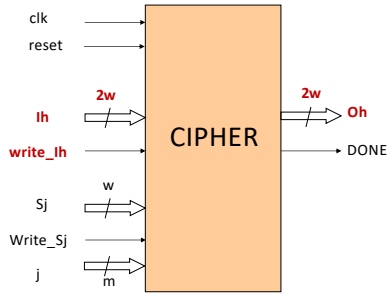

4

Notation

w: word size, e.g., w=32 (constant)
k: $\log_2(w)$ (constant)
A, B, C, D, U, T: w-bit variables
I3, I2, I1, I0: Four w-bit words of the input I
r: number of rounds (constant)
O: output of the size of 4w bits
S[j] : 2r+4 round keys stored in **one RAM**.
Each key is a w-bit word.

5

Circuit Interface



6

Interface Table

Port	Width	Meaning
clk	1	System clock.
reset	1	System reset – clears internal registers.
Ih	**2w**	**One half of an Input block. I3:I2 first, I1:I0 second.**
write_Ih	1	Synchronous write control signal for **a half of the input block Ih**. After **two halves of the input block** I are written to the CIPHER unit, the encryption of I starts automatically.
Sj	w	Round key S[j] loaded to one of the two internal memories.
write_Sj	1	Synchronous write control signal for the round key S[j].
j	m	Index of the round key S[j] loaded using input Sj.
Oh	**2w**	**One half of an Output block. O3:O2 first, O1:O0 second.**
DONE	1	Asserted for one clock cycle when the output O is ready.

Note:
m is a size of index j. It is a minimum integer, such that $2^m - 1 \geq 2r+3$.

7

7

Protocol (1)

An external circuit first loads all round keys
S[0], S[1], S[2], …, S[2r+2], [2r+3]
to **the internal memory** of the CIPHER unit.

Loading round keys is performed using inputs: Sj, j, write_Sj, clk.

Then, the external circuits, loads an input block I to the CIPHER unit, **in two clock cycles**, using inputs: **Ih**, **write_Ih**, clk.

After the input block I is loaded to the CIPHER unit, the encryption starts automatically.

8

8

Protocol (2)

When the encryption is completed, signal DONE becomes active, and the output O changes to the new value of the ciphertext.

The output O keeps the last value of the ciphertext at the output, until the next encryption is completed. Before the first encryption is completed, this output should be equal to zero.

9

9

Assumptions

- 2r+4 clock cycles are used to load round keys to internal RAMs
- one round of the main for loop of the pseudocode executes in **4 clock cycles**
- you can access only one position of **the internal memory** of round keys per clock cycle

As a result, the encryption of a single input block I should last **4r+4** clock cycles.

10

10

Pseudocode

i represents an index of a round key during encryption, not a round number

```
Split input I into four words, I3, I2, I1, I0, of the size of w bits each

A = I3; B = I2; C = I1; D=I0
B = B + S[i=0]
D = D + S[i=1]
    for n = 1 to r do
    {
        T = (B*(2B + 1)) <<< k
        U = (D*(2D + 1)) <<< k
        A' = ((A ⊕ T) <<< U) + S[i=2n]
        C' = ((C ⊕ U) <<< T) + S[i=2n + 1]
        (A, B, C, D)  =  (B, C', D, A')
    }
A = A + S[i=2r + 2]
C = C + S[i=2r + 3]
O = (A, B, C, D)
```

11

11

Pseudocode

i represents an index of a round key during encryption, not a round number

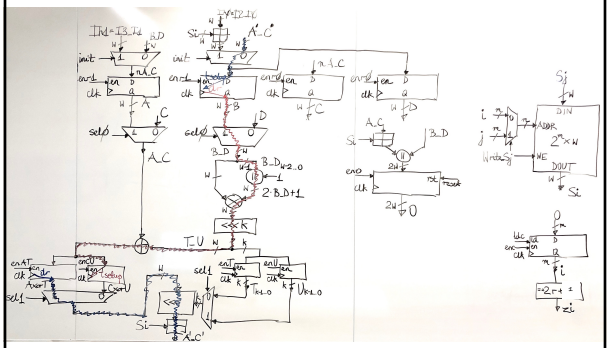
```
Split input I into four words, I3, I2, I1, I0, of the size of w bits each

A = I3; B = I2; C = I1; D=I0
B = B + S[i=0]
D = D + S[i=1]
    for n = 1 to r do
    {
        T = (B*(2B + 1)) <<< k
        AxorT = A ⊕ T
        U = (D*(2D + 1)) <<< k
        CxorU = C ⊕ U
        A' = (AxorT <<< U) + S[i=2n]
       (C, D)  =  (D, A')
        C' = (CxorU <<< T) + S[i=2n + 1]
       (A, B)  =  (B, C')
    }
A = A + S[i=2r + 2]
C = C + S[i=2r + 3]
O = (A, B, C, D)
```

12

12

Datapath

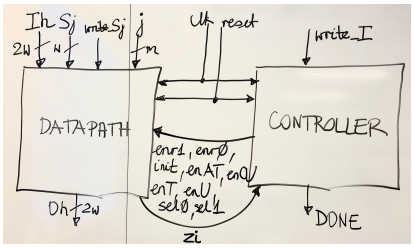


i represents an index of a round key during encryption, not a round number

13

13

Interface with the Division into the Datapath and Controller



14

14

Pseudocode

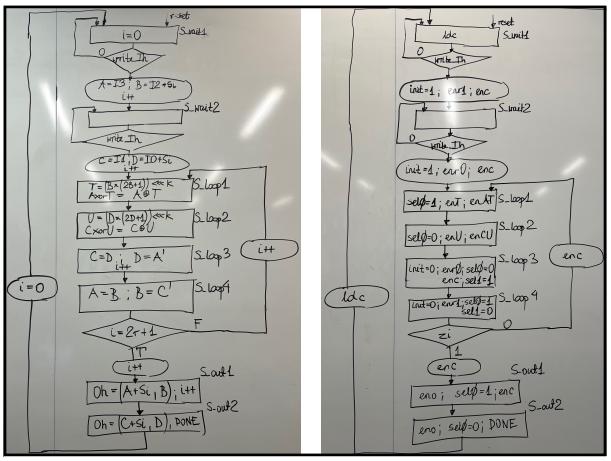
i represents an index of a round key during encryption, not a round number

Split input I into four words, I3, I2, I1, I0, of the size of w bits each

A = I3; B = I2; C = I1; D=I0
B = B + S[**i=0**]
D = D + S[**i=1**]
for **n** = 1 to r do
{
T = (B*(2B + 1)) <<< k
AxorT = A ⊕ T
U = (D*(2D + 1)) <<< k
CxorU = C ⊕ U
A' = (**AxorT** <<< **U**) + S[**i=2n**]
(C, D) = (D, **A'**)
C' = (**CxorU** <<< **T**) + S[**i=2n + 1**]
(A, B) = (B, **C'**)
}
A = A + S[**i=2r + 2**]
C = C + S[**i=2r + 3**]
O = (A, B, C, D)

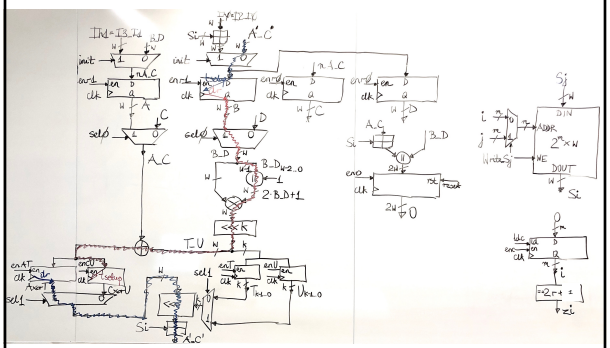
15

15



16

Datapath



i represents an index of a round key during encryption, not a round number

17

17

Timing Analysis – Key Setup & Encryption

Notation:
T_{CLK} – clock period in ns
N_M – number of encrypted message blocks I

$T'_{KeySetup}[cycles] = 2r+4$

$T'_{KeySetup}[ns] = (2r+4) \cdot T'_{CLK}$

$T'_{Encryption}(N_M)[cycles] = (4r+4) \cdot N_M$

$T'_{Encryption}(N_M)[ns] = 4 \cdot (r+1) \cdot N_M \cdot T'_{CLK}$

18

18

Timing Analysis - Throughput

Notation:
T_{CLK} – clock period in ns
N_M – number of encrypted message blocks I
w – word size

$$Thr'_{Encryption}(N_M)[Gbit/s] = \frac{4 \cdot w \cdot N_M}{4 \cdot (r+1) \cdot N_M \cdot T'_{CLK}} = \frac{w}{(r+1) \cdot T'_{CLK}}$$

r & w constants, determined by the security analysis of the cipher
For the security equivalent to AES-128, r=20 & w=32

T'_{CLK} dependent on w, returned by the FPGA tools based on the static timing analysis of the critical path

19

19

Timing Analysis – Throughput Ratio

$$\frac{Thr}{Thr'} = \frac{\frac{4w}{(r+2)} \cdot T_{ck}}{\frac{w}{(r+1)} T'_{ck}} = 4 \cdot \frac{r+1}{r+2} \cdot \frac{T'_{ck}}{T_{ck}} \approx 4 \cdot \frac{1}{2} = 2$$

Throughput decreases by approximately a factor of 2

20

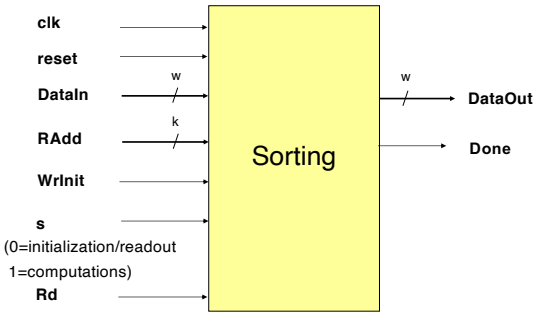
20

Class Exercise 4
Sorting

21

21

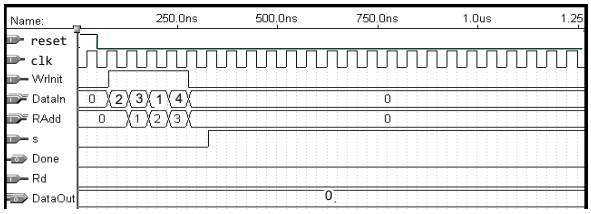
Sorting



22

22

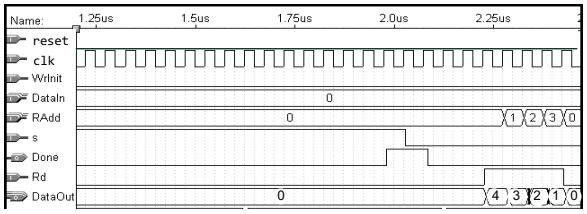
Simulation results for the sort operation (1)
Loading memory and starting sorting



23

23

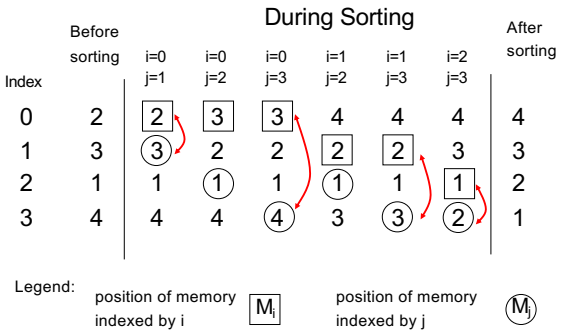
Simulation results for the sort operation (2)
Completing sorting and reading out memory



24

24

Sorting – Example for N=4

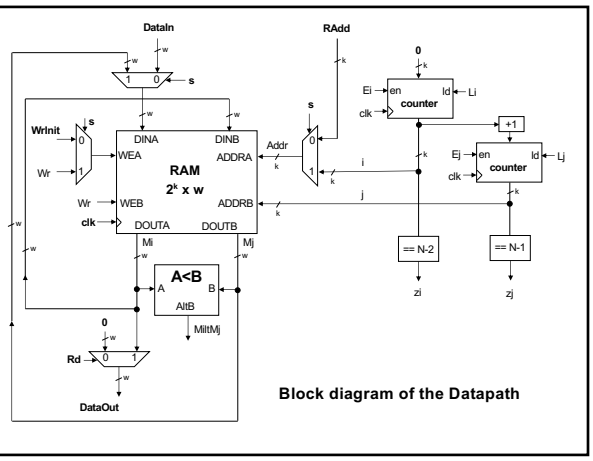


25

Pseudocode

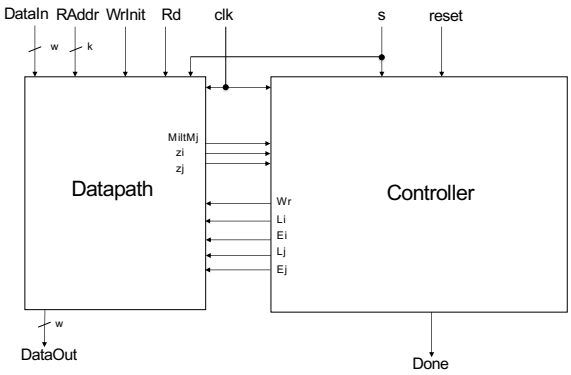
```
wait for s=1
for i=0 to N-2 do
    for j=i+1 to N-1 do
        if Mi < Mj then
            Swap Mi with Mj
        end if
    end for
end for
Done
wait for s=0
go to the beginning
```

26



27

Interface with the division into the Datapath and Controller



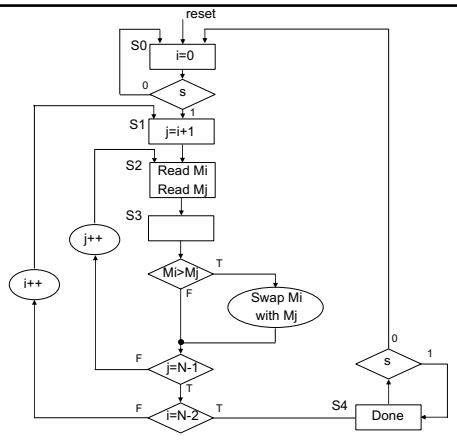
28

Pseudocode

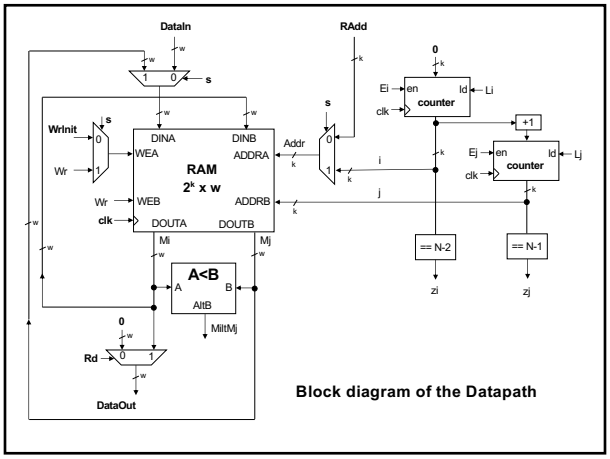
```
wait for s=1
for i=0 to N-2 do
    for j=i+1 to N-1 do
        if Mi < Mj then
            Swap Mi with Mj
        end if
    end for
end for
Done
wait for s=0
go to the beginning
```

29

ASM Chart



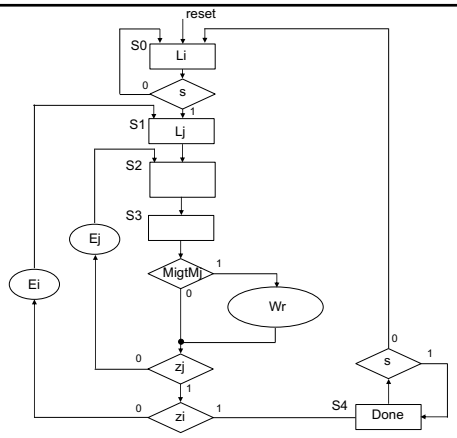
30



Block diagram of the Datapath

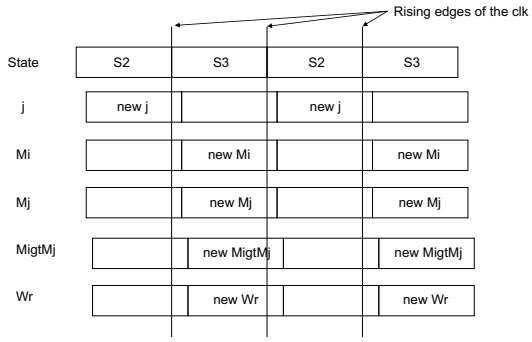
31

ASM Chart



32

Timing Analysis



33

Timing Analysis

Execution Time(N) =

1 + (N-1)·2 +

1 + (N-2)·2 +

….

1 + 1·2 =

= (N-1) + (N-1)(N/2) ·2 =

= (N-1) (N+1) =

= N^2 - 1

34