ECE 545
Lecture 12

Design of Controllers

Part 2

Required reading

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

Chapter 10, Finite State Machine: Principle & Practice



Class Exercise 3

CIPHER – Lightweight Architecture

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 $\oplus$ T) <<< U) + S[2i]
C = ((C $\oplus$ 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)

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

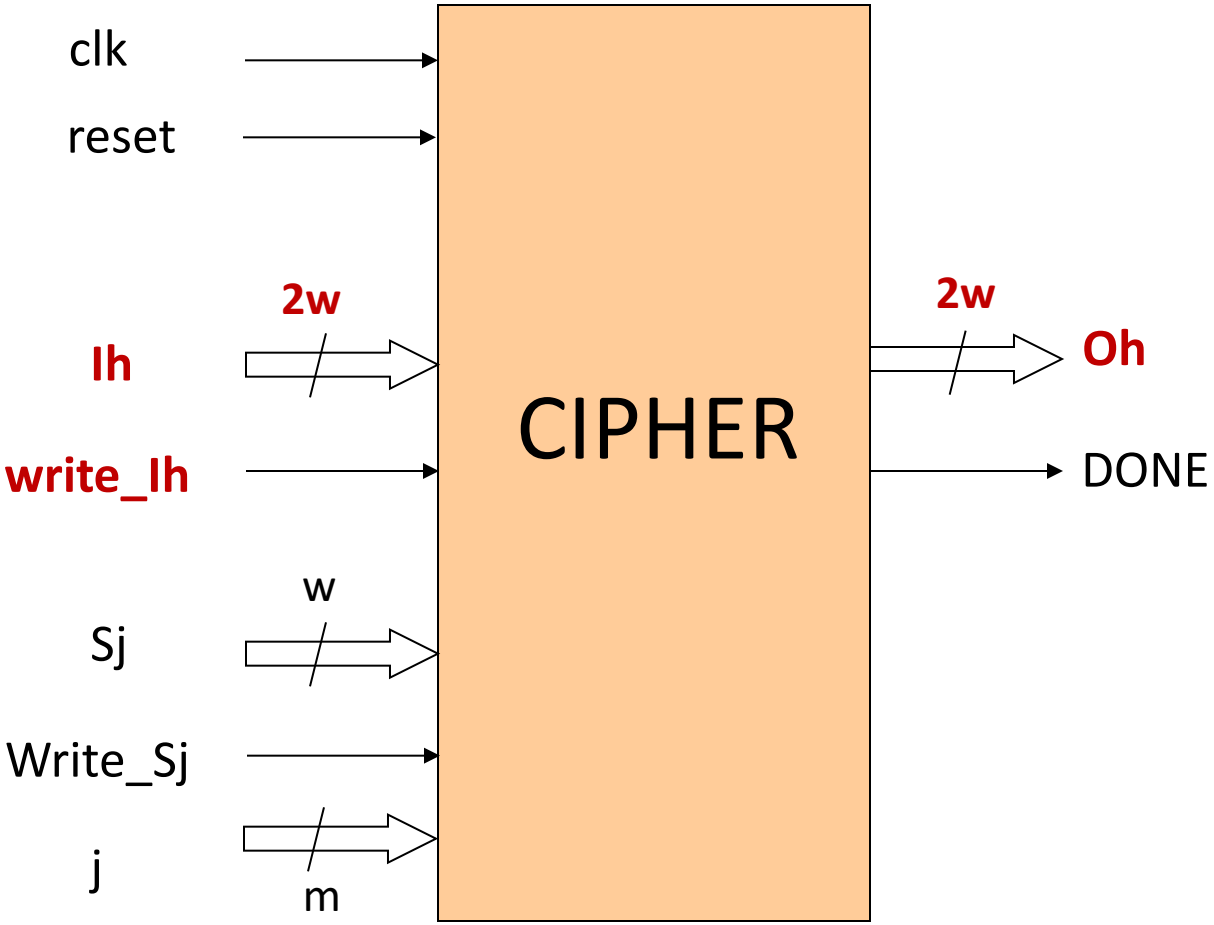
r: number of rounds (constant)

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.

Circuit Interface



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 $2^m - 1 \geq 2r+3$.

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.

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.

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.

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)
```

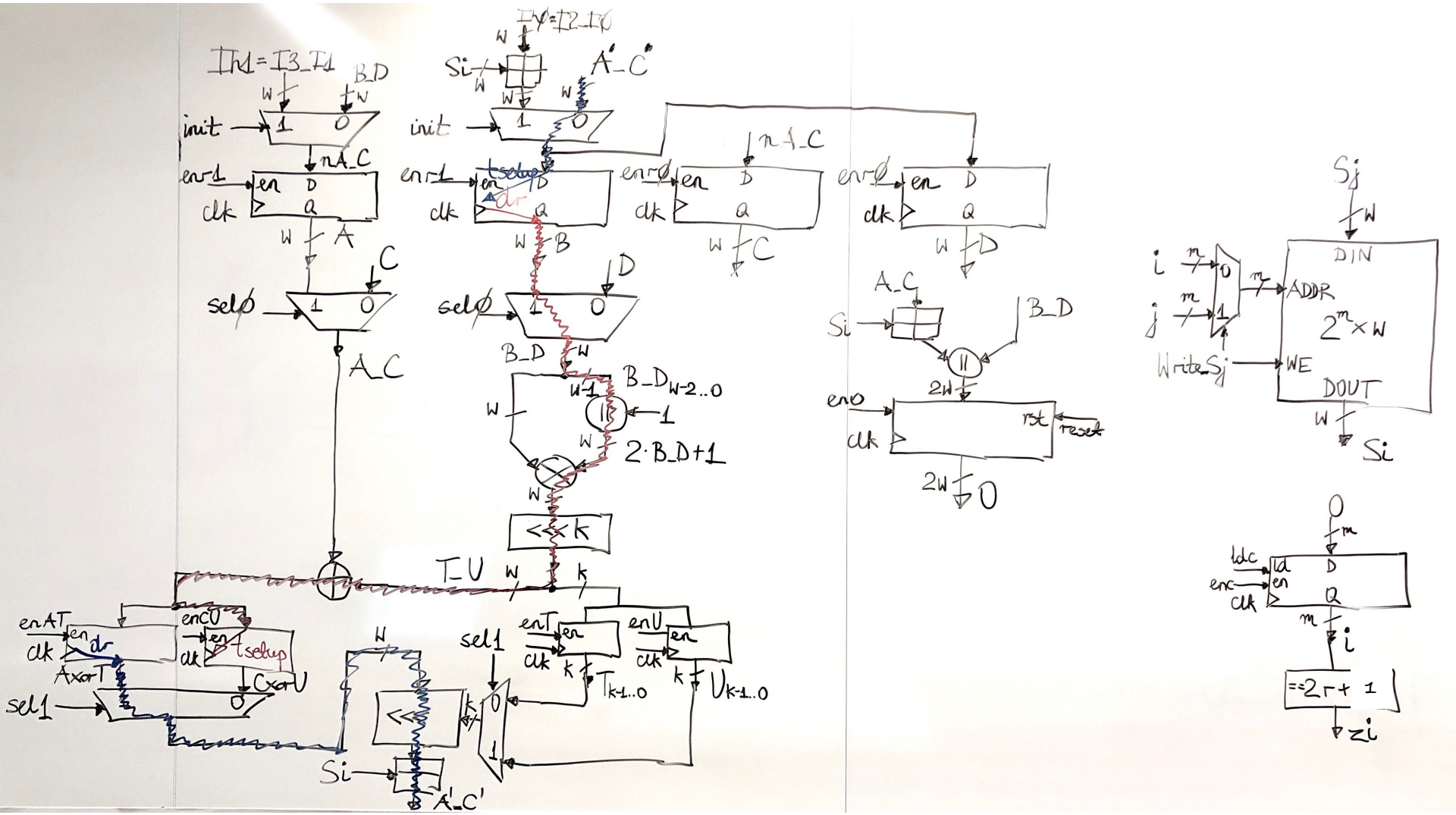
Pseudocode

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

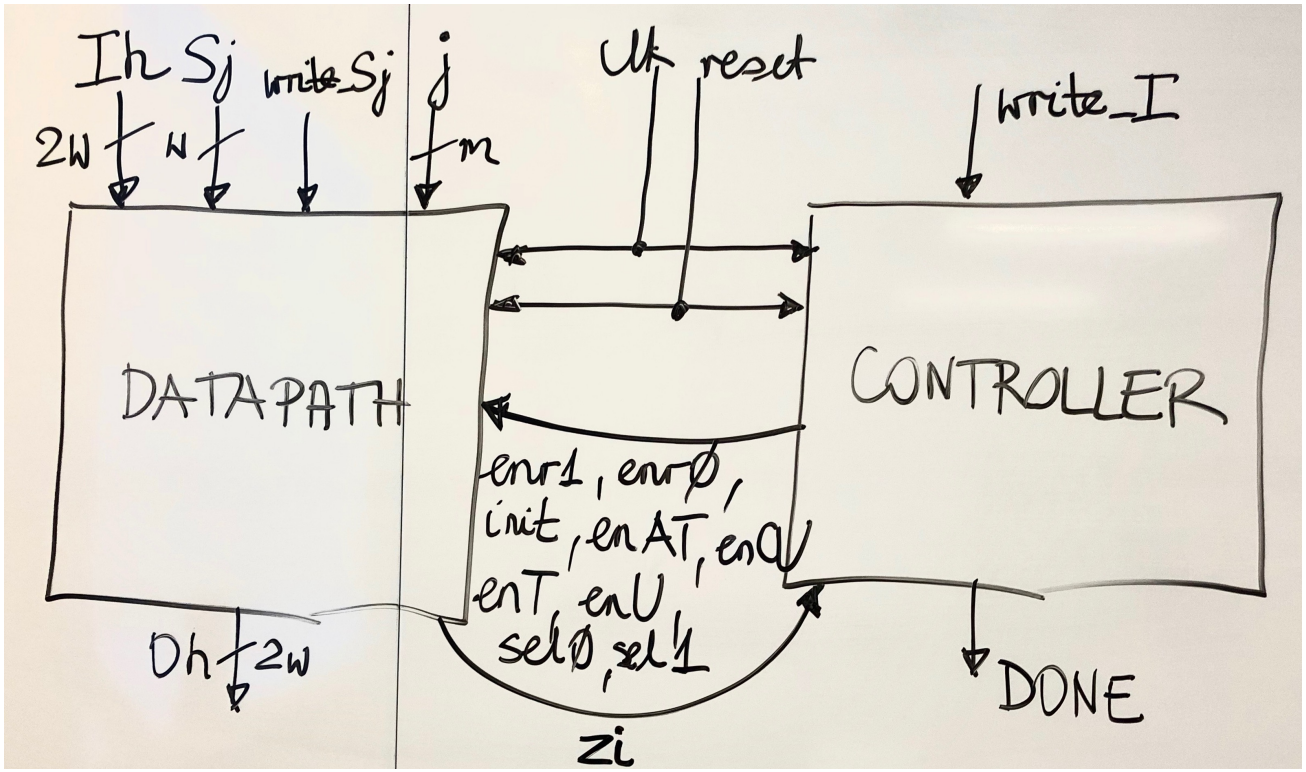
```
A = I3; B = I2; C = I1; D=I0
B = B + S[i=0]
D = D + S[i=1]
      for n = 1 to r do
      {
            T = (B*(2B + 1)) <<< k
            AxorT = A ⊕ T
            U = (D*(2D + 1)) <<< k
            CxorU = C ⊕ U
            A' = (AxorT <<< U) + S[i=2n]
           (C, D)  =  (D, A')
            C'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

Datapath



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

Interface with the Division into the Datapath and Controller

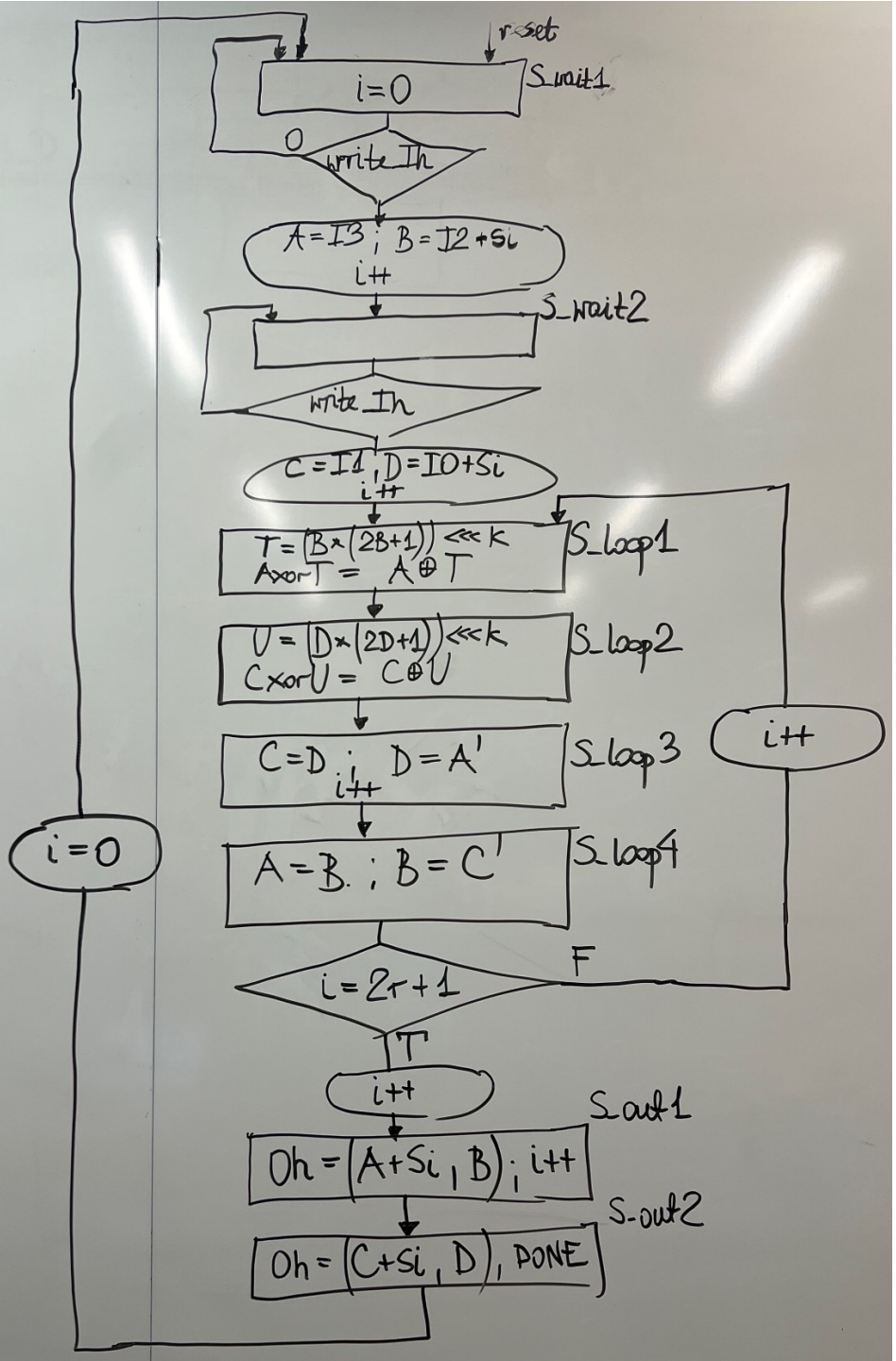


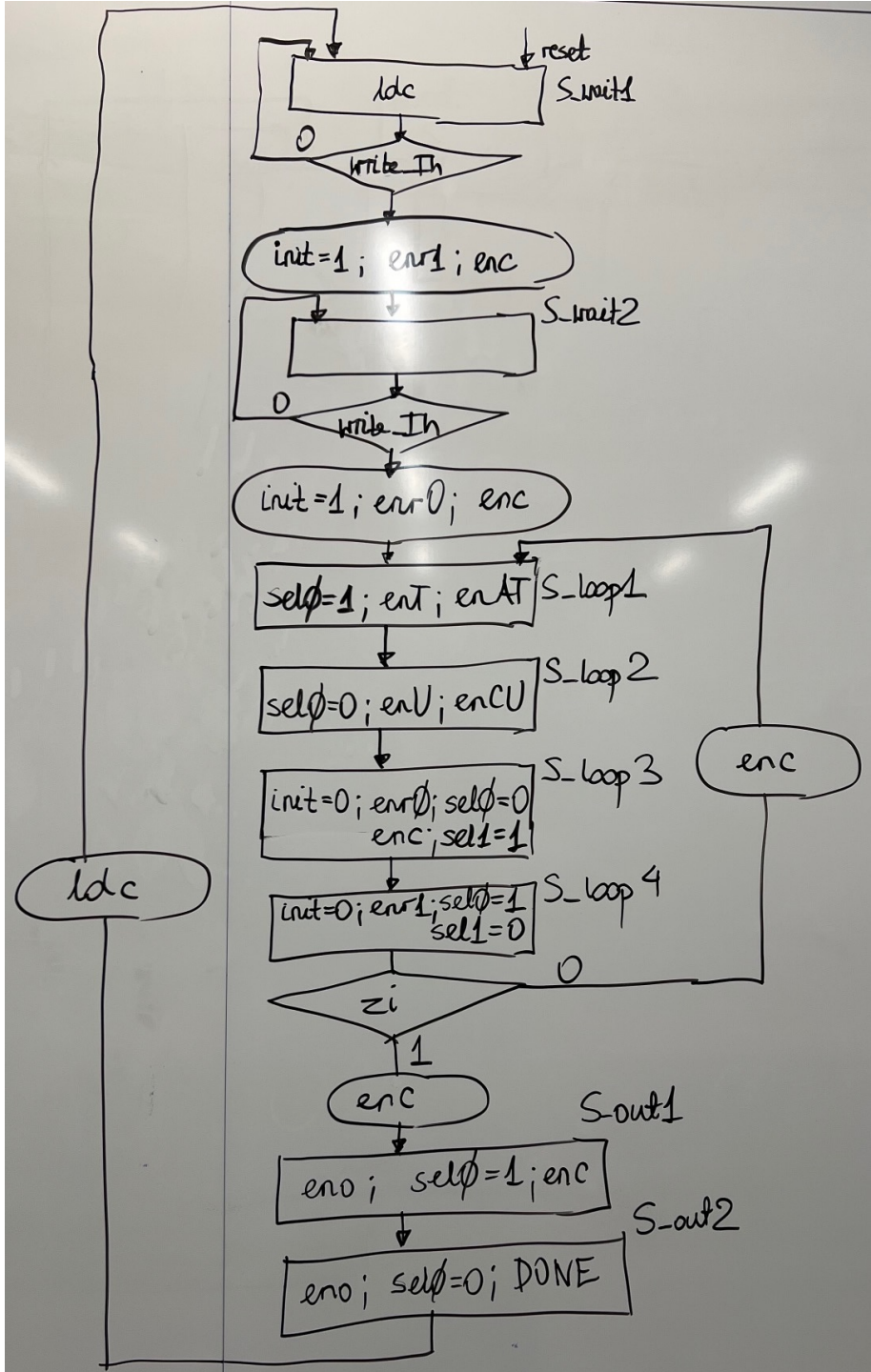
Pseudocode

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

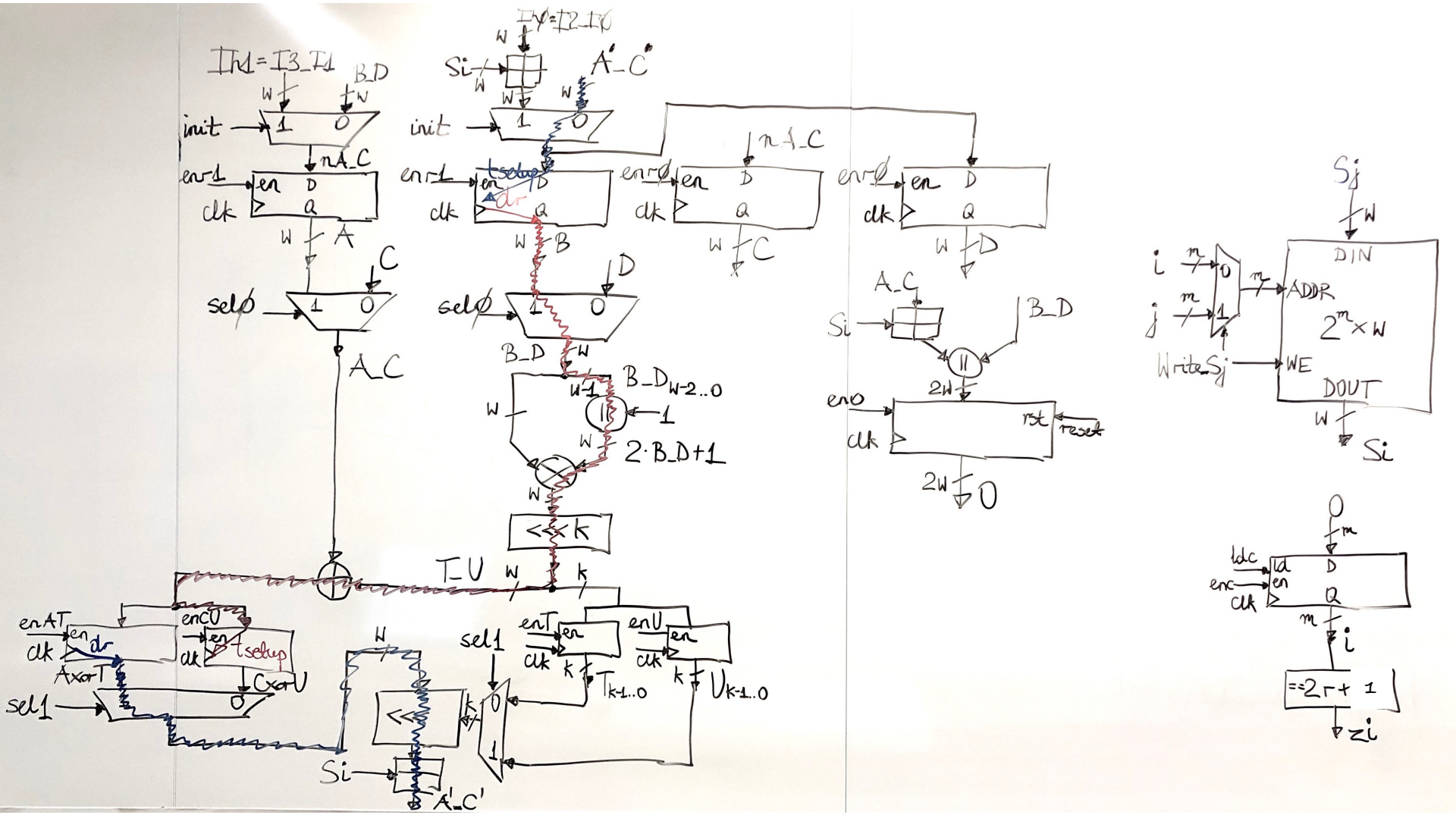
Split input I into four words, I3, I2, I1, I0, of the size of w bits each

```
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)
```


reset
i=0
S_wait1
0
write_Ih
A=I3 ; B=I2+Si
i++
S_wait2
write_Ih
C=I1 ; D=I0+Si
i++
T = (B × (2B+1)) <<< k
AxorT = A ⊕ T
S_loop1
V = (D × (2D+1)) <<< k
CxorV = C ⊕ V
S_loop2
C=D ; D=A'
i++
S_loop3
A=B ; B=C'
S_loop4
i=2r+1
F
T
i++
i++
i=0
Oh = (A+Si, B) ; i++
S_out1
Oh = (C+Si, D) ; DONE
S_out2


reset
ldc
S_wait1
0
write_Ih
init=1 ; enr1 ; enc
S_wait2
0
write_Ih
init=1 ; enr0 ; enc
selØ=1 ; enT ; enAT
S_loop1
selØ=0 ; enV ; enCV
S_loop2
init=0 ; enrØ ; selØ=0
enc ; sel1=1
S_loop3
init=0 ; enr1 ; selØ=1
sel1=0
S_loop4
zi
0
1
enc
enc
ldc
eno ; selØ=1 ; enc
S_out1
eno ; selØ=0 ; DONE
S_out2

Datapath



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

Timing Analysis – Key Setup & Encryption

Notation:

T_{CLK} – clock period in ns

N_M – number of encrypted message blocks I

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

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

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

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

Timing Analysis - Throughput

Notation:

T_{CLK} – clock period in ns

N_M – number of encrypted message blocks I

w – word size

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

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

Timing Analysis – Throughput Ratio

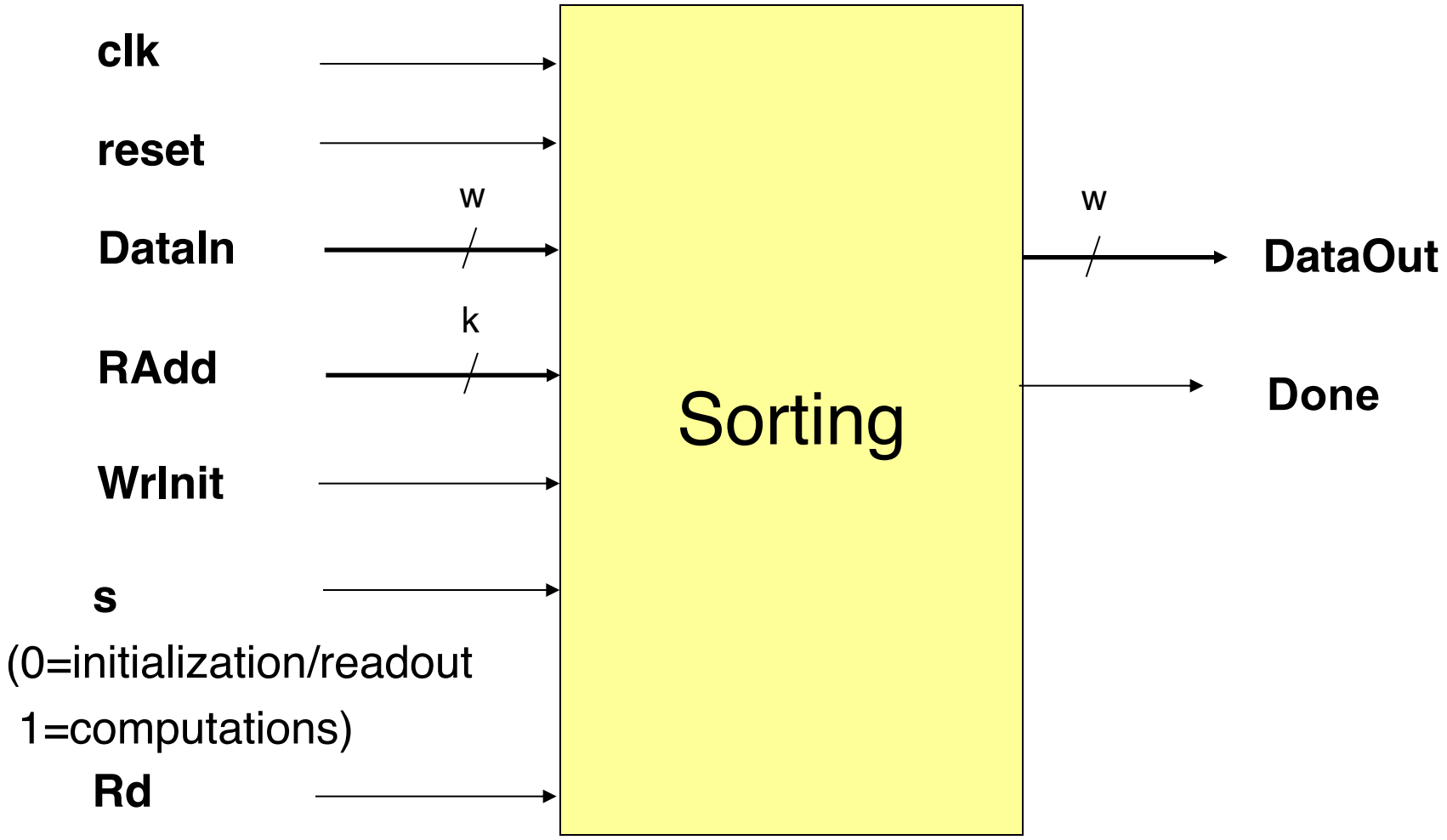
$$\frac{Thr}{Thr'} = \frac{\frac{4 \cdot W}{(r+2) \cdot T_{ak}}}{\frac{W}{(r+1)\, T_{ak}'}} = 4 \cdot \frac{r+1}{r+2} \cdot \frac{T_{ak}'}{T_{ak}} \approx 4 \cdot \frac{1}{2} = 2$$

Throughput decreases by approximately a factor of 2

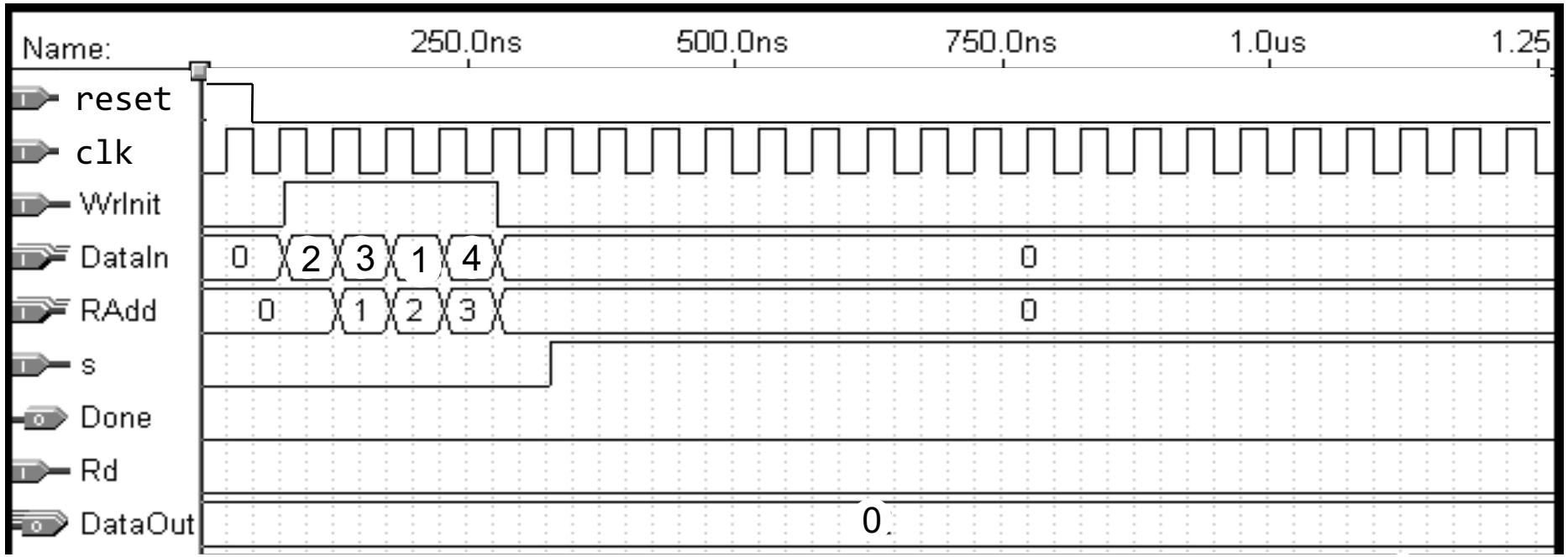


Class Exercise 4
Sorting

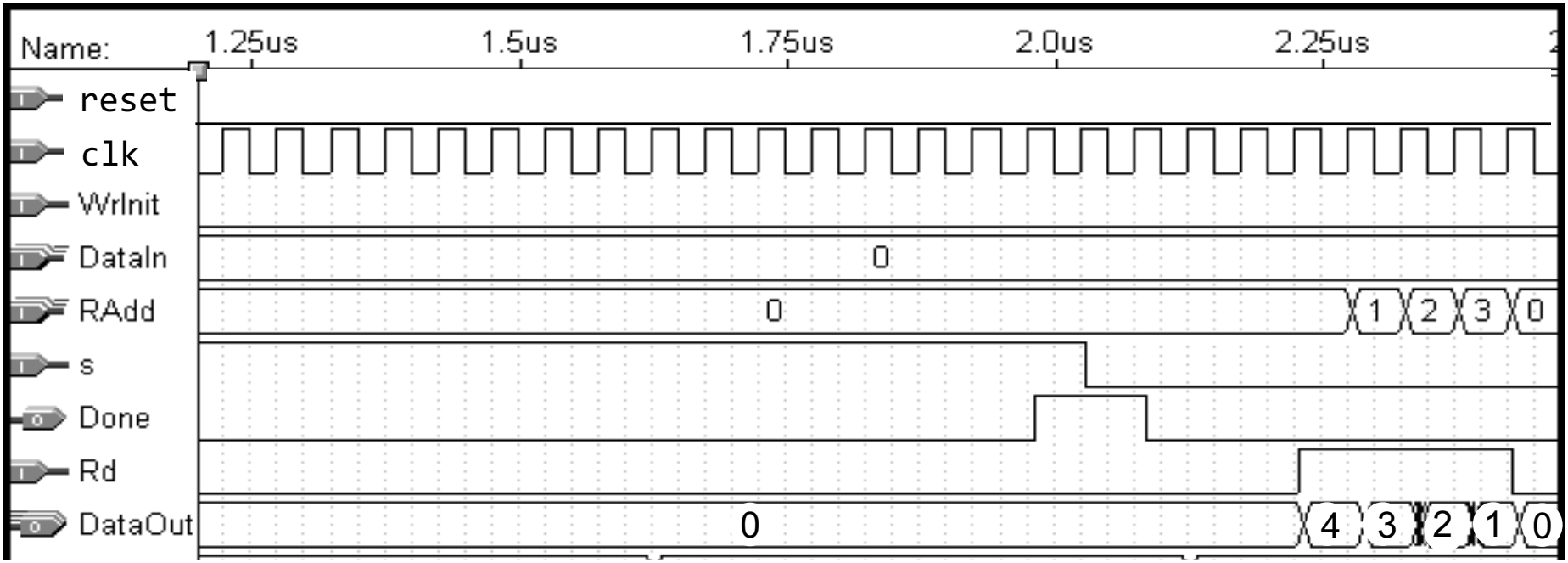
Sorting


clk
reset
w
DataIn
k
RAdd
WrInit
s
(0=initialization/readout
1=computations)
Rd
Sorting
w
DataOut
Done

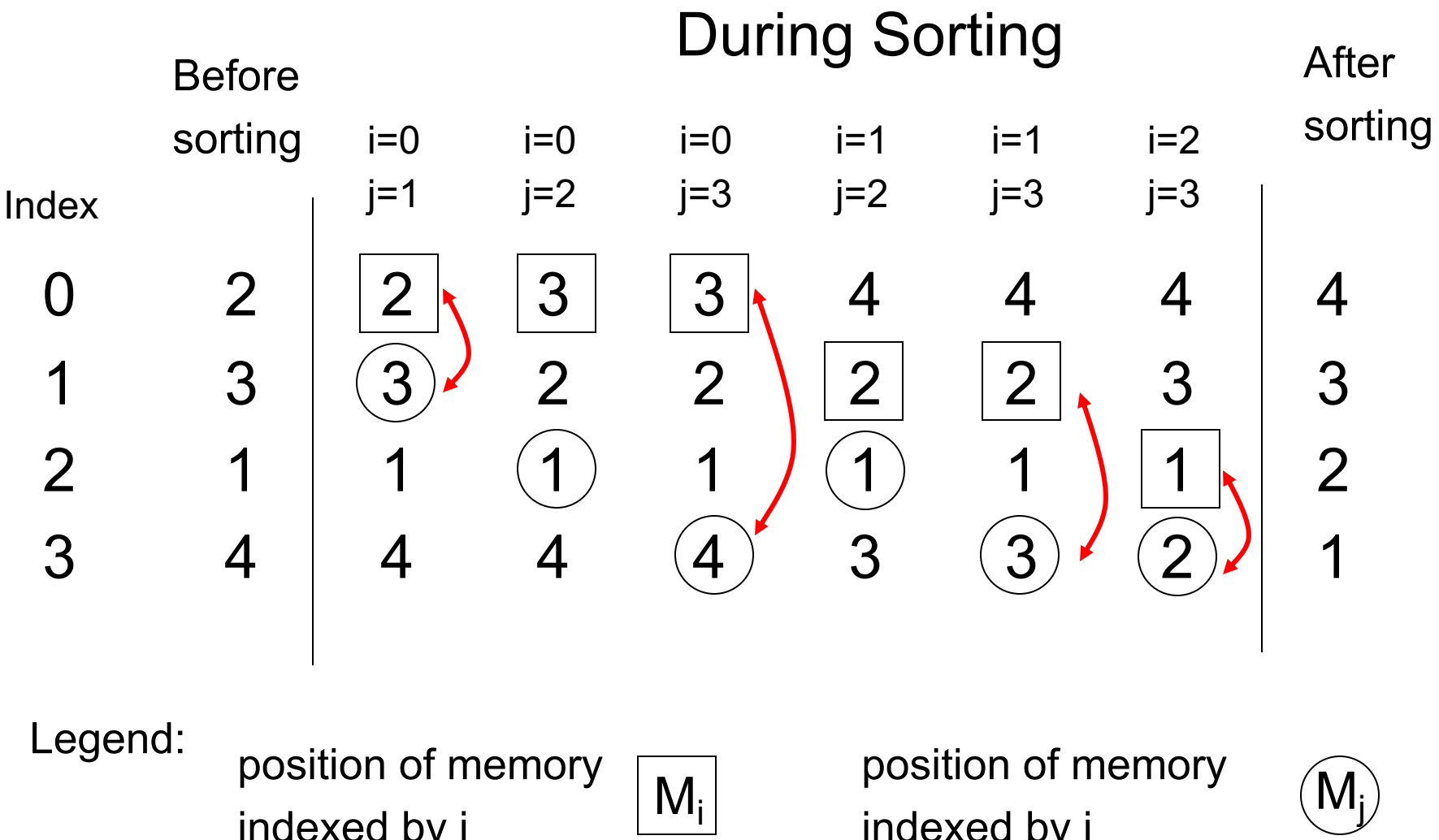
Simulation results for the sort operation (1)
Loading memory and starting sorting



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



Sorting – Example for N=4



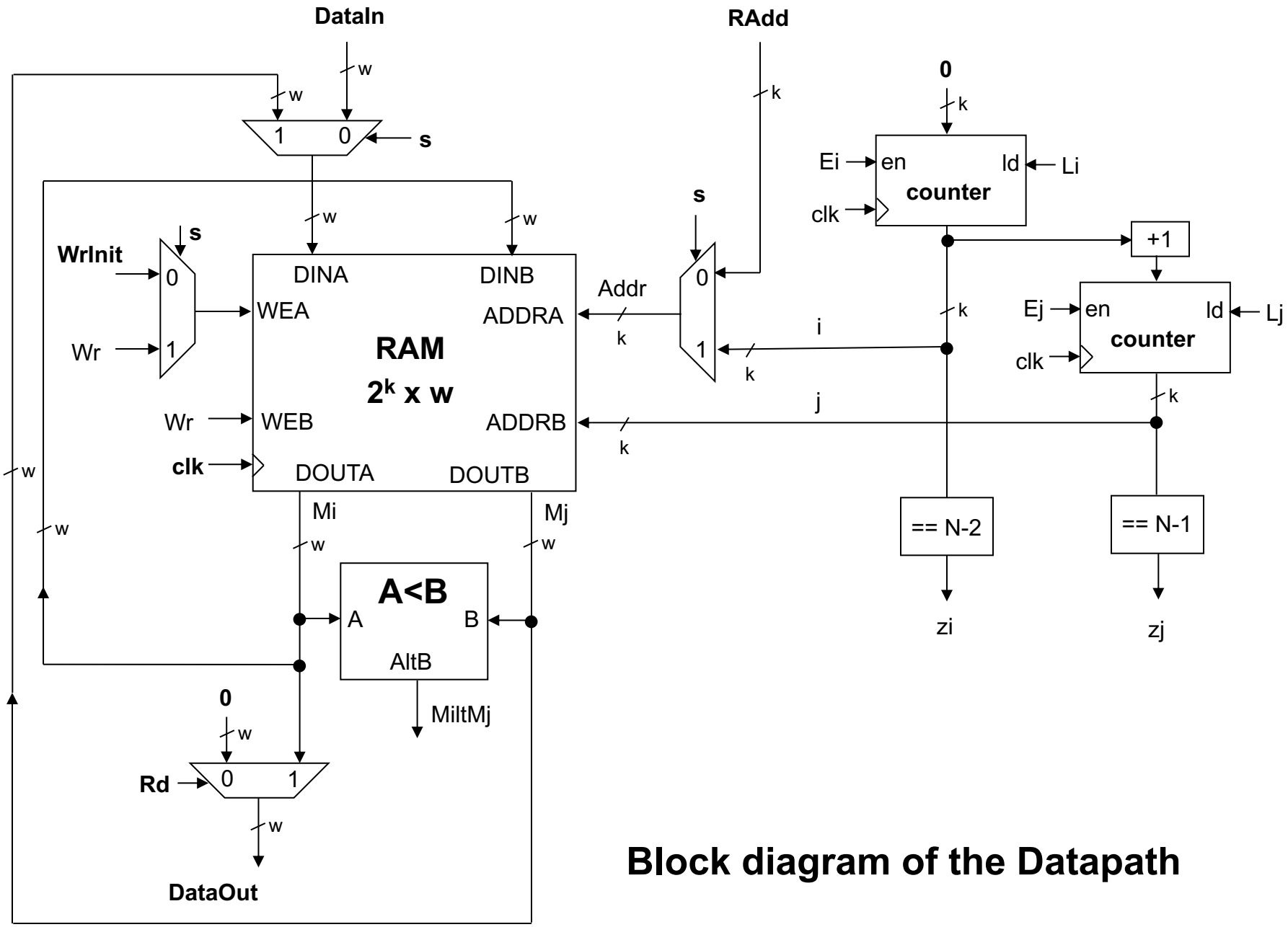
Legend:

position of memory indexed by i M_i

position of memory indexed by j M_j

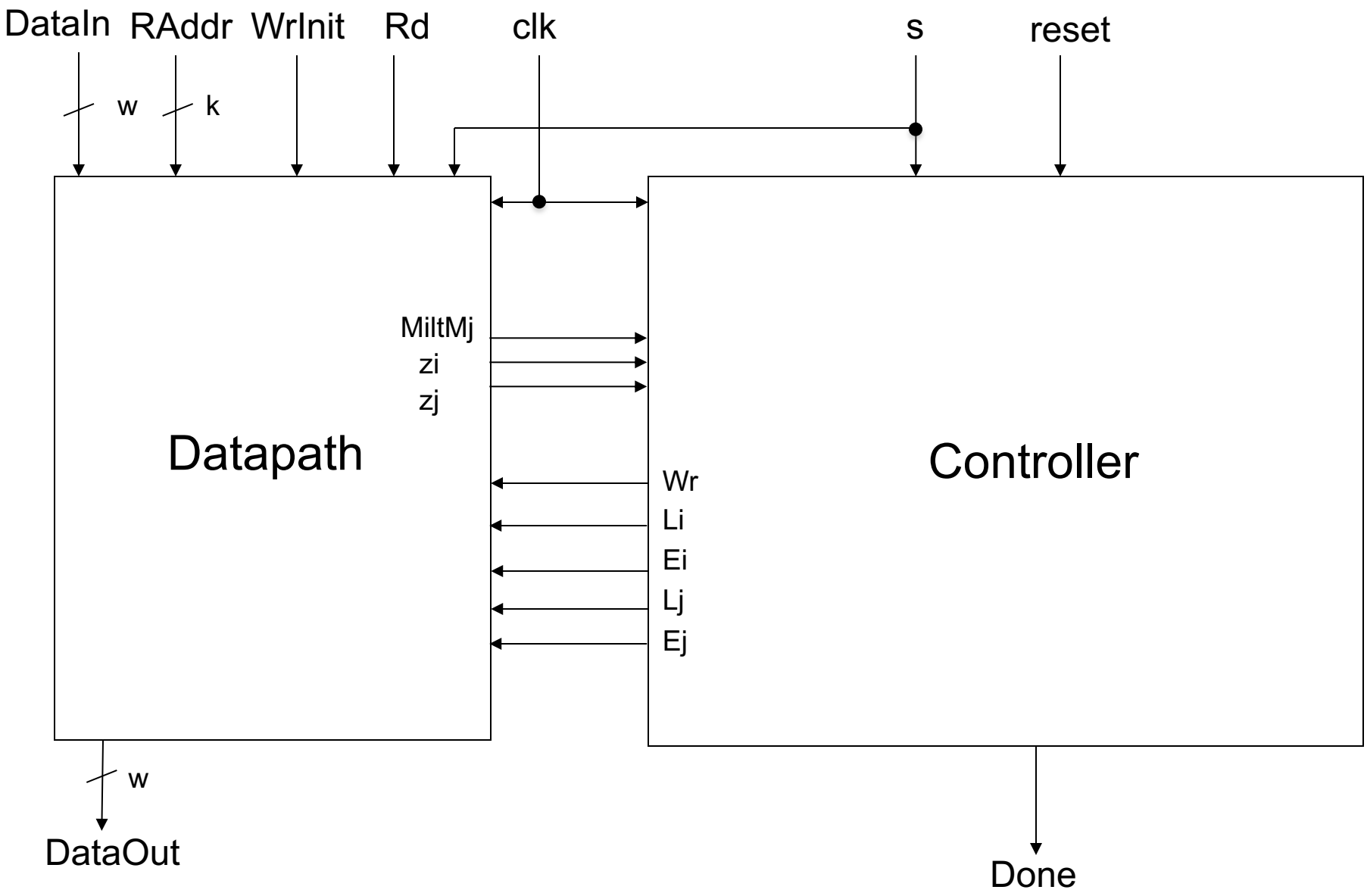
Pseudocode

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



Block diagram of the Datapath

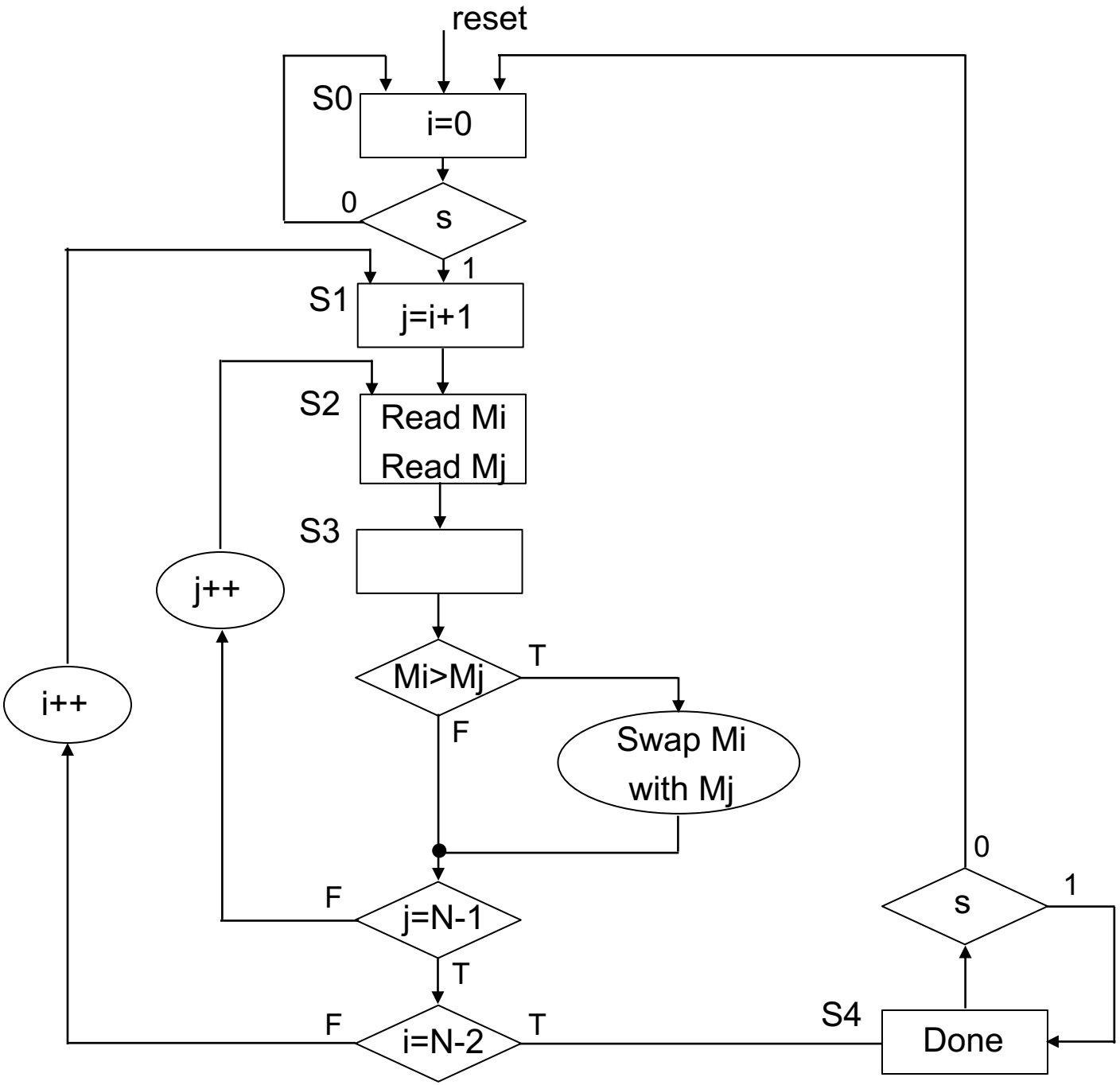
Interface with the division into the Datapath and Controller

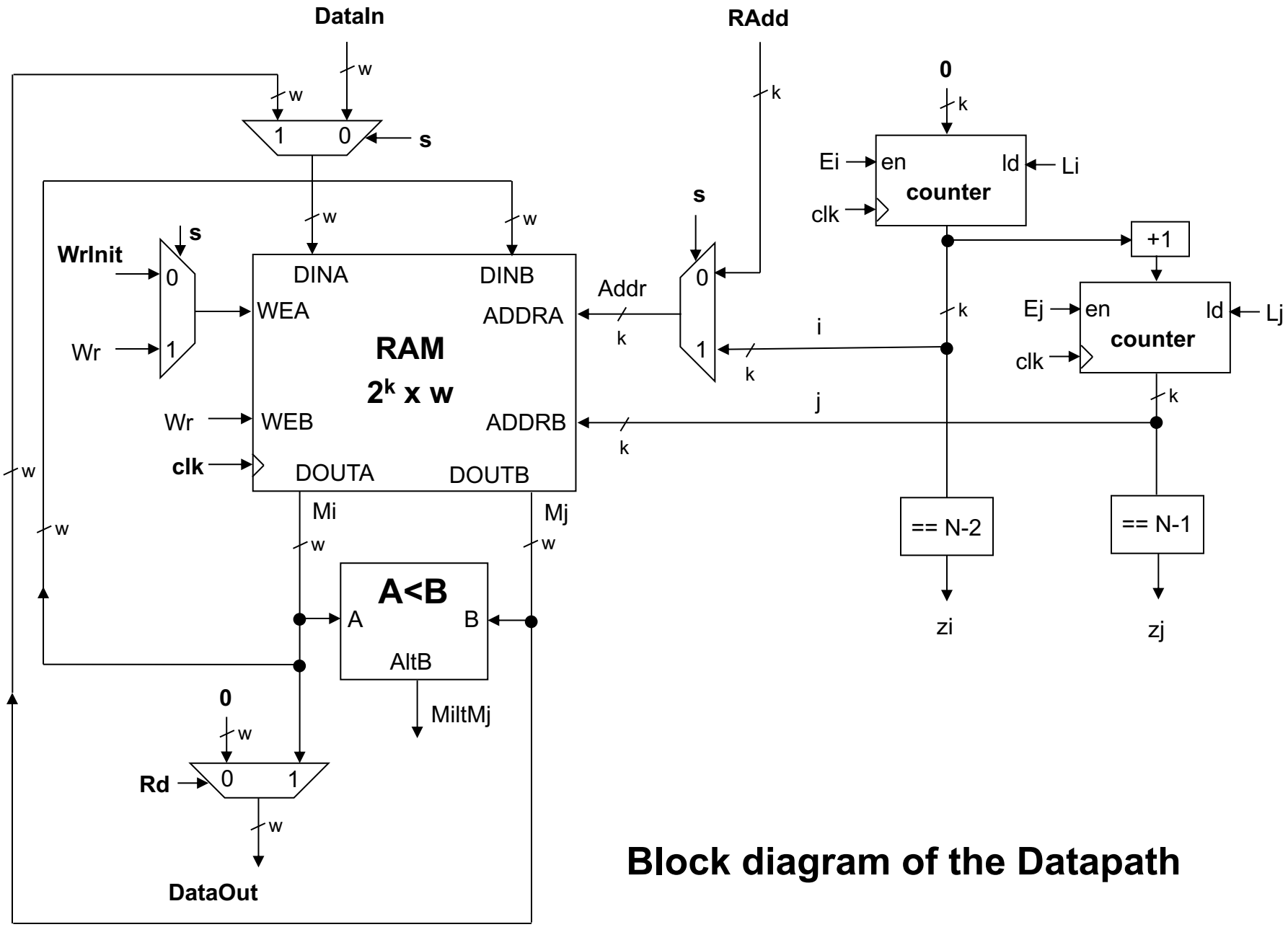


Pseudocode

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

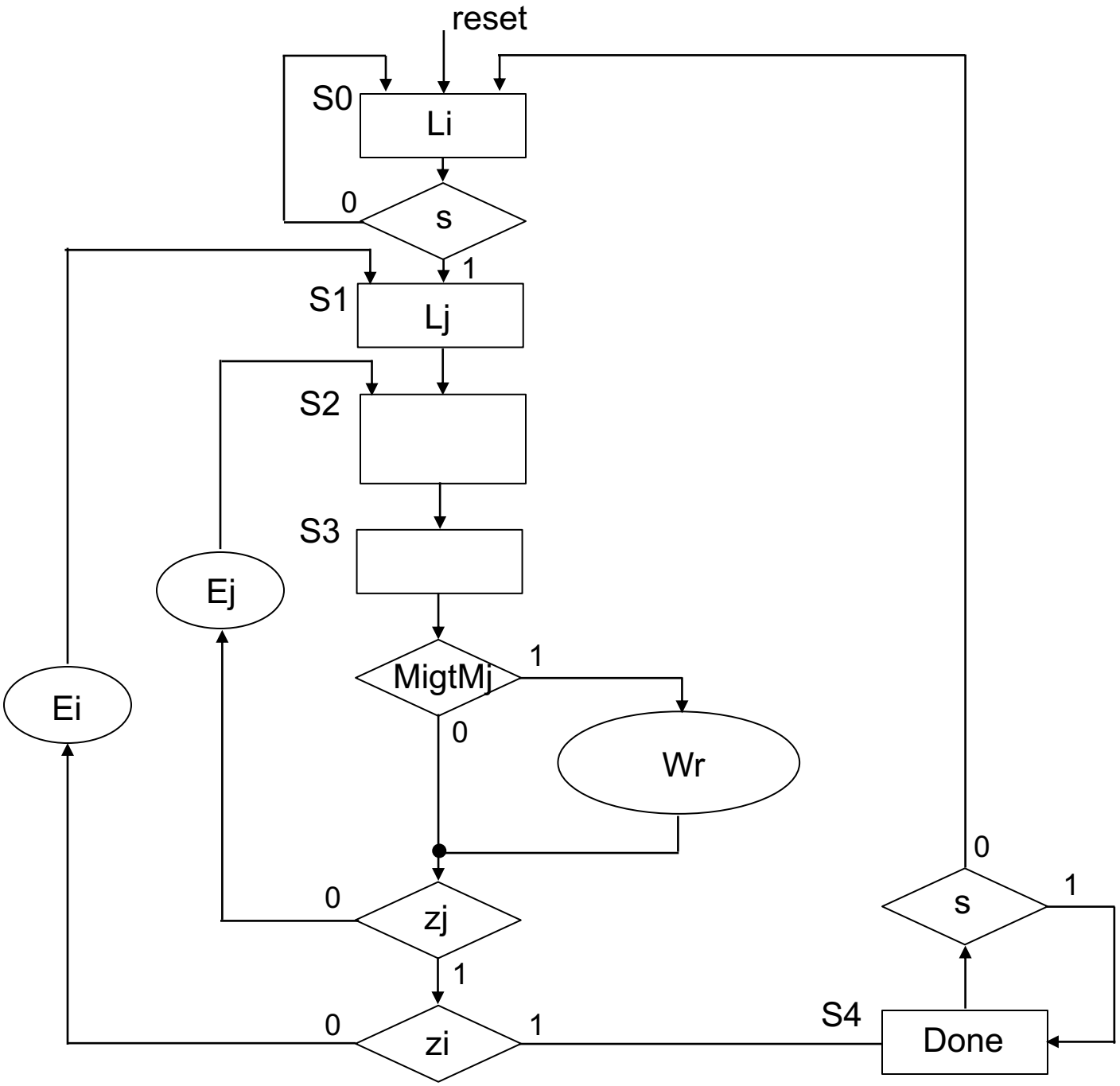
ASM Chart


reset
S0
i=0
0
s
1
S1
j=i+1
S2
Read Mi
Read Mj
S3
j++
Mi>Mj
T
F
Swap Mi
with Mj
i++
F
j=N-1
T
F
i=N-2
T
S4
Done
0
s
1

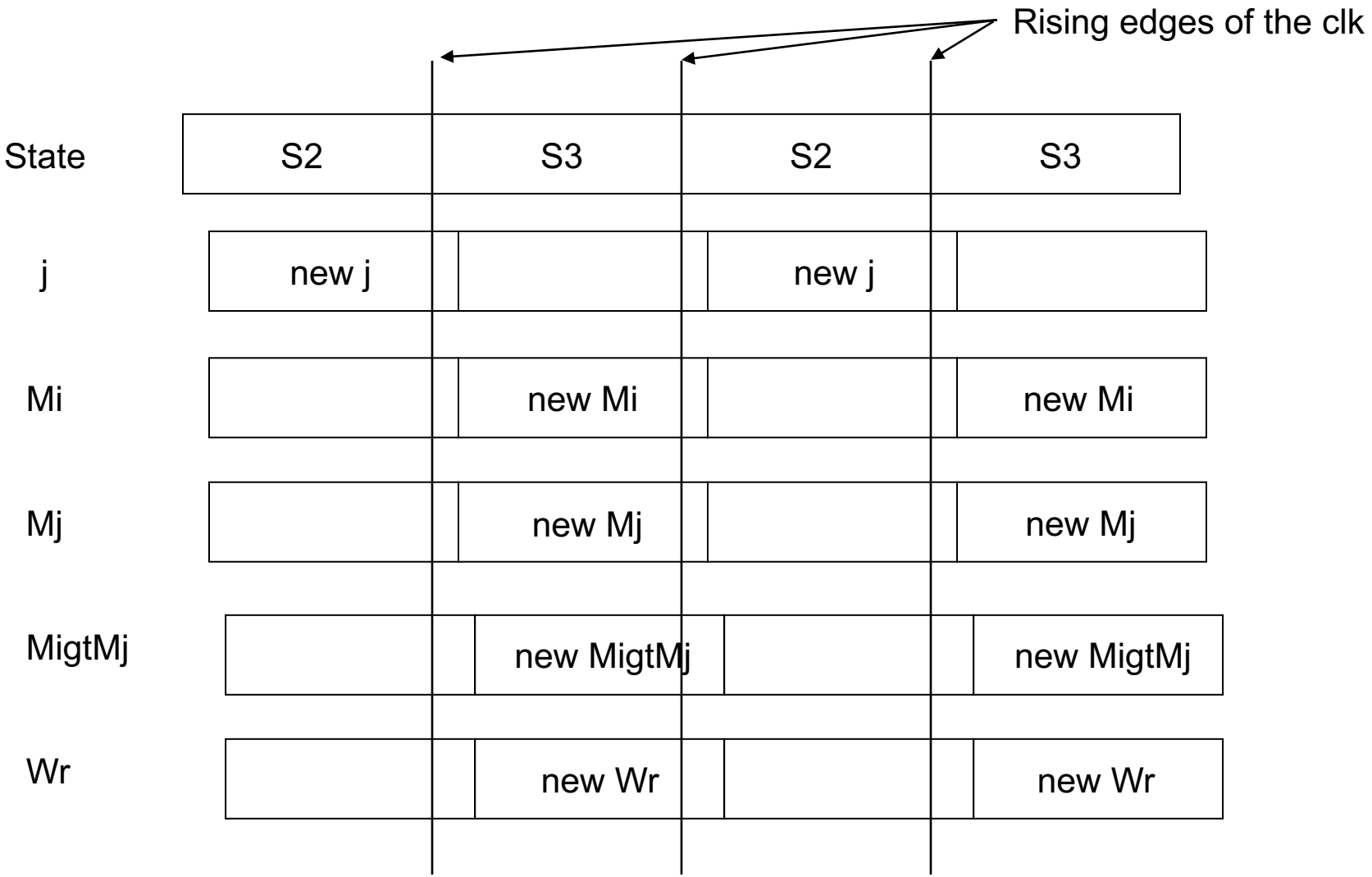


Block diagram of the Datapath

ASM Chart


reset
S0
Li
0
s
1
S1
Lj
S2
S3
Ej
MigtMj
1
Ei
0
Wr
0
zj
1
0
zi
1
S4
Done
0
s
1

Timing Analysis



Timing Analysis

Execution Time(N) =

$1 + (N-1)\cdot 2 +$

$1 + (N-2)\cdot 2 +$

….

$1 + 1\cdot 2 =$

$= (N-1) + (N-1)(N/2)\cdot 2 =$

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

$= N^2 - 1$