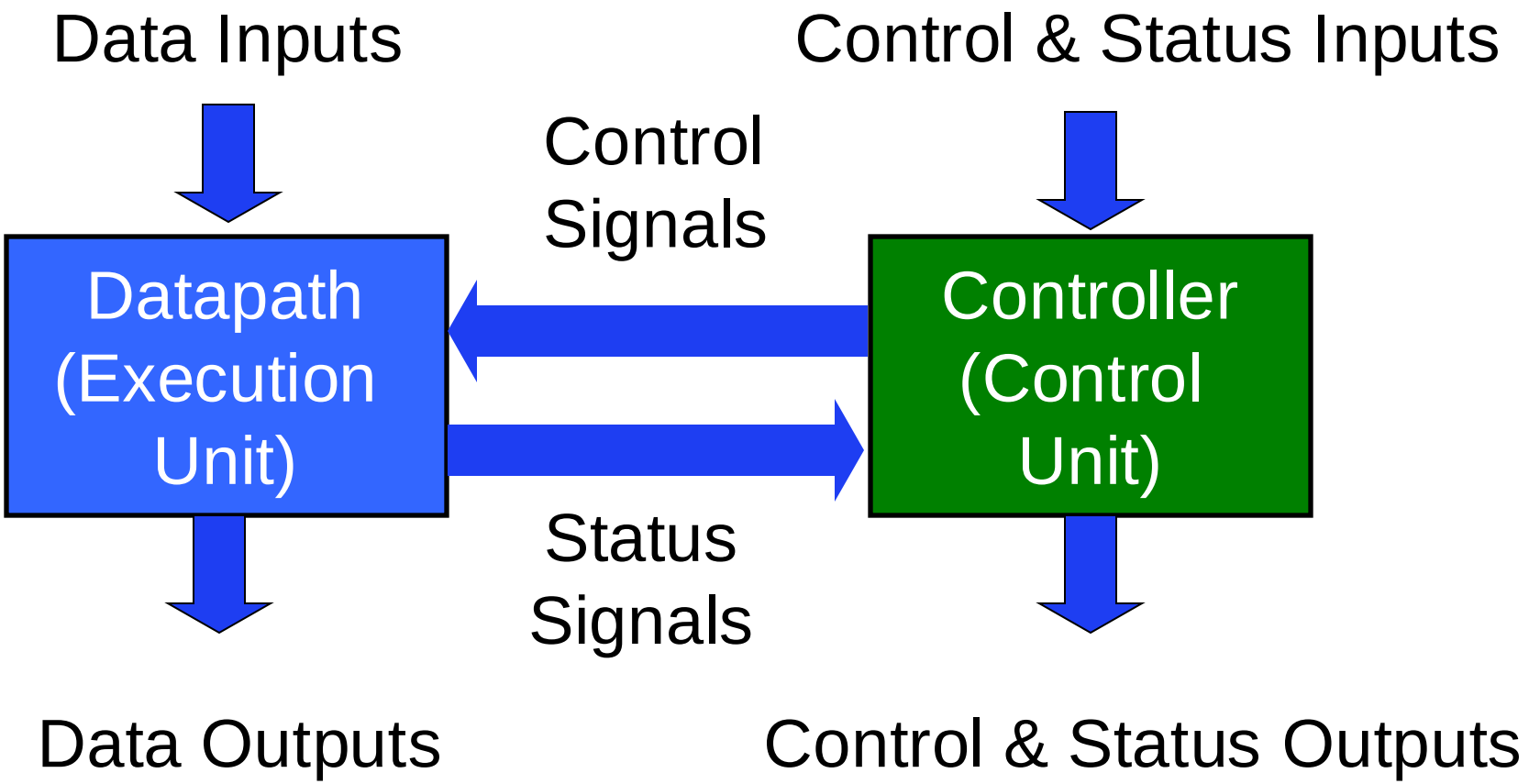
Lecture 4

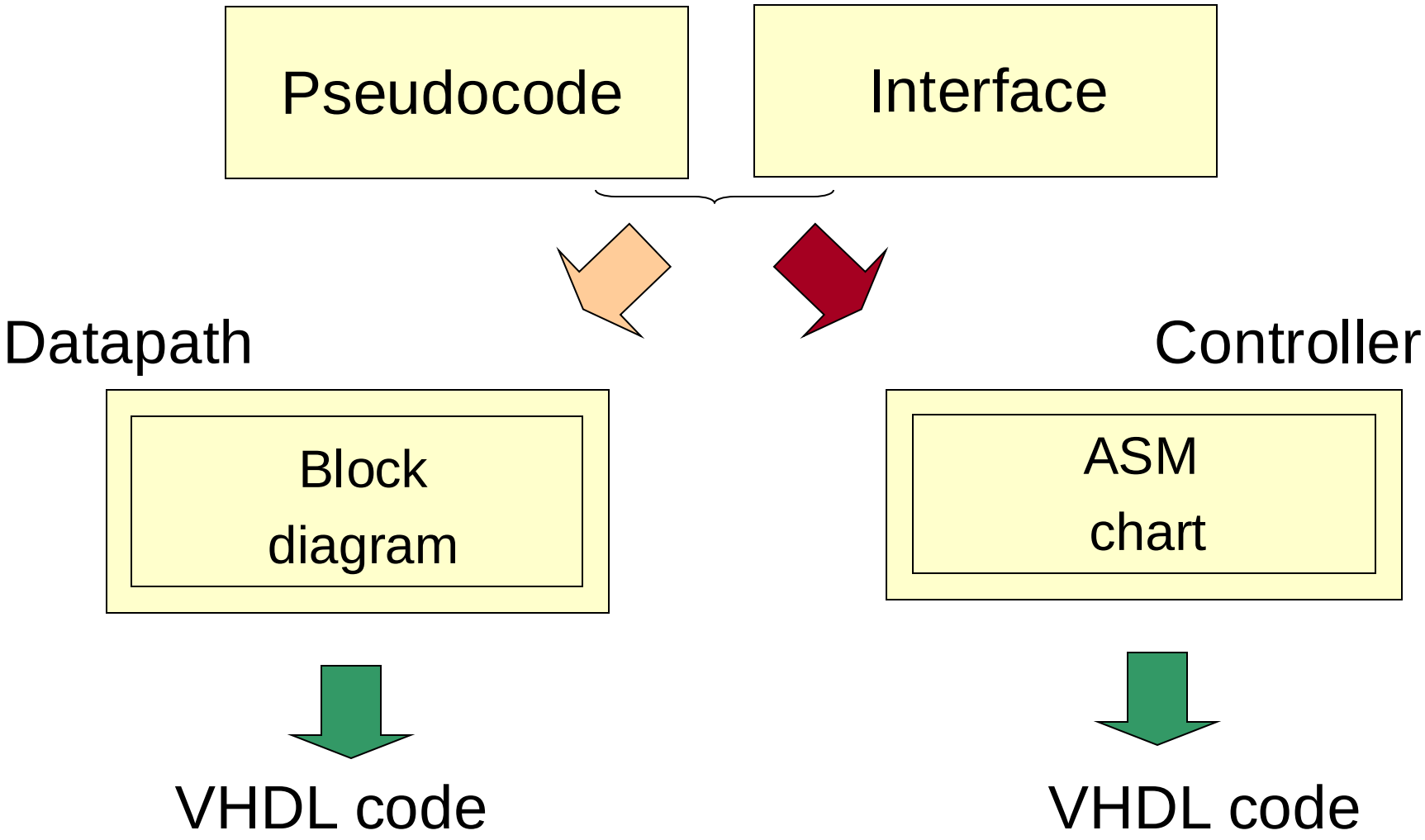
RTL Design Methodology
Part A

Transition from
the Pseudocode & Interface
to a Corresponding Block Diagram

Structure of a Typical Digital System



Hardware Design with RTL VHDL



Steps of the Design Process

1. Text description
2. Interface
3. Pseudocode
4. Block diagram of the Datapath
5. Interface divided into the Datapath and Controller
6. ASM chart of the Controller
7. RTL VHDL code of the Datapath, Controller, and Top-Level Unit
8. Testbench for the Datapath, Controller, and Top-Level Unit
9. Functional simulation and debugging
10. Synthesis and post-synthesis simulation
11. Implementation and timing simulation
12. Experimental testing using FPGA board

Steps of the Design Process
Introduced in Class Today

1. Text description
2. Interface
3. Pseudocode
4. **Block diagram of the Datapath**
5. **Interface divided into the Datapath and Controller**
6. ASM chart of the Controller
7. RTL VHDL code of the Datapath, Controller, and Top-level Unit
8. Testbench for the Datapath, Controller, and Top-Level Unit
9. Functional simulation and debugging
10. Synthesis and post-synthesis simulation
11. Implementation and timing simulation
12. Experimental testing using FPGA board

Class Exercise 1
Statistics

Text Description

Draw a block diagram of the datapath of the STATISTICS circuit capable of computing the first three largest numbers in the set of $k=2^m$ n-bit unsigned numbers provided at its input.

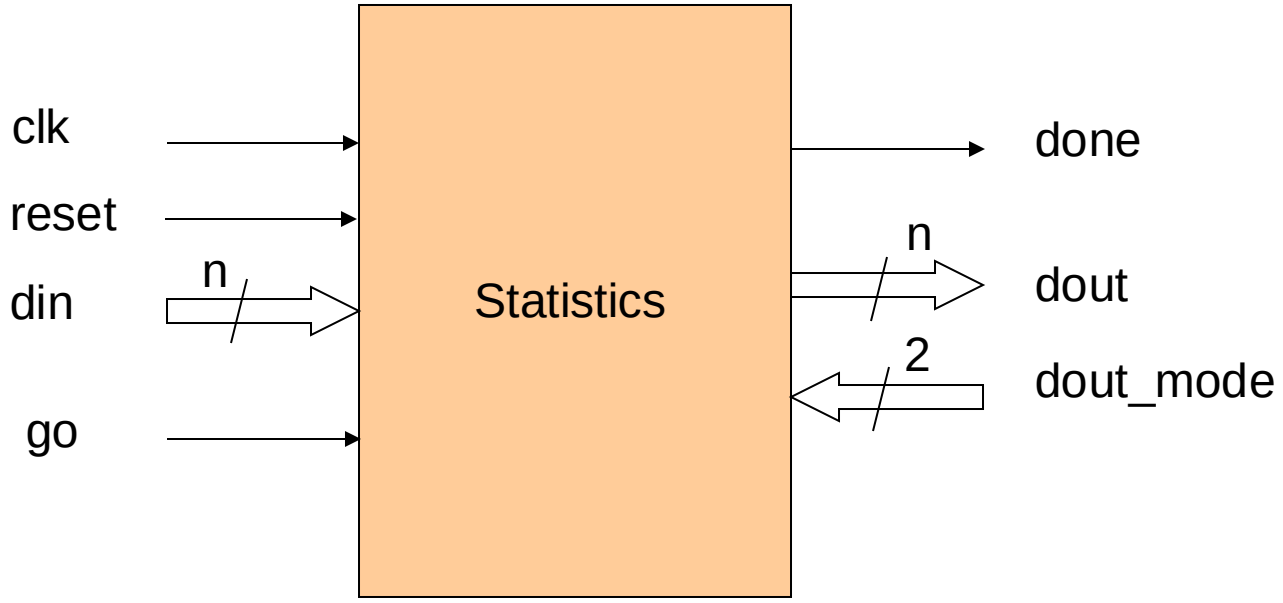
In parallel, the circuit should also compute an average of all k inputs.

The circuit should be optimized for minimum latency (i.e., execute as many operations as possible in parallel), and should take as little resources as possible.

Pseudocode

```
no_1 = no_2 = no_3 = sum = 0
for i=0 to k-1 do
    sum = sum + din
    if  (din > no_1) then
            no_3 = no_2
            no_2 = no_1
            no_1 = din
    elseif (din > no_2) then
            no_3 = no_2
            no_2 = din
    elseif (din > no_3) then
            no_3 = din
    end if
end for
avr = sum / k
```

Circuit Interface



Interface Table

Port	Width	Meaning
clk	1	System clock.
reset	1	System reset.
din	n	Input Data.
go	1	Control signal indicating that the first input is ready. Active for one clock cycle.
done	1	Signal set to high after the output is ready.
dout	n	Output dependent on the dout_mode input.
dout_mode	2	Control signal determining value available at the output. 00: avr, 01: no_1, 10: no_2, 11: no_3.

Pseudocode

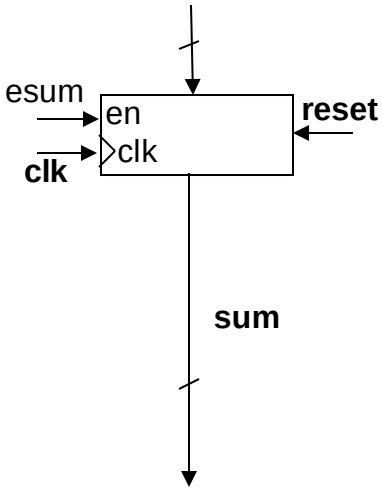
$k=2^m$

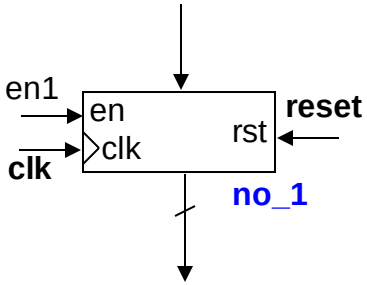
```
no_1 = no_2 = no_3 = sum = 0
wait for go
for i=0 to k-1 do
    sum = sum + din
    if  (din > no_1) then
            no_3 = no_2
            no_2 = no_1
            no_1 = din
    elseif (din > no_2) then
            no_3 = no_2
            no_2 = din
    elseif (din > no_3) then
            no_3 = din
    end if
end for
avr = sum / k
```

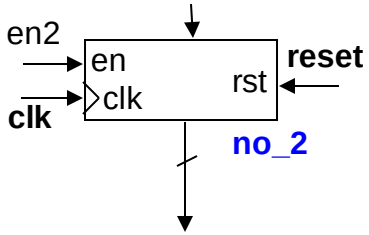
Pseudocode

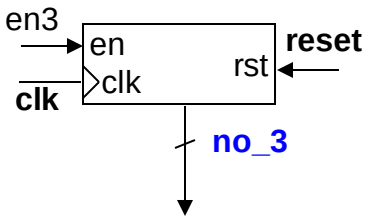
$k=2^m$

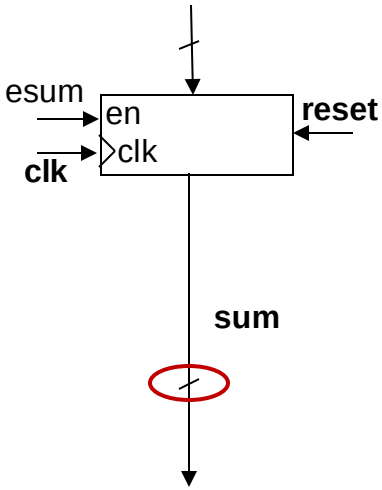
```
no_1 = no_2 = no_3 = sum = 0
wait for go
for i=0 to k-1 do
    sum = sum + din
    if  (din > no_1) then
            no_3 = no_2
            no_2 = no_1
            no_1 = din
    elseif (din > no_2) then
            no_3 = no_2
            no_2 = din
    elseif (din > no_3) then
            no_3 = din
    end if
end for
avr = sum / k
```

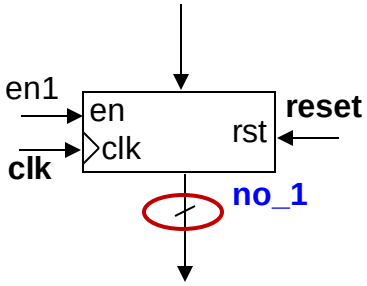

esum
en
clk
clk
reset
sum

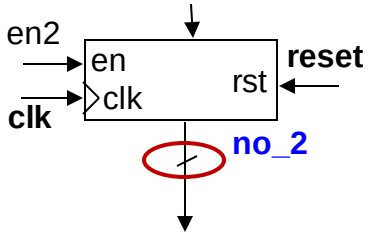

en1
en
clk
clk
rst
reset
no_1

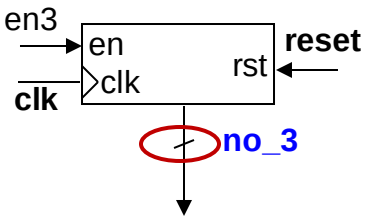

en2
en
clk
clk
rst
reset
no_2


en3
en
clk
clk
rst
reset
no_3

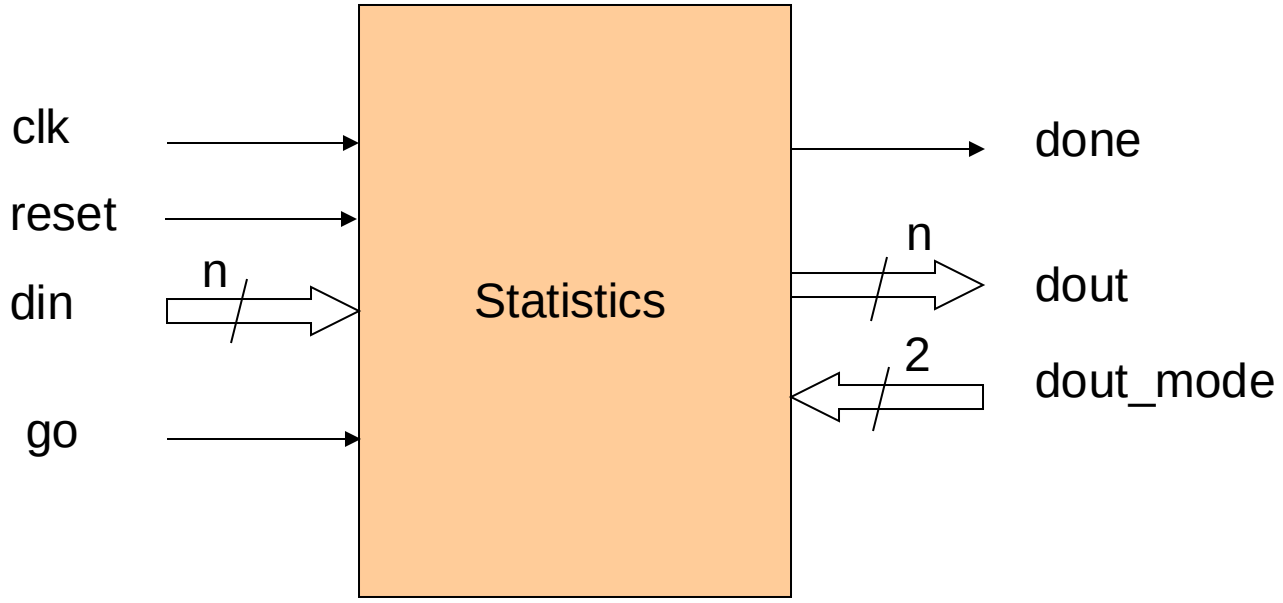

esum
en
clk
clk
reset
sum


en1
en
clk
clk
rst
reset
no_1


en2
en
clk
clk
rst
reset
no_2


en3
en
clk
clk
rst
reset
no_3

Circuit Interface



Size of the sum register

$k = 2^m$

$k = 2^3 = 8$

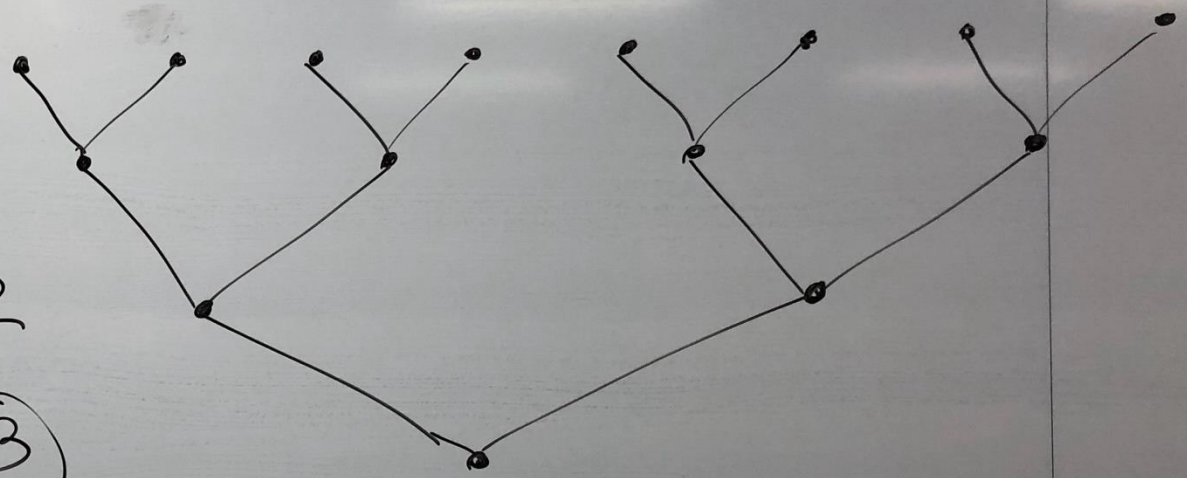
n-bit numbers

n

$n+1$

$n+2$

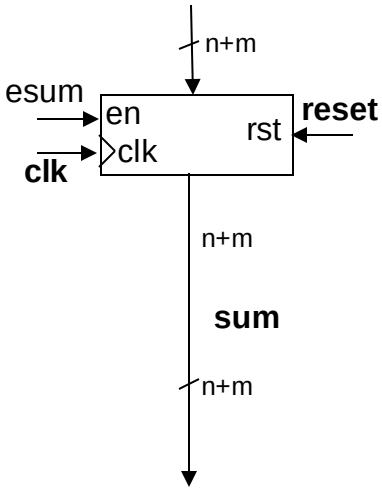
$n+3$

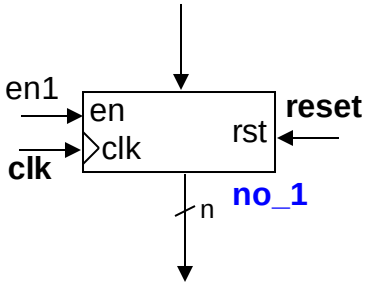


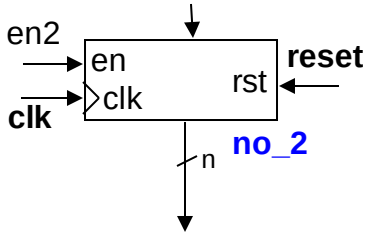
$n+m = n+\lceil \log_2 k \rceil$

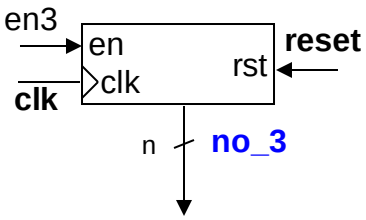
$i = 0 .. k-1$

$0 .. 2^m - 1$


n+m
esum
en
clk
rst
reset
clk
n+m
sum
n+m


en1
en
clk
rst
reset
clk
n
no_1

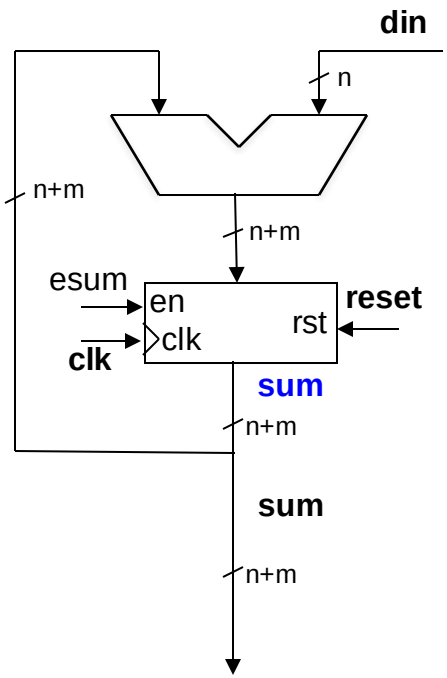

en2
en
clk
rst
reset
clk
n
no_2

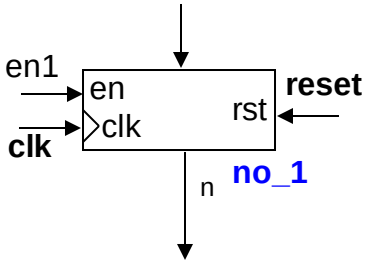

en3
en
clk
rst
reset
clk
n
no_3

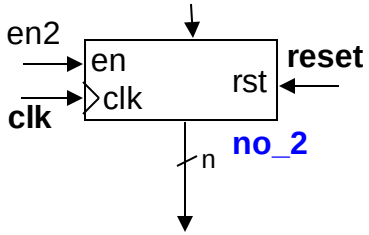
Pseudocode

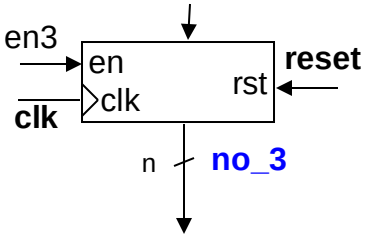
$k=2^m$

```
no_1 = no_2 = no_3 = sum = 0
wait for go
for i=0 to k-1 do
    sum = sum + din
    if  (din > no_1) then
            no_3 = no_2
            no_2 = no_1
            no_1 = din
    elseif (din > no_2) then
            no_3 = no_2
            no_2 = din
    elseif (din > no_3) then
            no_3 = din
    end if
end for
avr = sum / k
```


din
n
n+m
n+m
esum
en
clk
rst
reset
clk
sum
n+m
sum
n+m


en1
en
clk
rst
reset
clk
n
no_1

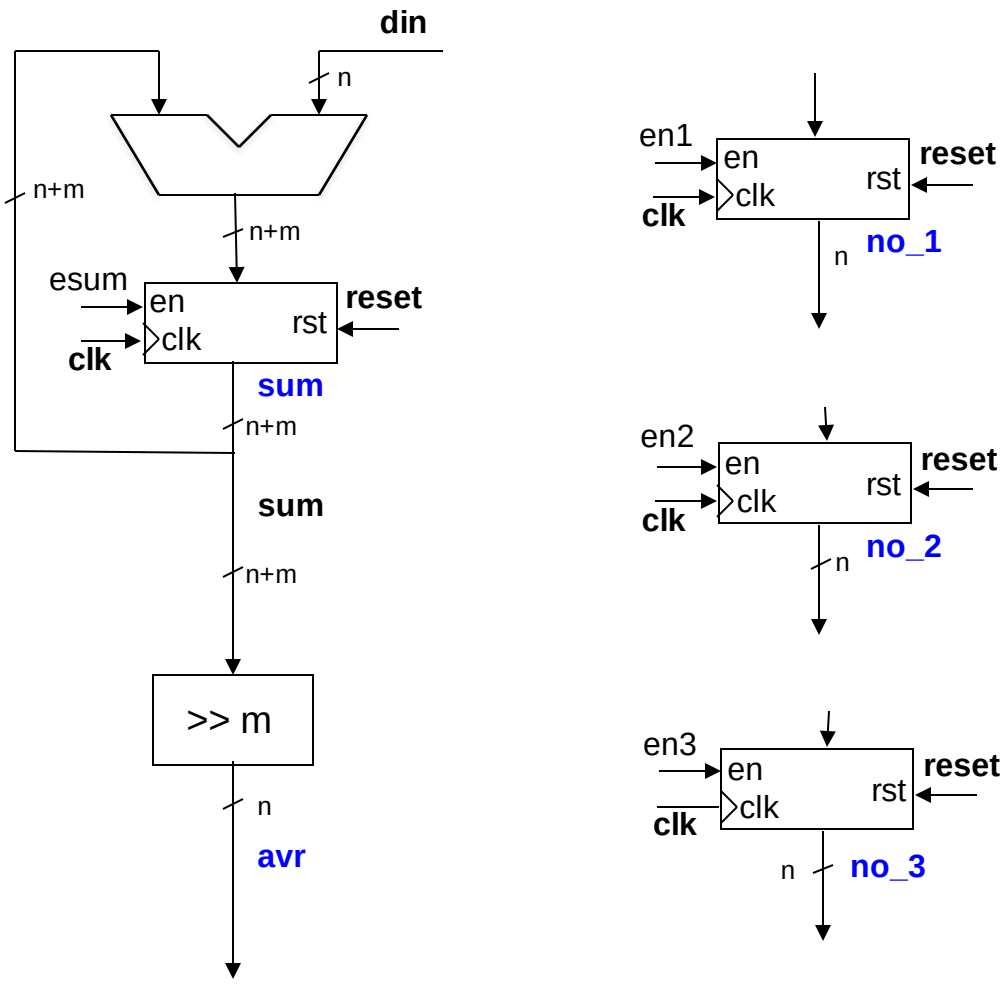

en2
en
clk
rst
reset
clk
n
no_2


en3
en
clk
rst
reset
clk
n
no_3

Pseudocode

$k=2^m$

```
no_1 = no_2 = no_3 = sum = 0
wait for go
for i=0 to k-1 do
    sum = sum + din
    if  (din > no_1) then
            no_3 = no_2
            no_2 = no_1
            no_1 = din
    elseif (din > no_2) then
            no_3 = no_2
            no_2 = din
    elseif (din > no_3) then
            no_3 = din
    end if
end for
avr = sum / k
```


din
n
n+m
n+m
esum
en
clk
rst
reset
clk
sum
n+m
sum
n+m
>> m
n
avr

en1
en
clk
rst
reset
clk
n
no_1

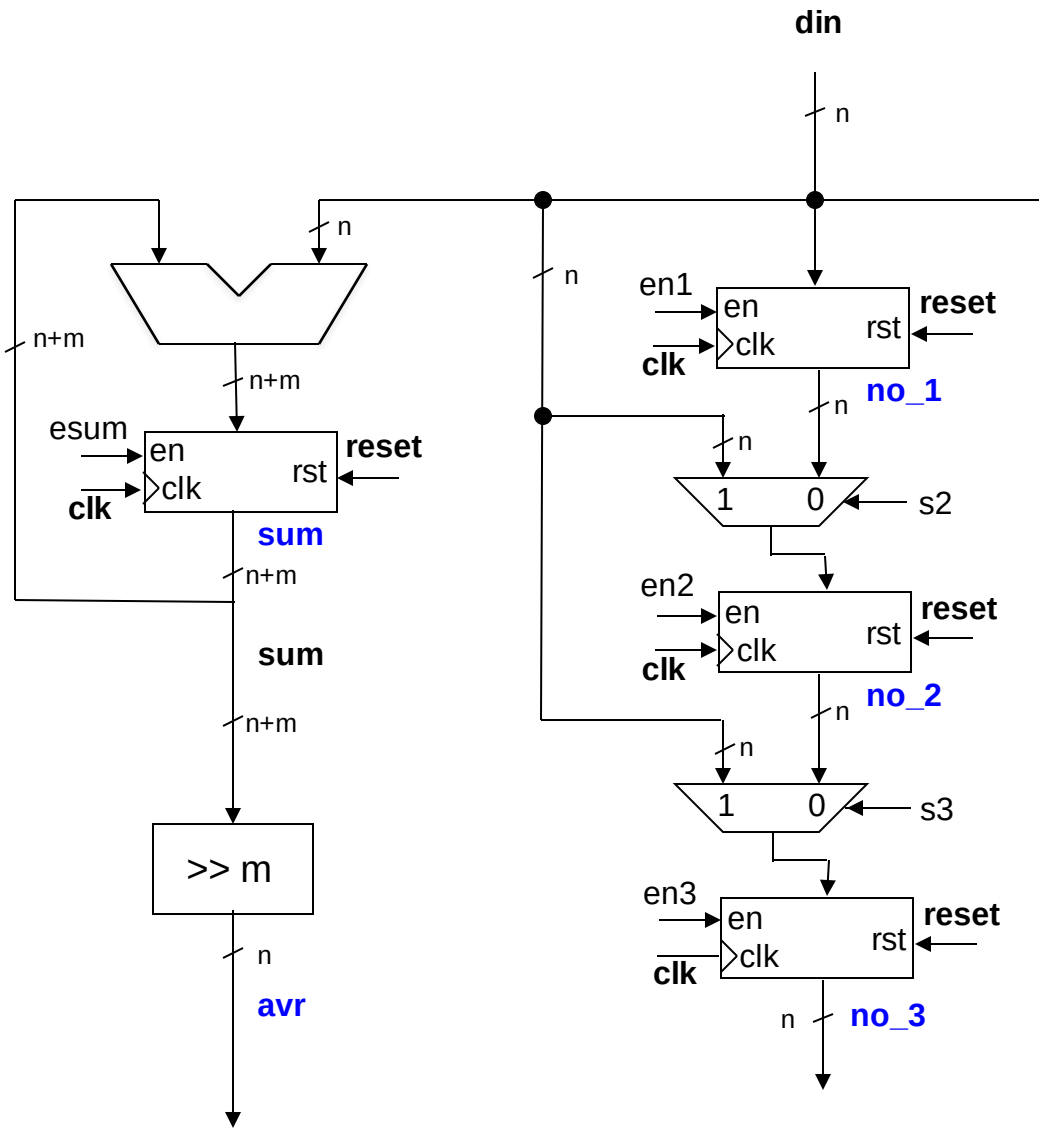
en2
en
clk
rst
reset
clk
n
no_2

en3
en
clk
rst
reset
clk
n
no_3

Pseudocode

$k=2^m$

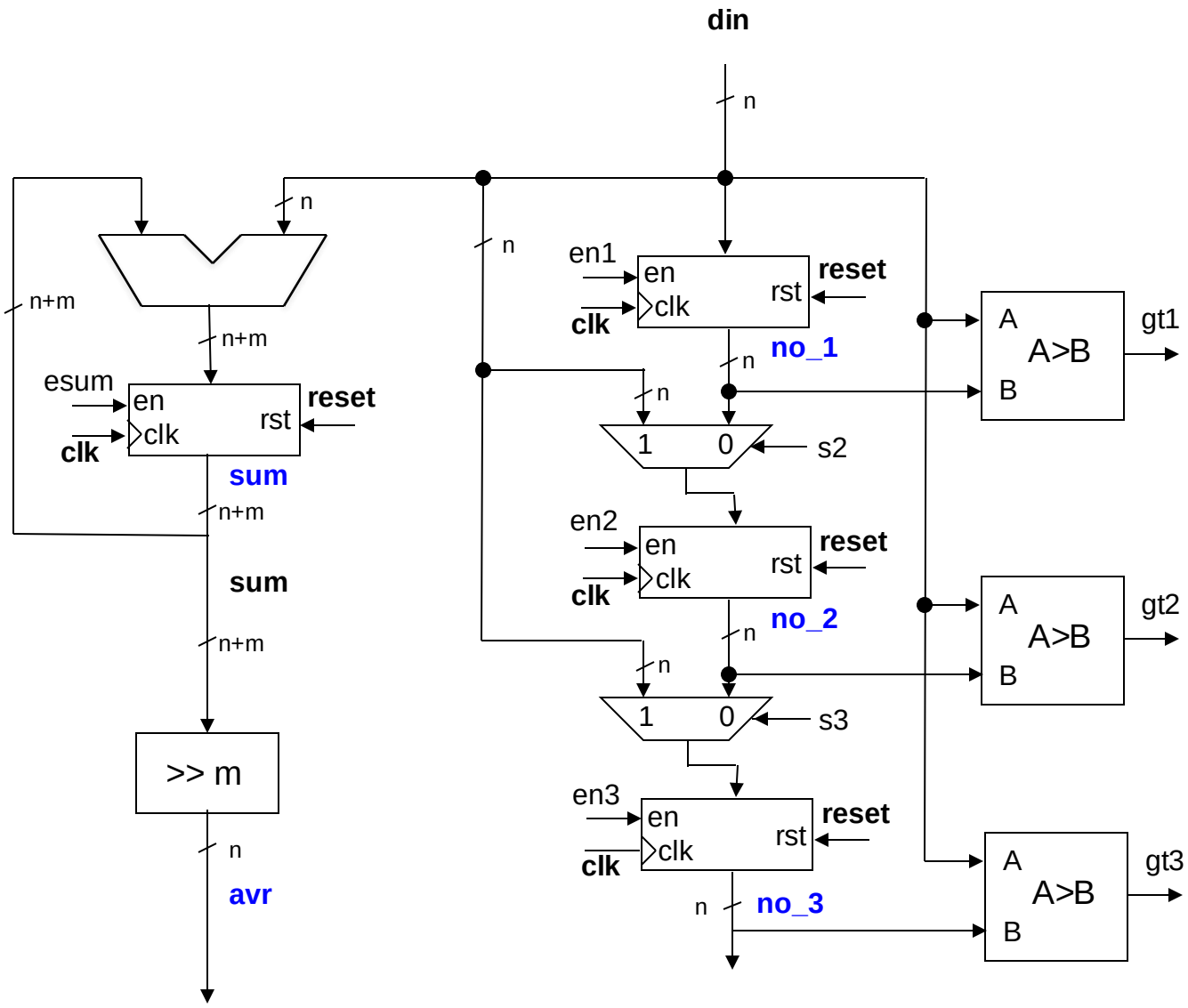
```
no_1 = no_2 = no_3 = sum = 0
wait for go
for i=0 to k-1 do
    sum = sum + din
    if  (din > no_1) then
        no_3 = no_2
        no_2 = no_1
        no_1 = din
    elseif (din > no_2) then
        no_3 = no_2
        no_2 = din
    elseif (din > no_3) then
        no_3 = din
    end if
end for
avr = sum / k
```


din
n
n
n
n+m
n+m
esum
en
clk
rst
reset
clk
sum
n+m
sum
n+m
>> m
n
avr
en1
en
clk
rst
reset
clk
no_1
n
n
1
0
s2
en2
en
clk
rst
reset
clk
no_2
n
n
1
0
s3
en3
en
clk
rst
reset
clk
n
no_3

Pseudocode

$k=2^m$

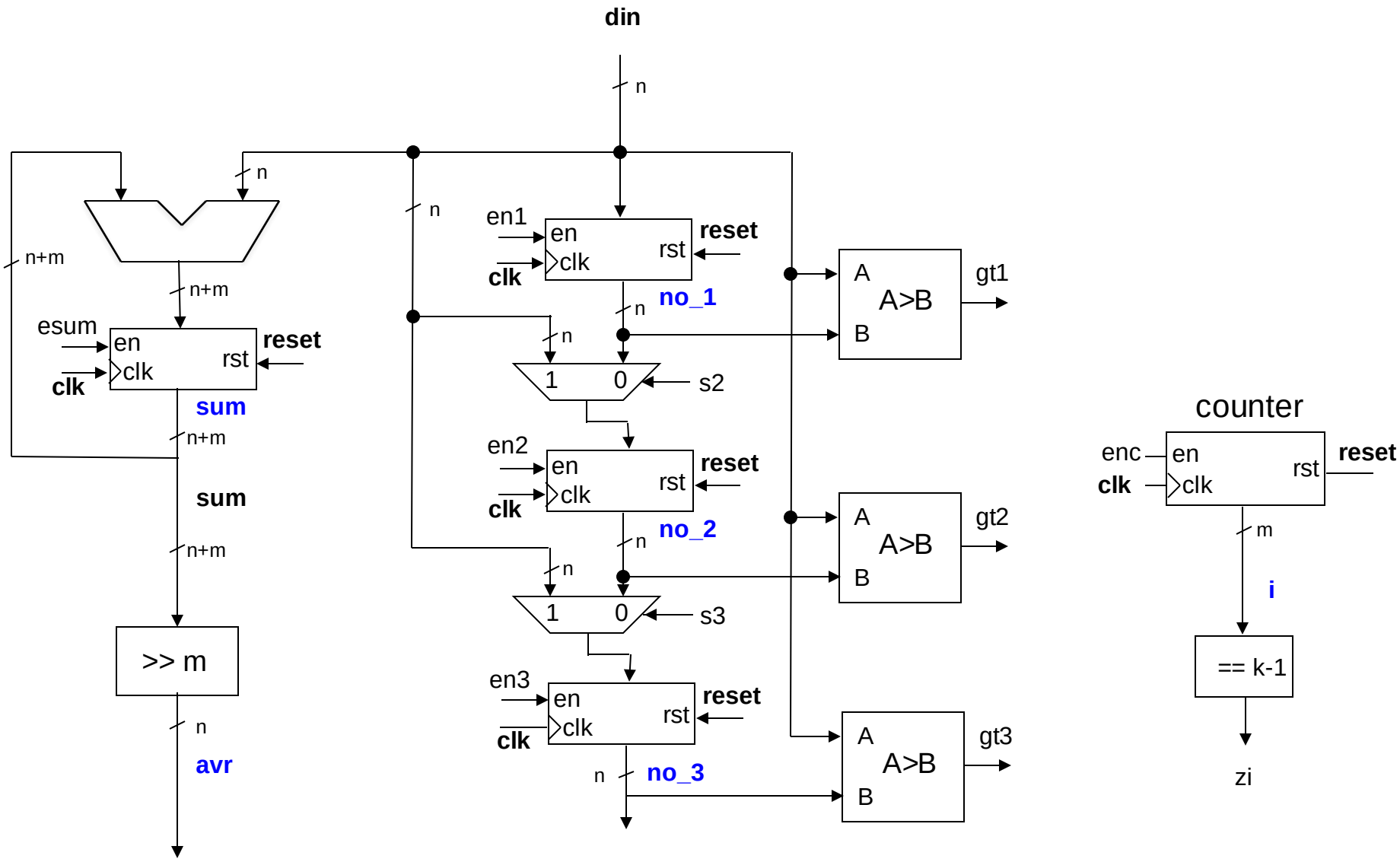
```
no_1 = no_2 = no_3 = sum = 0
wait for go
for i=0 to k-1 do
    sum = sum + din
    if (din > no_1) then
        no_3 = no_2
        no_2 = no_1
        no_1 = din
    elseif (din > no_2) then
        no_3 = no_2
        no_2 = din
    elseif (din > no_3) then
        no_3 = din
    end if
end for
avr = sum / k
```


din
n
n
n
n+m
n+m
esum
en
clk
rst
reset
clk
sum
n+m
sum
n+m
>> m
n
avr
en1
en
clk
rst
reset
clk
no_1
n
n
1
0
s2
en2
en
clk
rst
reset
clk
no_2
n
n
1
0
s3
en3
en
clk
rst
reset
clk
n
no_3
A
A>B
B
gt1
A
A>B
B
gt2
A
A>B
B
gt3

Pseudocode

$k=2^m$

```
no_1 = no_2 = no_3 = sum = 0
wait for go
for i=0 to k-1 do
    sum = sum + din
    if  (din > no_1) then
            no_3 = no_2
            no_2 = no_1
            no_1 = din
    elseif (din > no_2) then
            no_3 = no_2
            no_2 = din
    elseif (din > no_3) then
            no_3 = din
    end if
end for
avr = sum / k
```



Block diagram of the Datapath

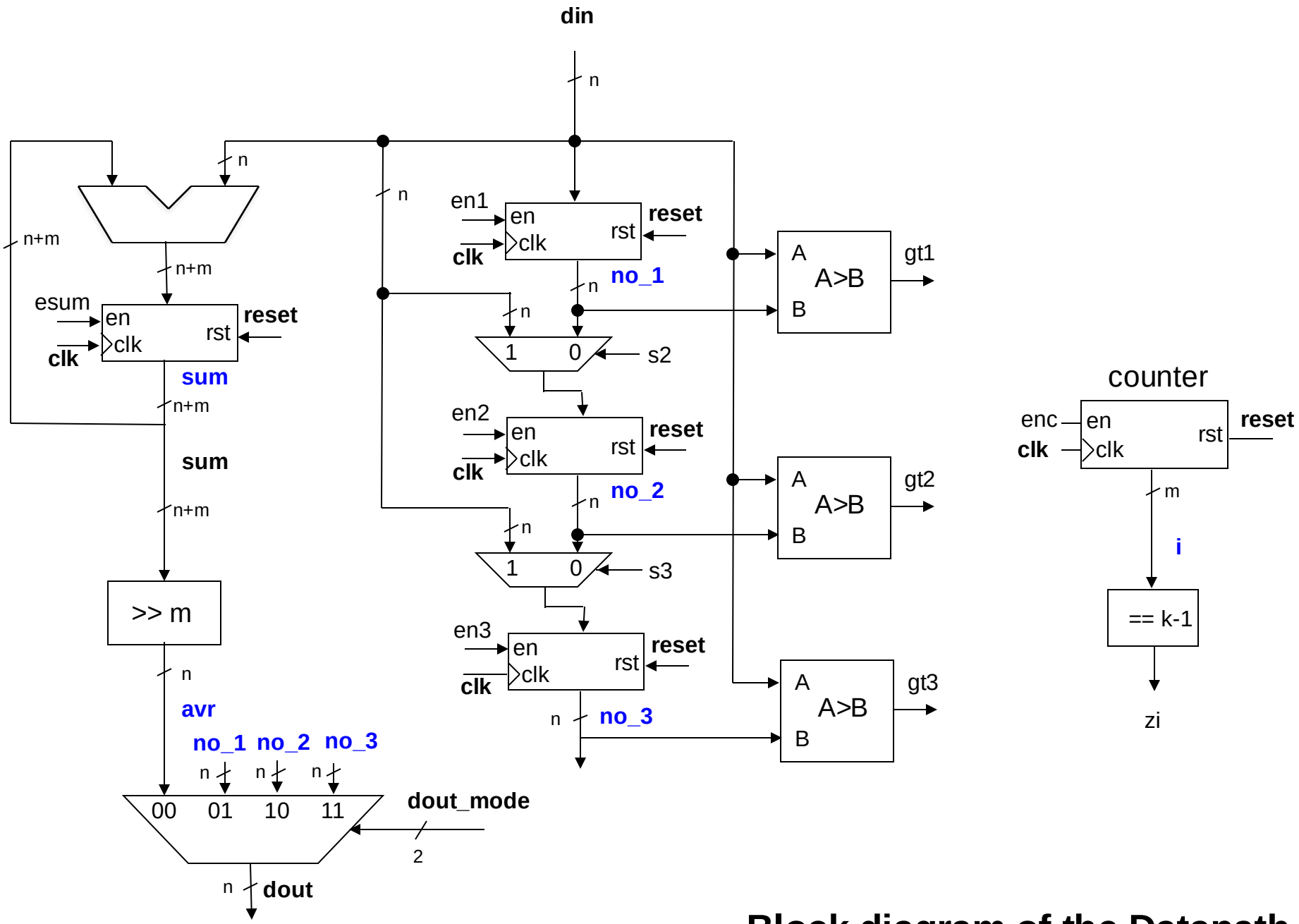
Pseudocode

$k=2^m$

```
no_1 = no_2 = no_3 = sum = 0
wait for go
for i=0 to k-1 do
    sum = sum + din
    if  (din > no_1) then
            no_3 = no_2
            no_2 = no_1
            no_1 = din
    elseif (din > no_2) then
            no_3 = no_2
            no_2 = din
    elseif (din > no_3) then
            no_3 = din
    end if
end for
avr = sum / k
```

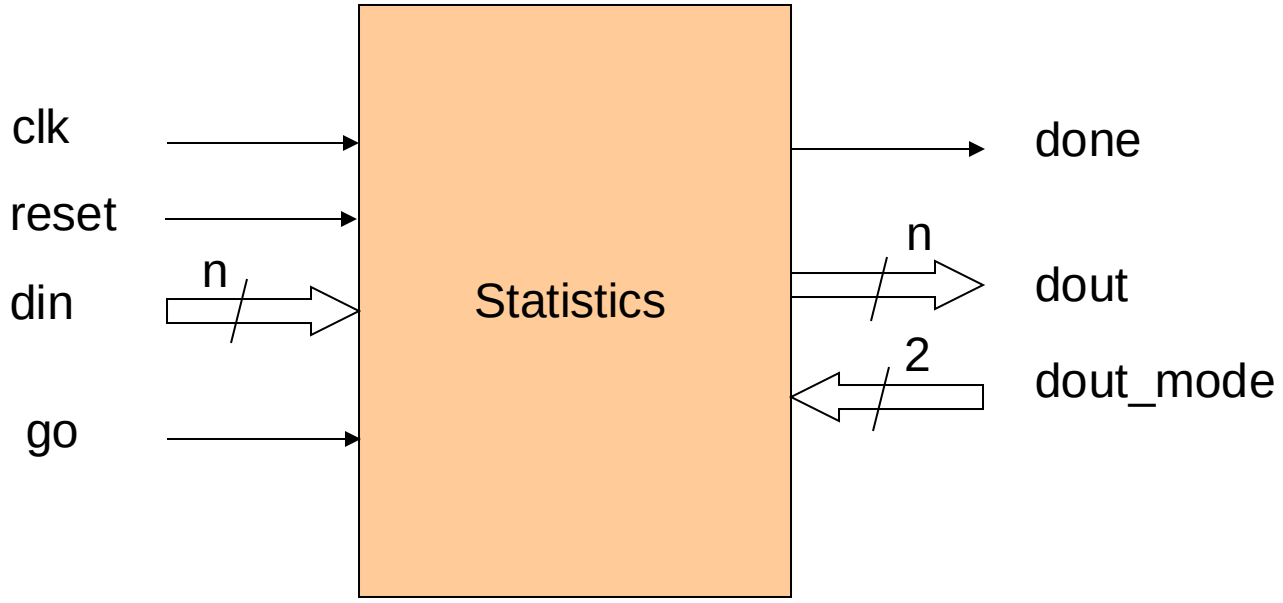
Interface Table

Port	Width	Meaning
clk	1	System clock.
reset	1	System reset.
din	n	Input Data.
go	1	Control signal indicating that the first input is ready. Active for one clock cycle.
done	1	Signal set to high after the output is ready.
dout	n	Output dependent on the dout_mode input.
dout_mode	2	Control signal determining value available at the output. 00: avr, 01: no_1, 10: no_2, 11: no_3.

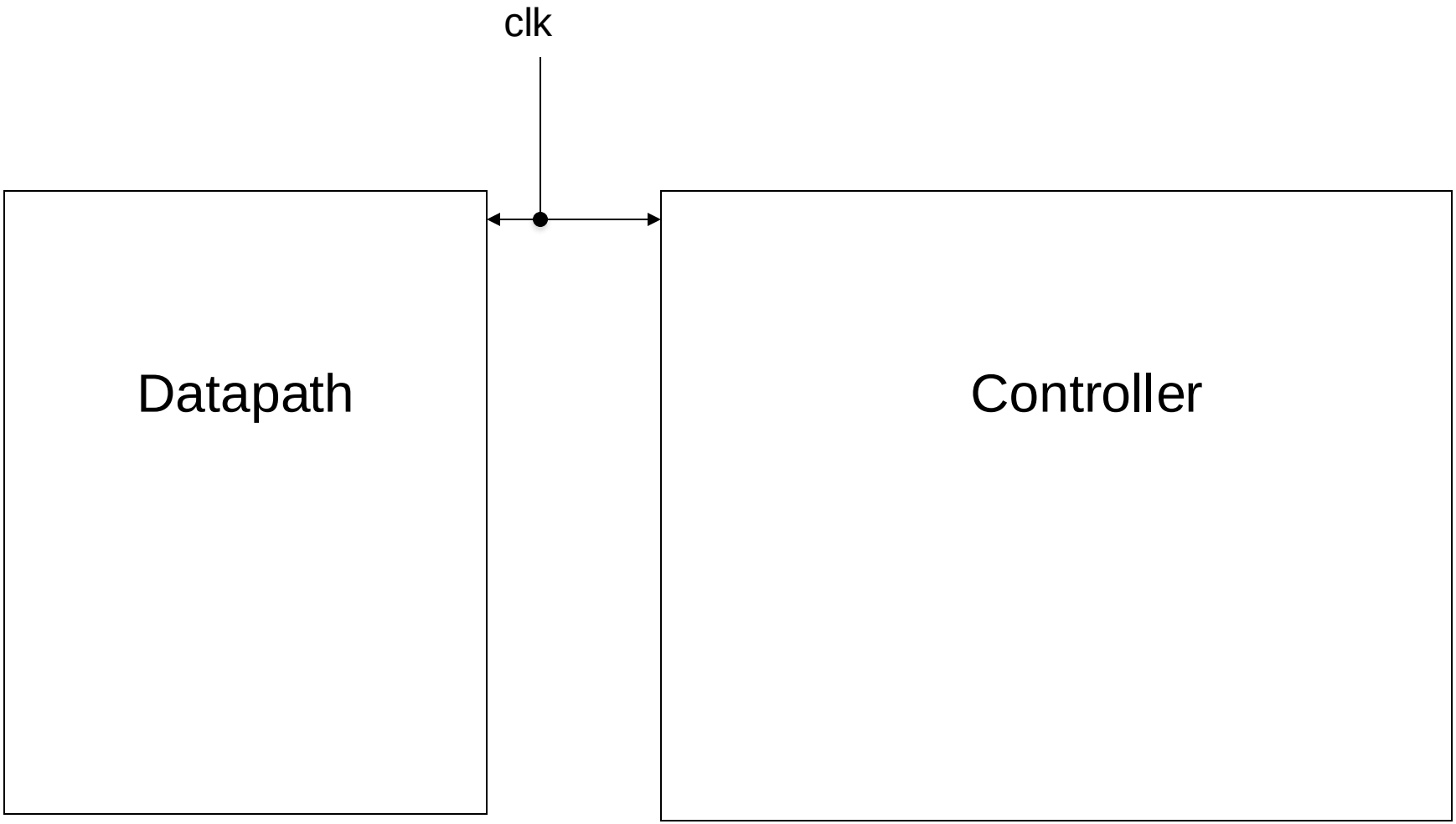


Block diagram of the Datapath

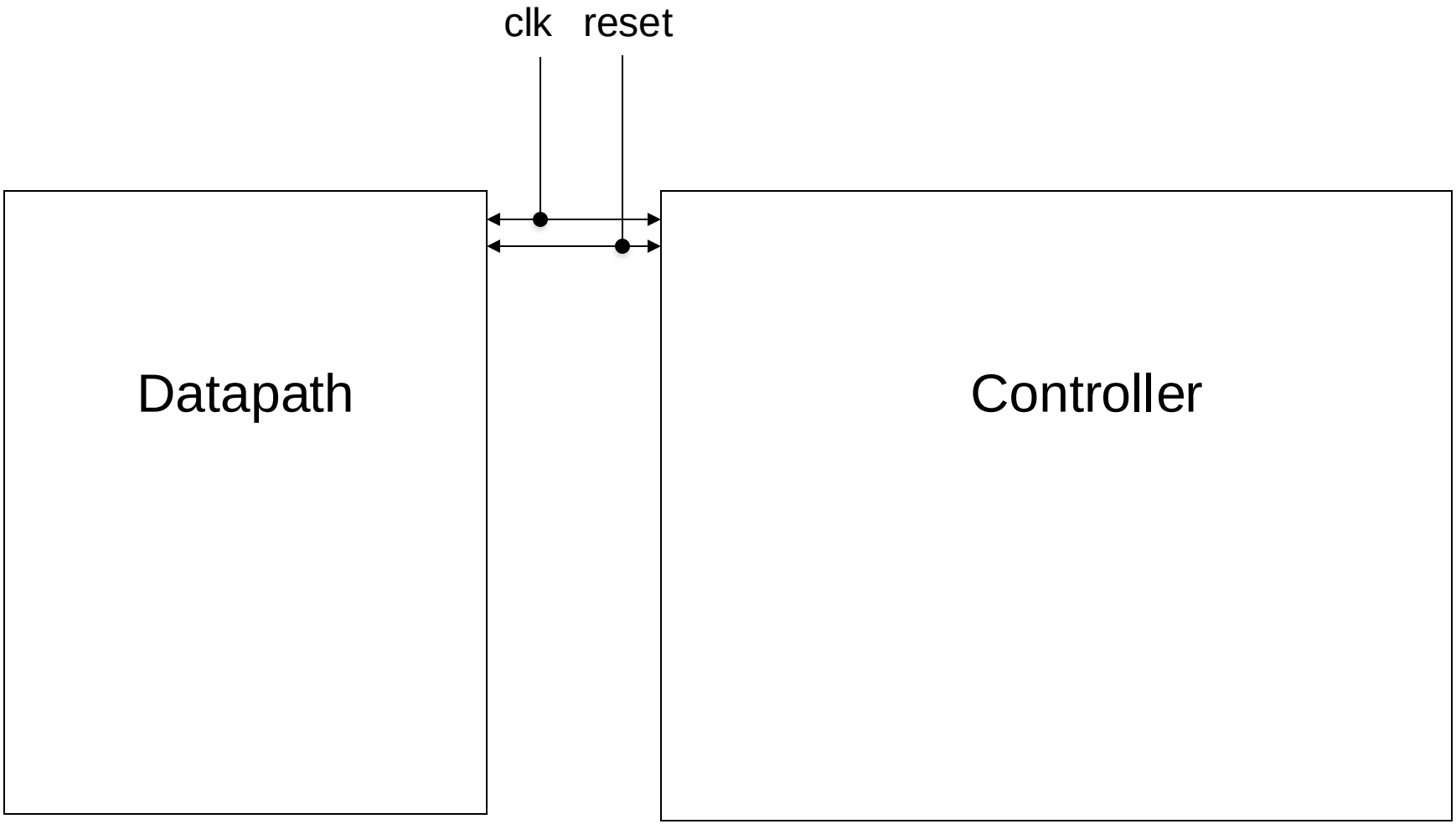
Circuit Interface



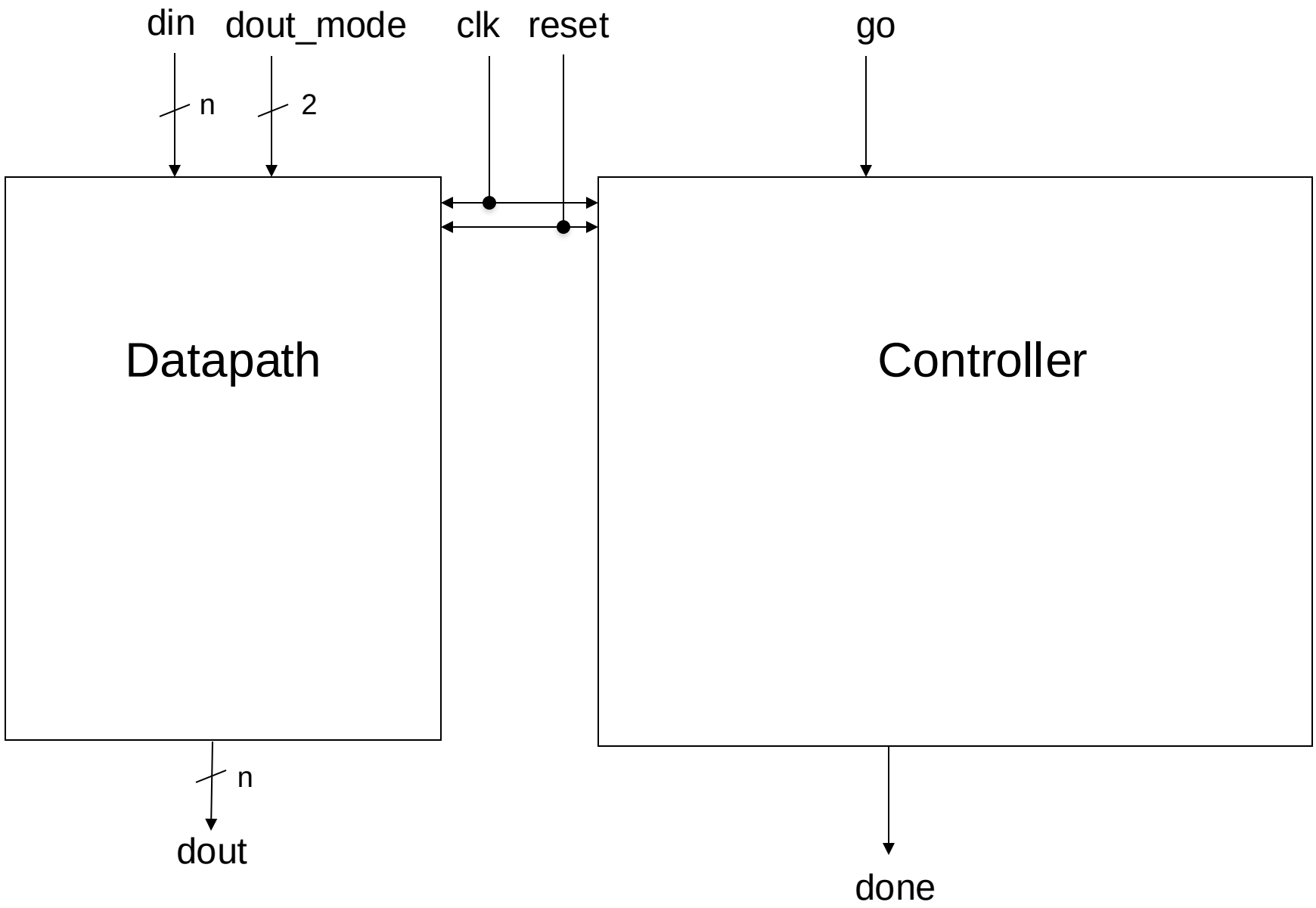
Interface with the division into the Datapath and Controller

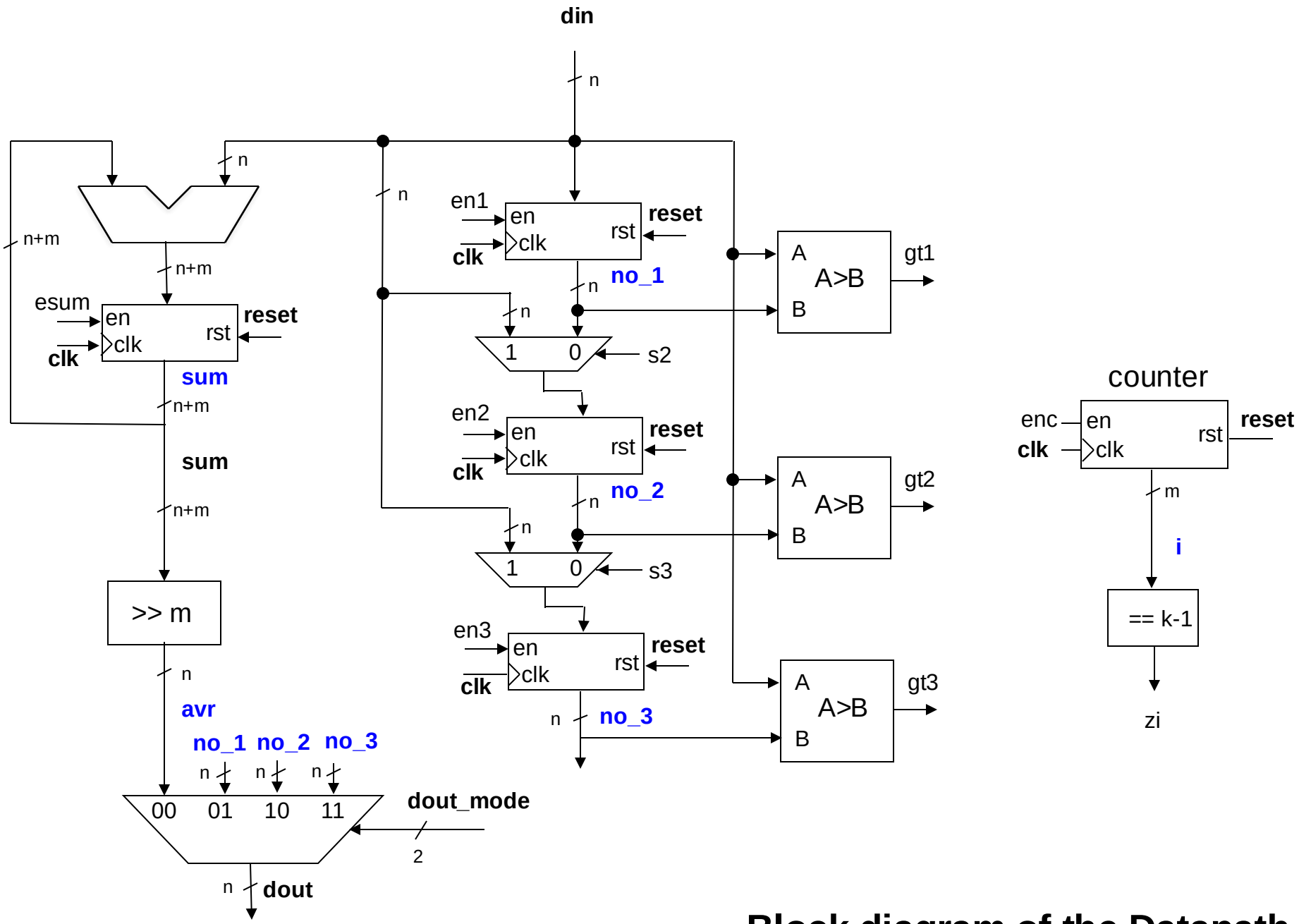


Interface with the division into the Datapath and Controller



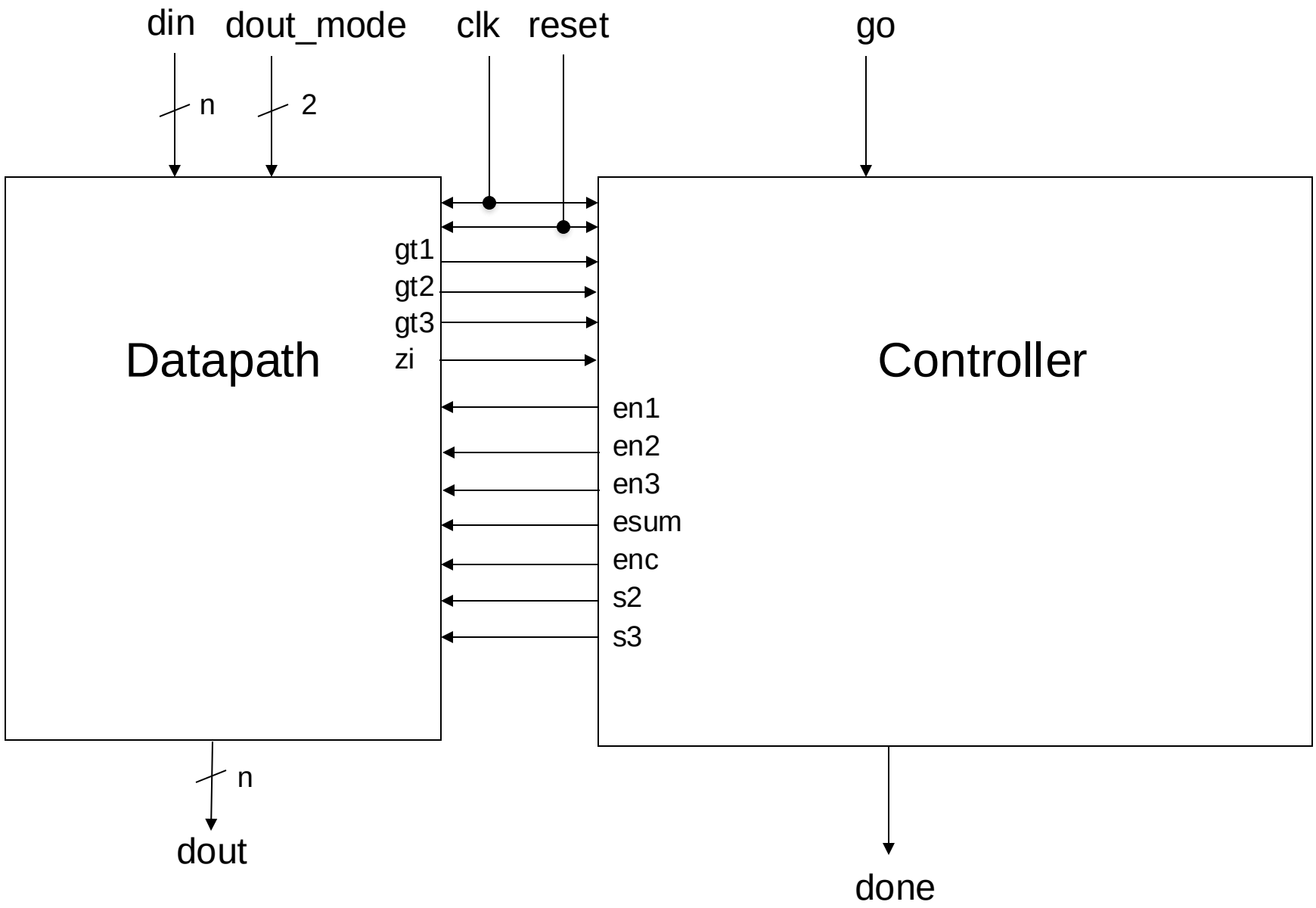
Interface with the division into the Datapath and Controller



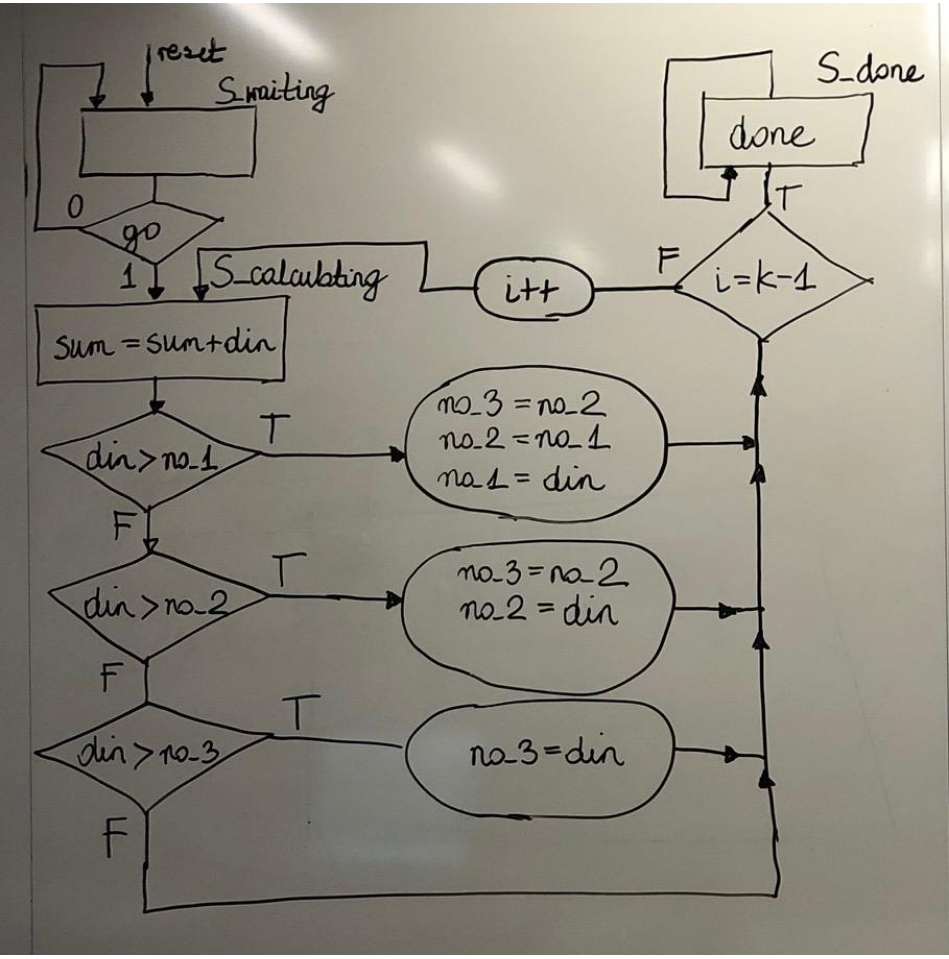


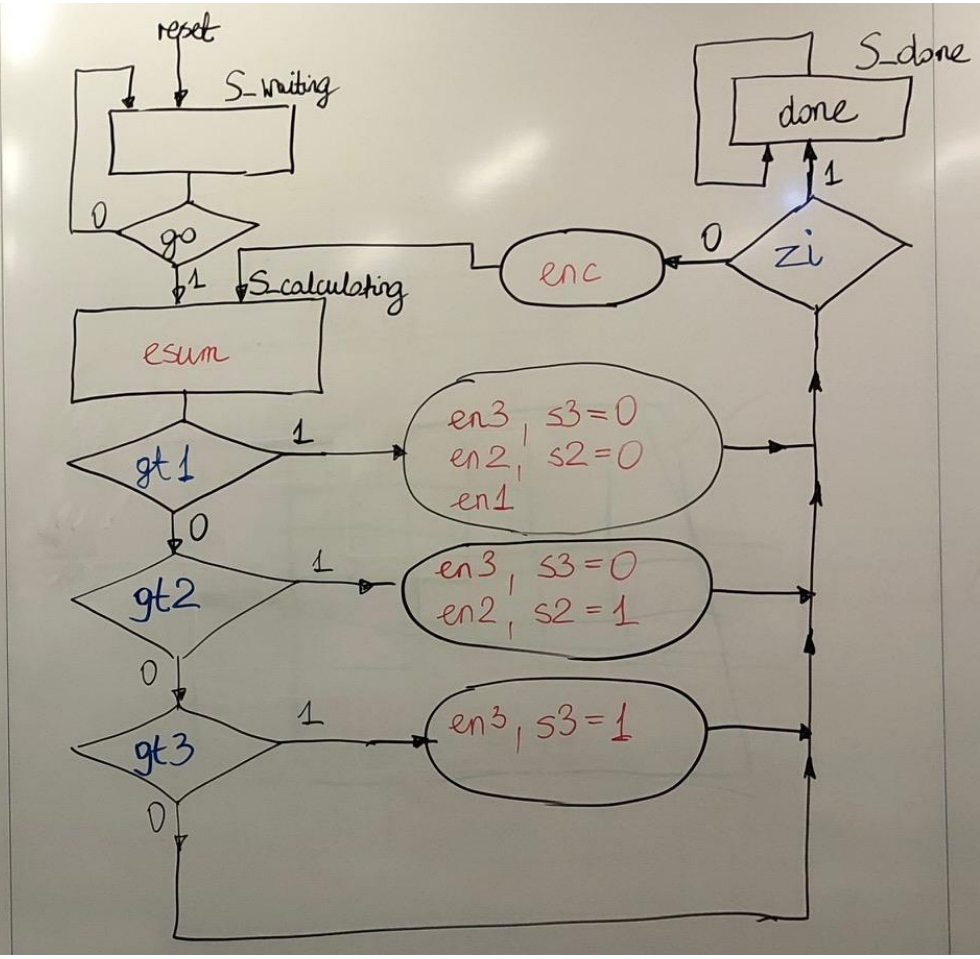
Block diagram of the Datapath

Interface with the division into the Datapath and Controller

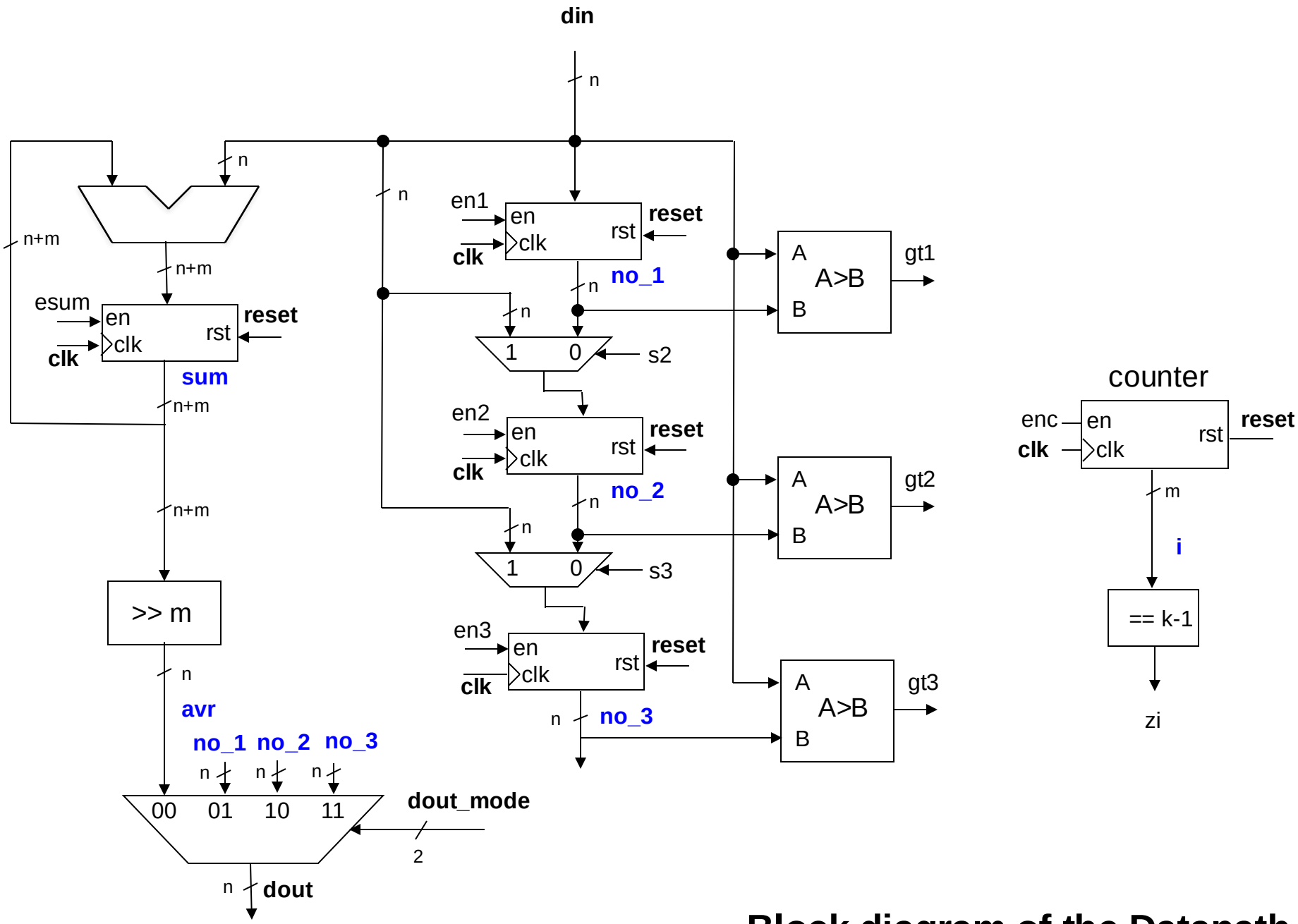


ASM Charts


reset
S_waiting
0
go
1
S_calculating
sum = sum+din
din > no_1
T
F
no_3 = no_2
no_2 = no_1
no_1 = din
din > no_2
T
F
no_3 = no_2
no_2 = din
din > no_3
T
F
no_3 = din
S_done
done
T
F
i=k-1
i++

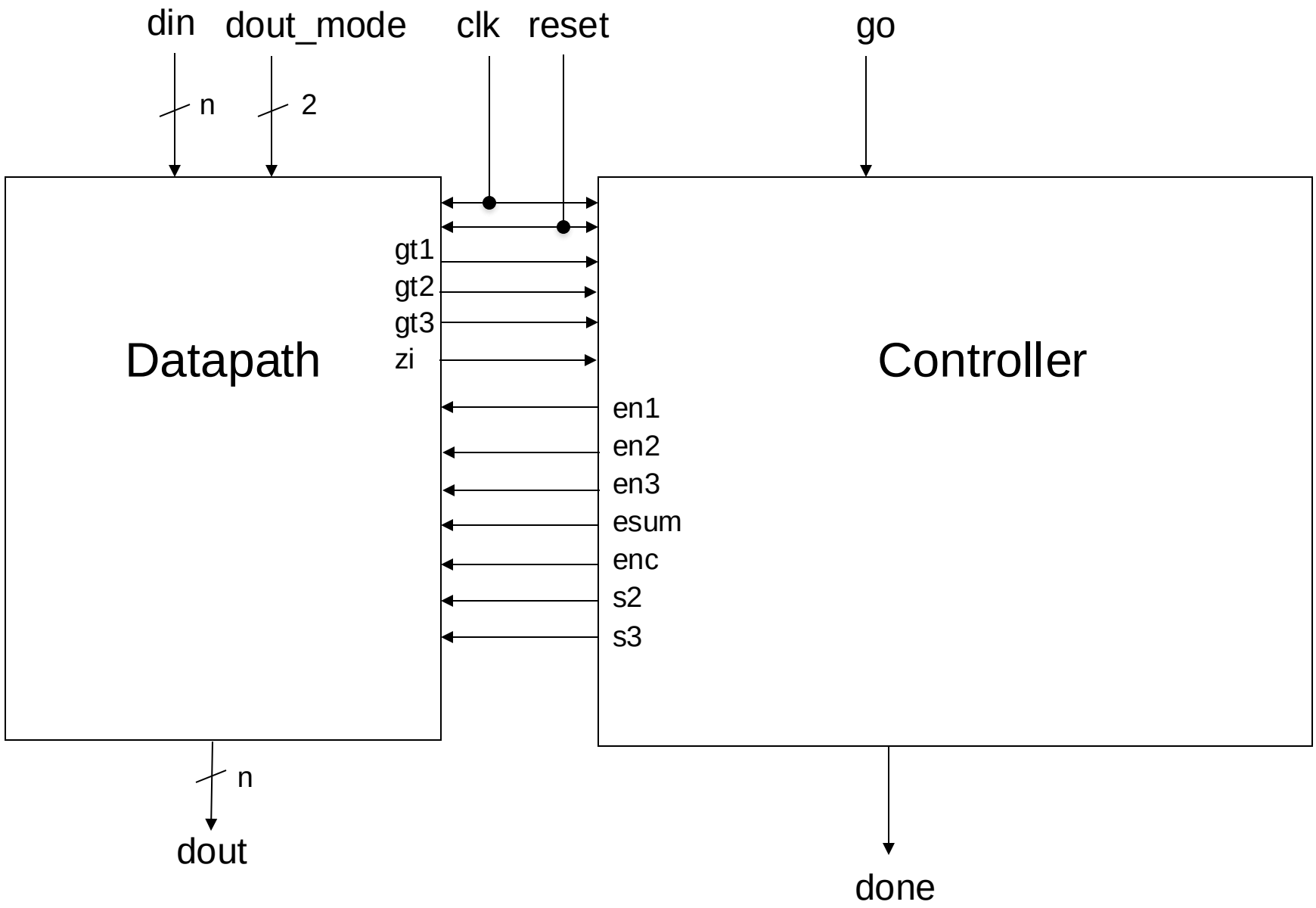

reset
S_waiting
0
go
1
S_calculating
esum
gt1
1
0
en3, s3=0
en2, s2=0
en1
gt2
1
0
en3, s3=0
en2, s2=1
gt3
1
0
en3, s3=1
enc
0
zi
1
S_done
done

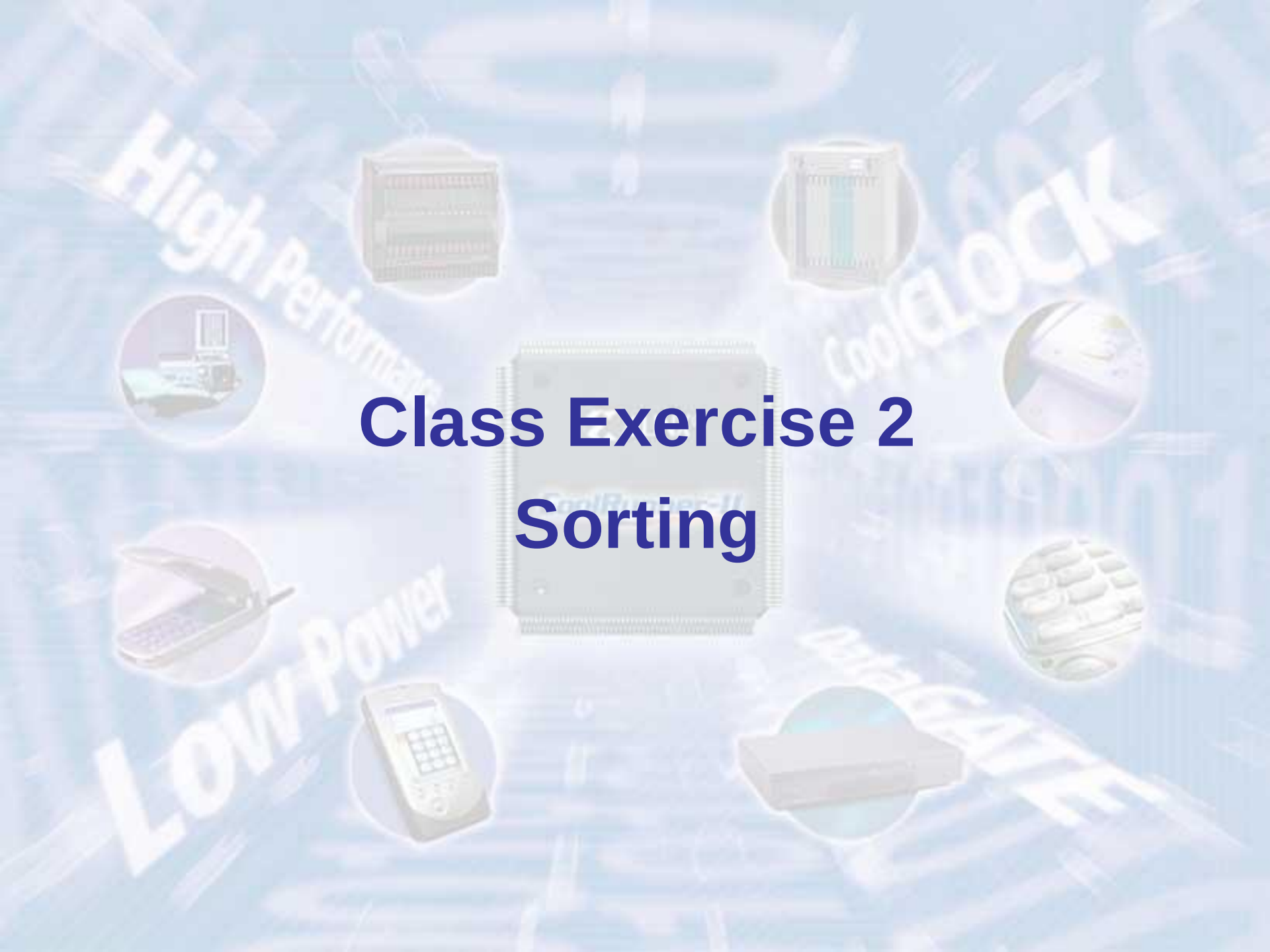
STATISTICS:
Solutions



Block diagram of the Datapath

Interface with the division into the Datapath and Controller





Class Exercise 2
Sorting

Text Description

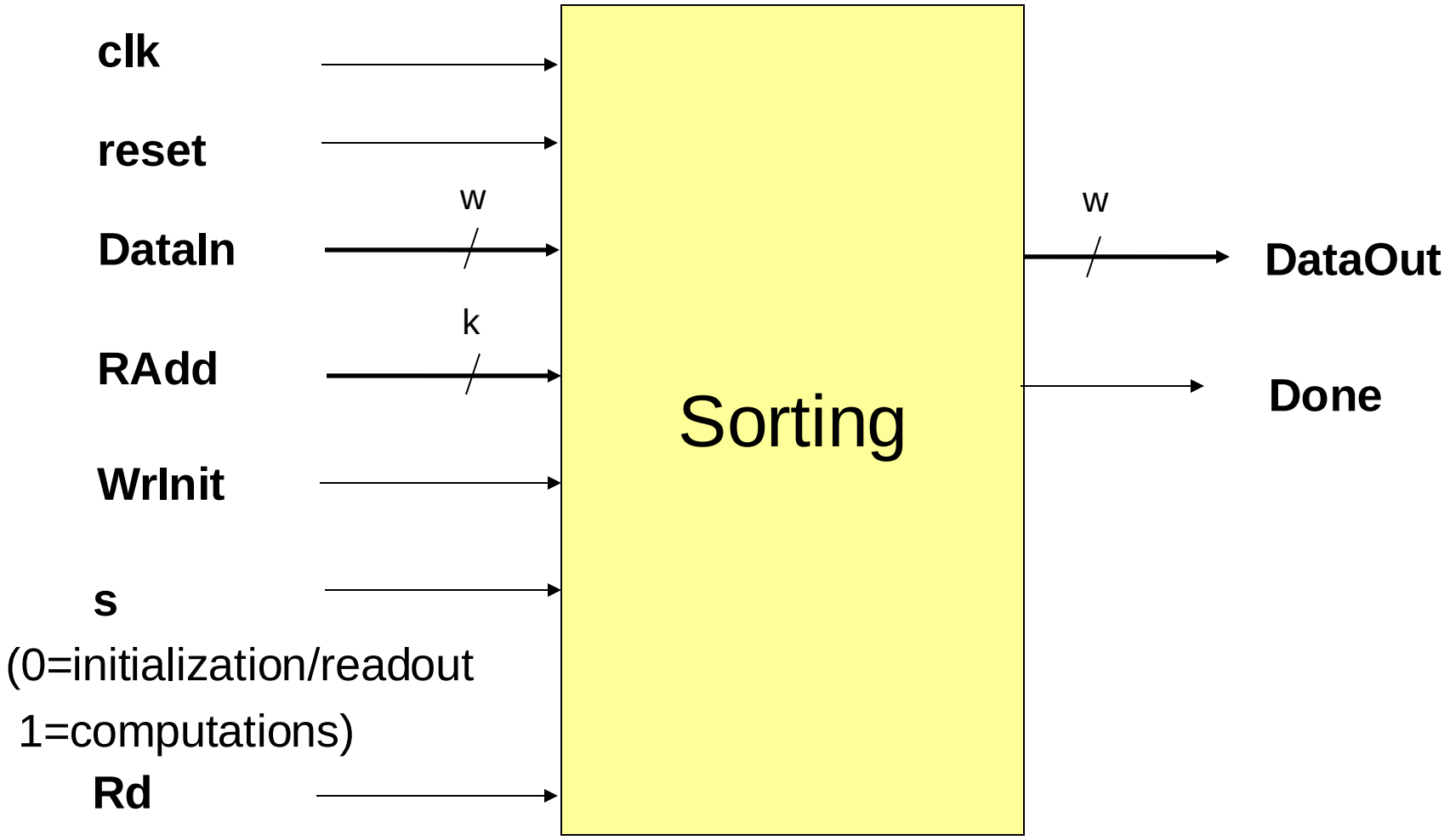
Function:

Design a circuit capable of sorting $N=2^k$ w-bit numbers in descending order.

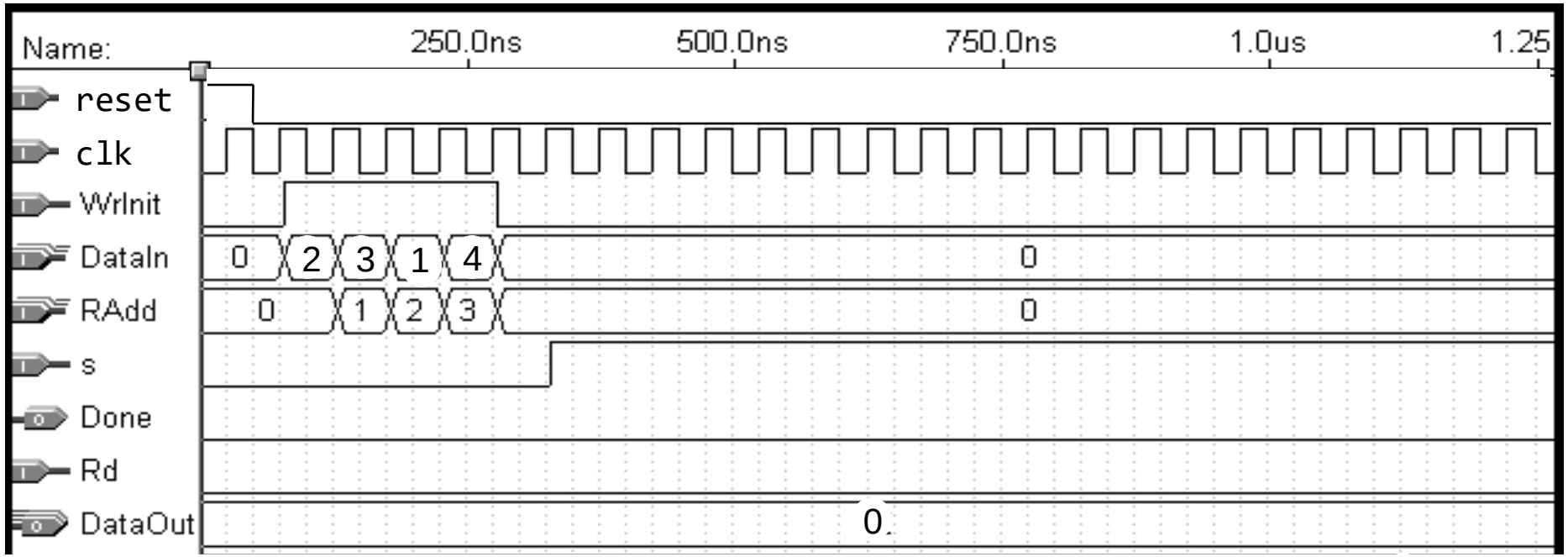
Optimization:

Optimize your circuit for the minimum total execution time. When choosing between two circuits with the same or very similar execution time, give preference to the circuit with the smaller area.

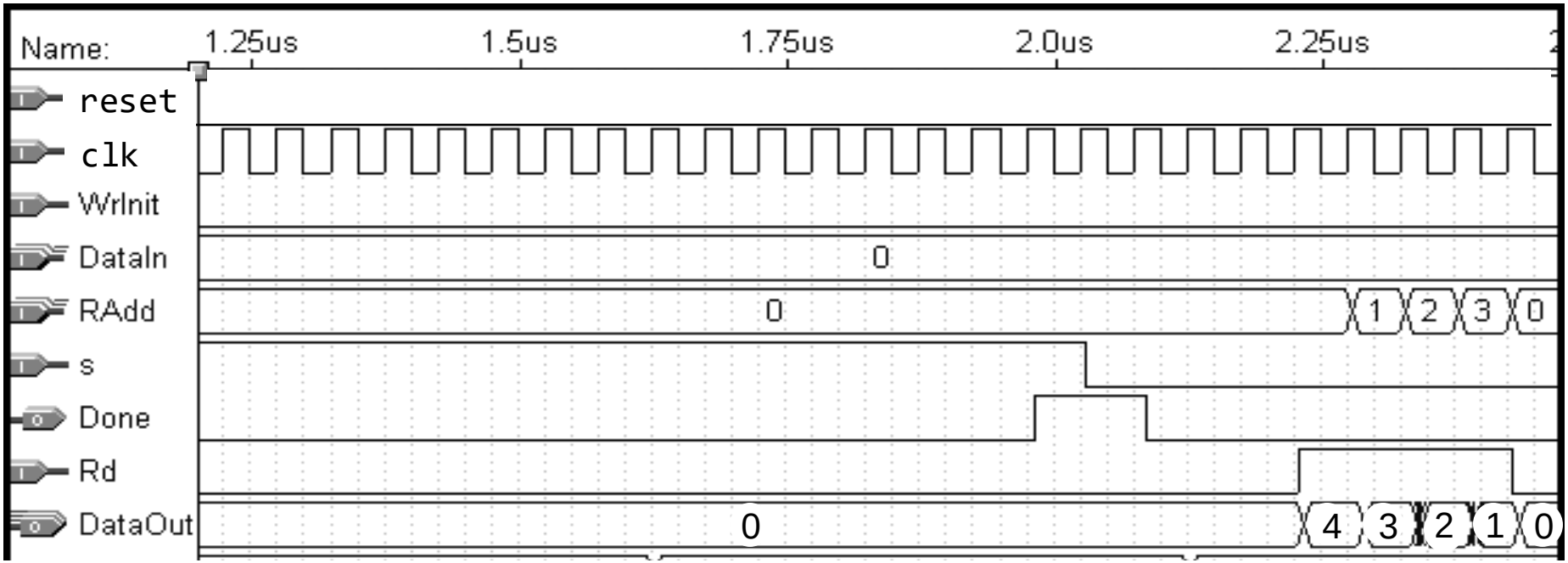
Interface


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

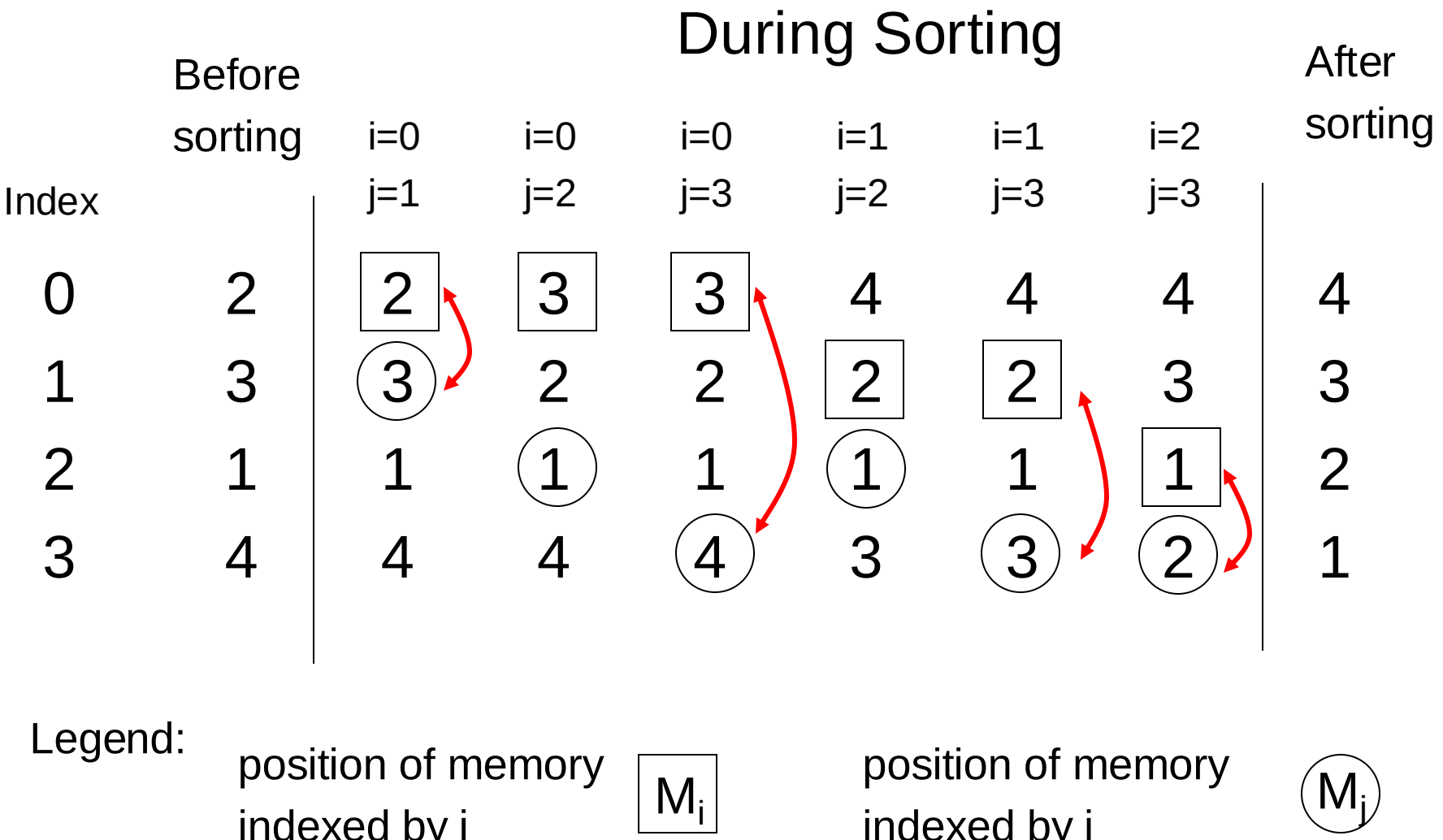
Simulation results for the sort operation (1)
Loading memory and starting sorting



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



Sorting – Example for N=4



Legend:

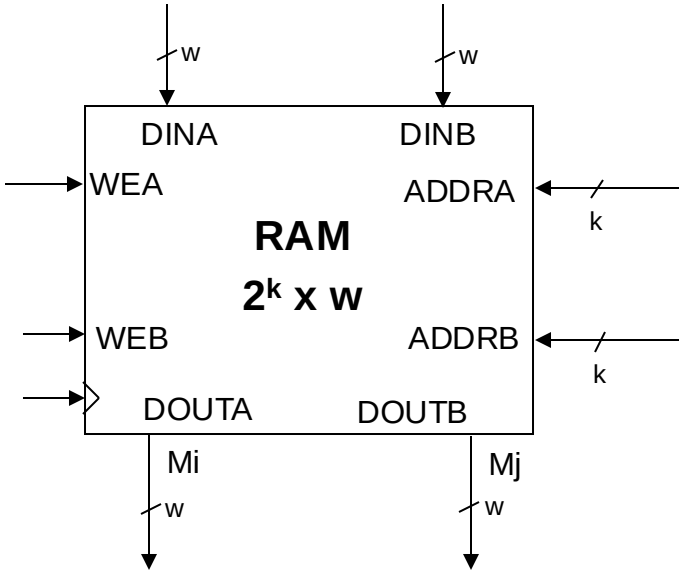
position of memory indexed by i M_i

position of memory indexed by j M_j

$N=2^k$

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

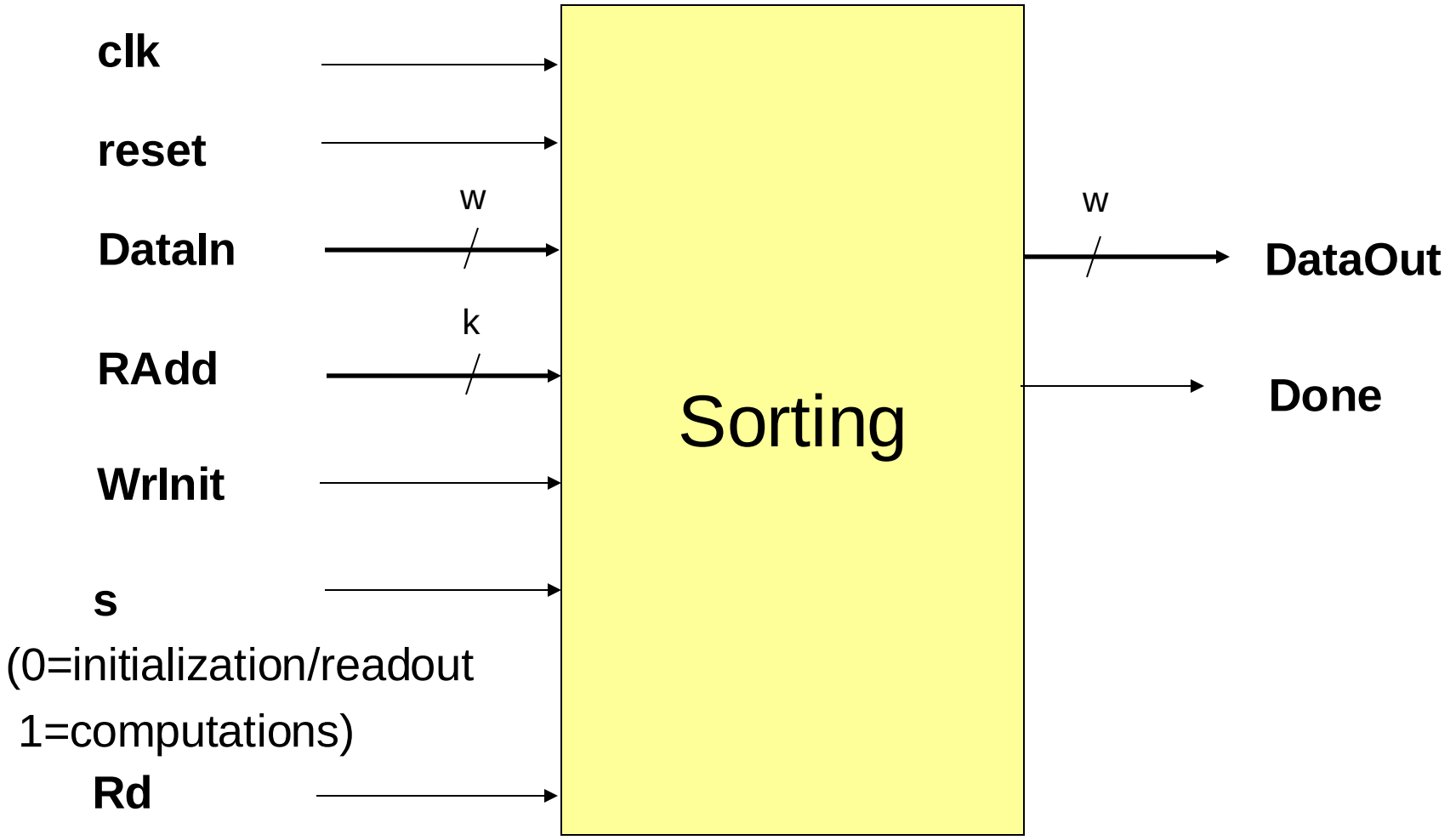


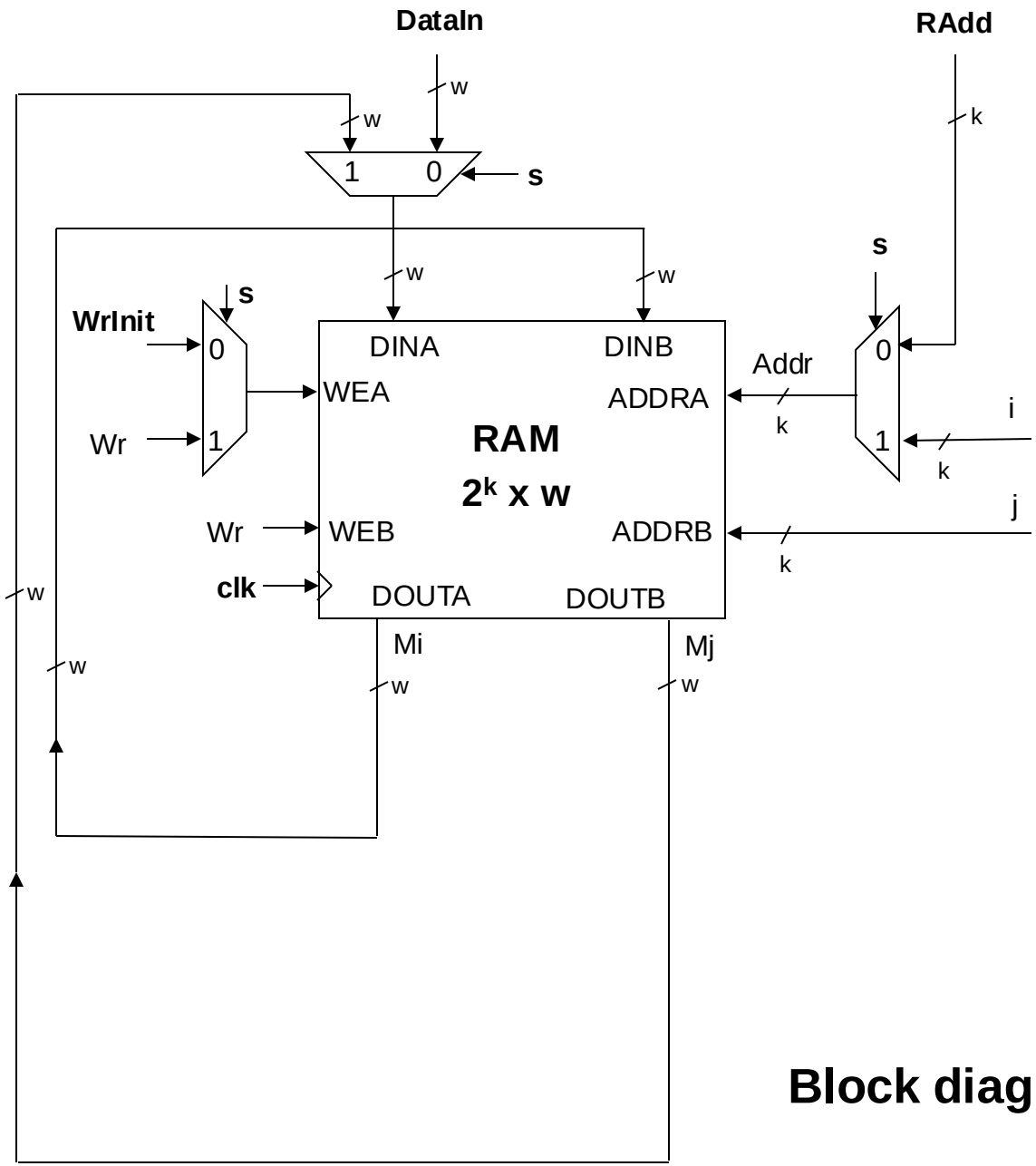
Block diagram of the Datapath

Pseudocode

wait for s=1 // initialize internal memory
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

Interface


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

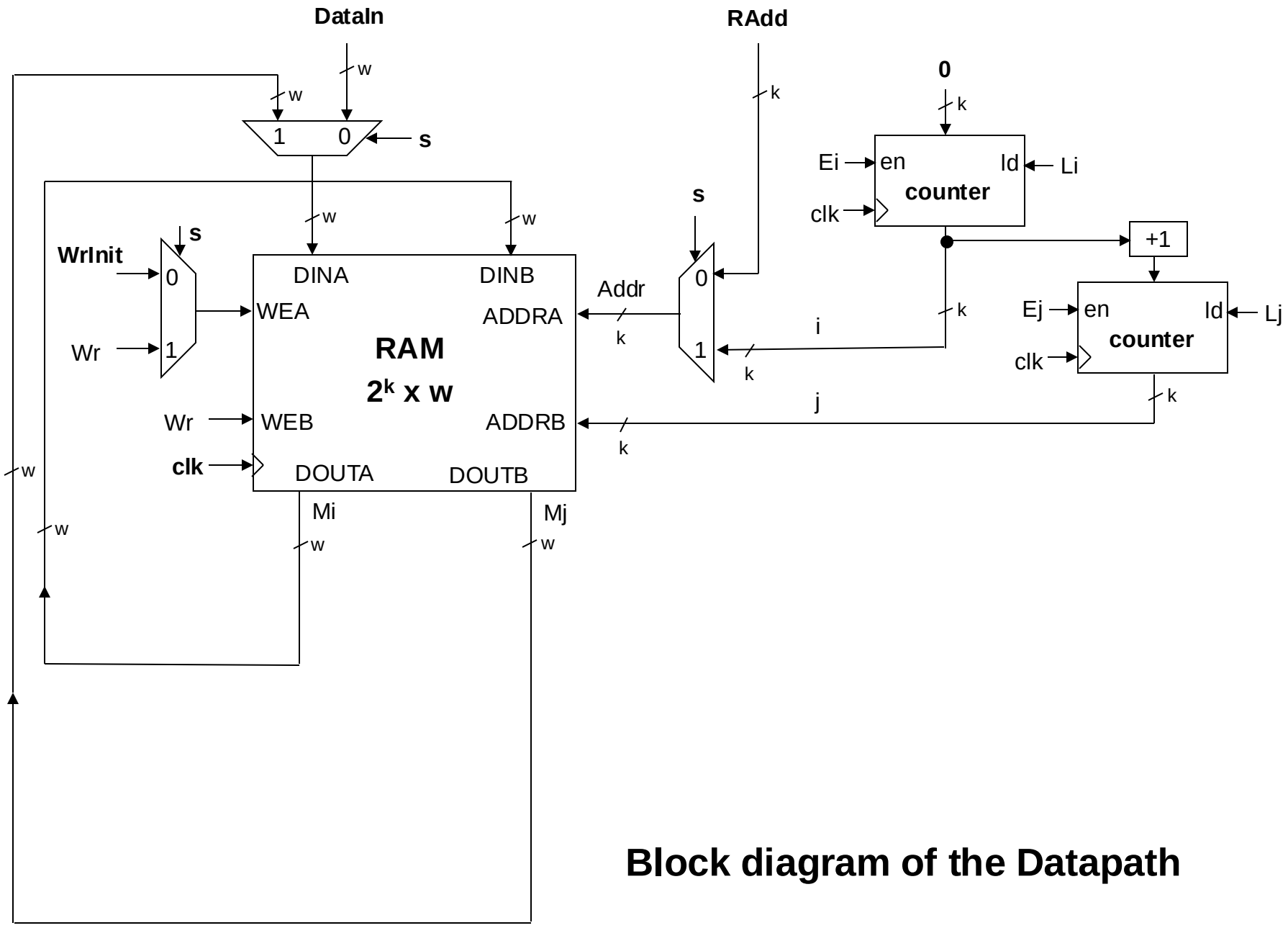


Block diagram of the Datapath

$N=2^k$

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
```

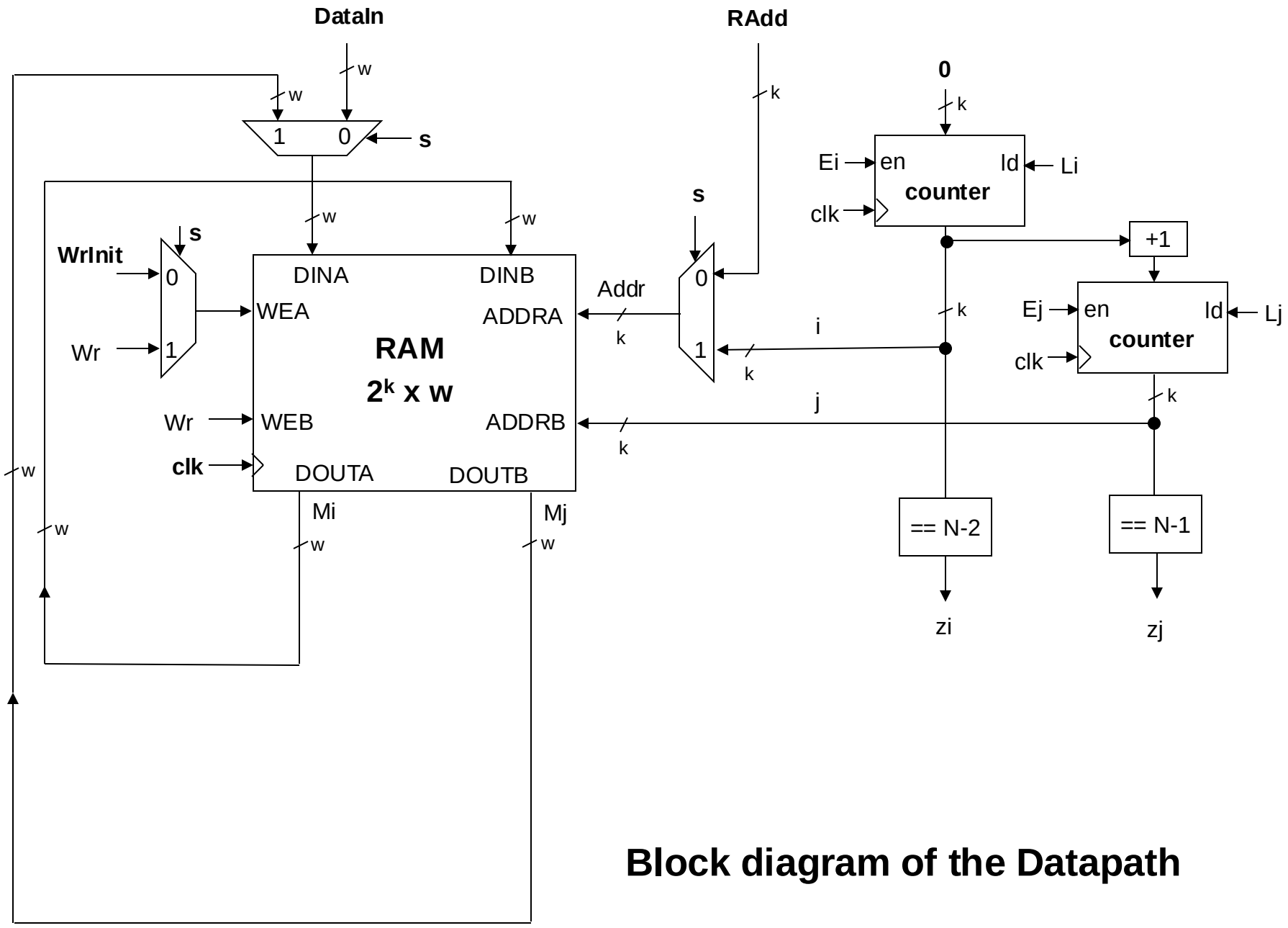


Block diagram of the Datapath

$N=2^k$

Pseudocode

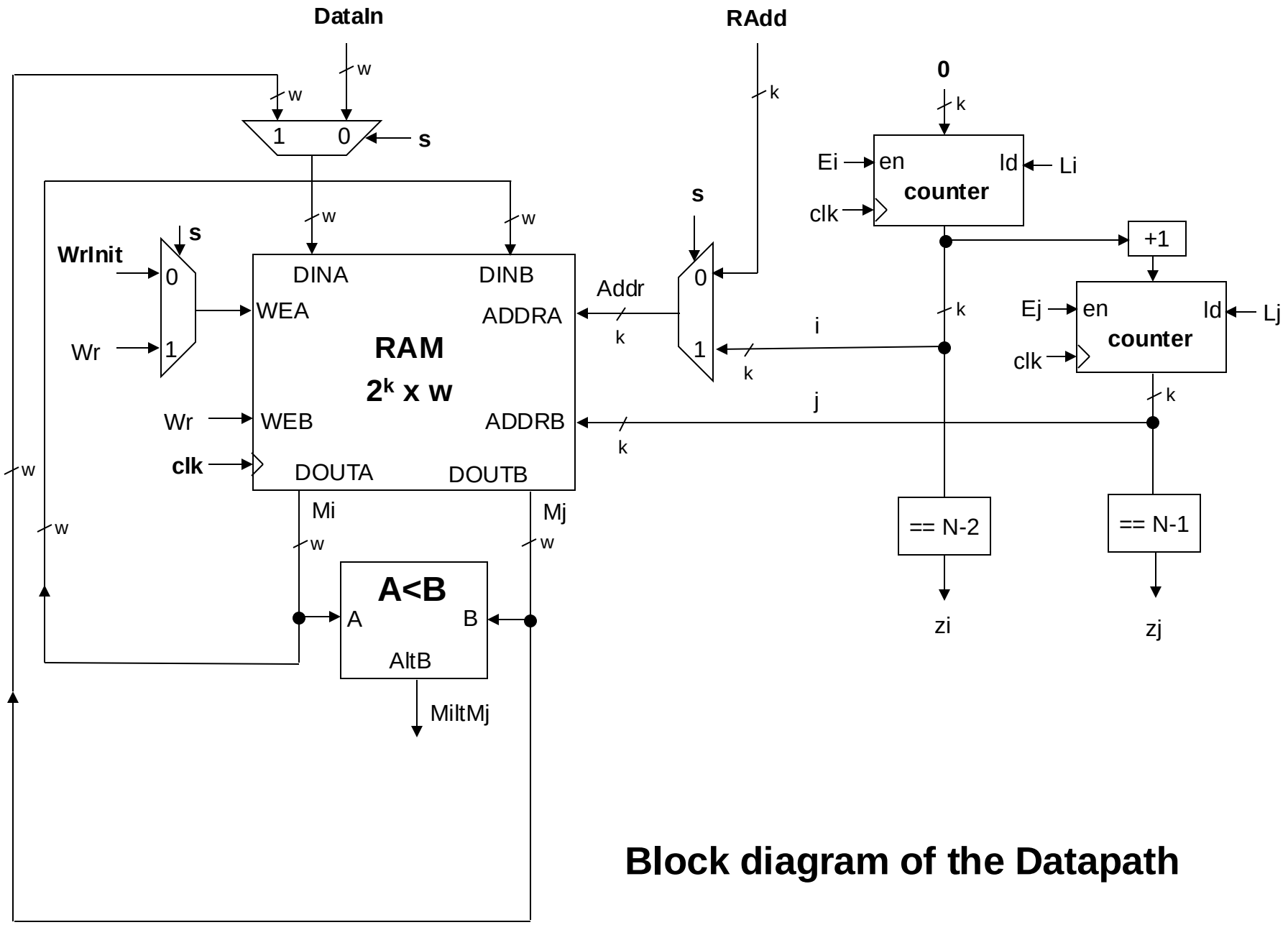
wait for s=1
for i=0 to N-2 do
    for j=i+1 to N-1 do
        if $M_i < M_j$ then
            Swap M_i with M_j
        end if
    end for
end for
Done
wait for s=0
go to the beginning



Block diagram of the Datapath

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
```

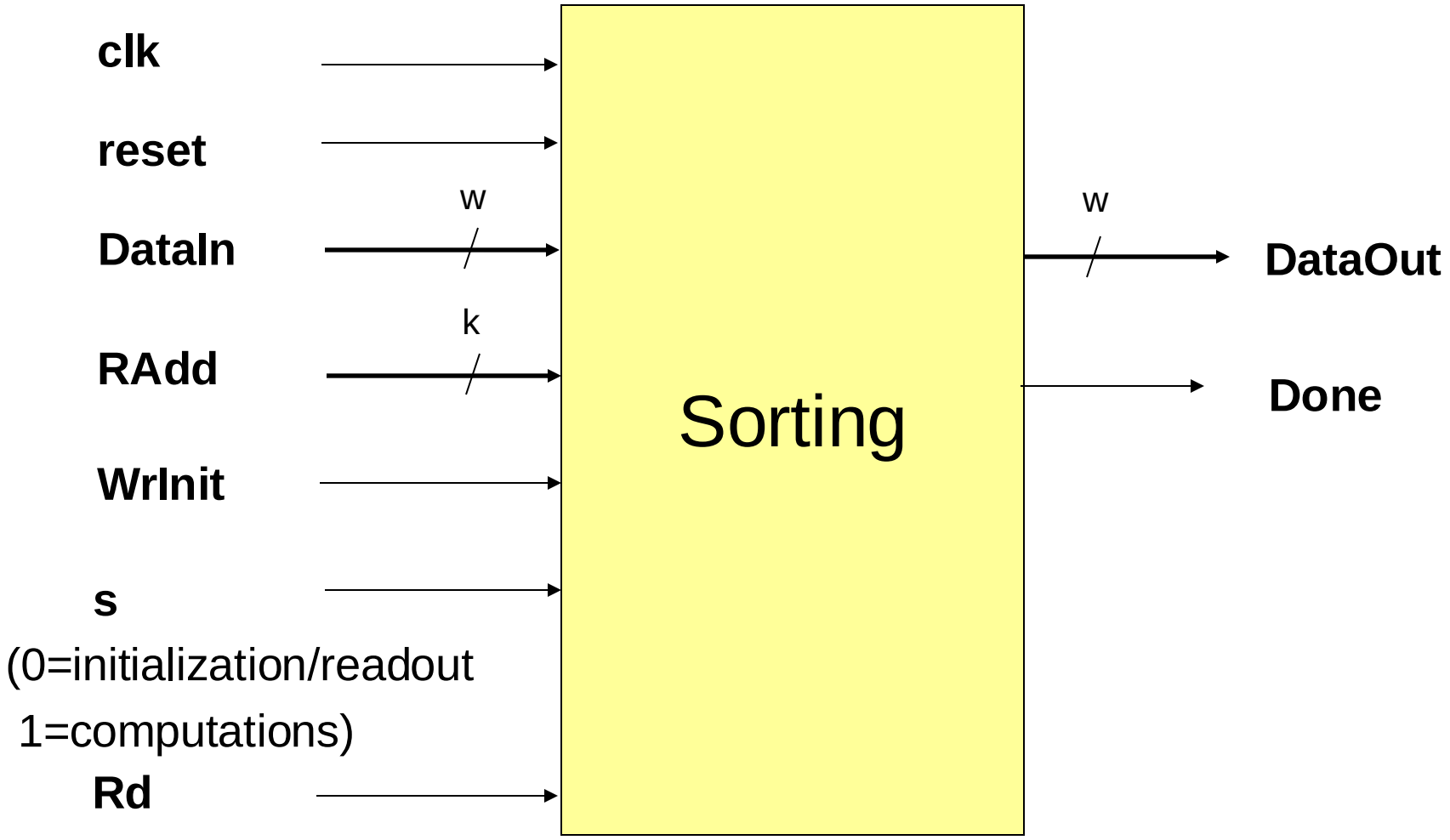


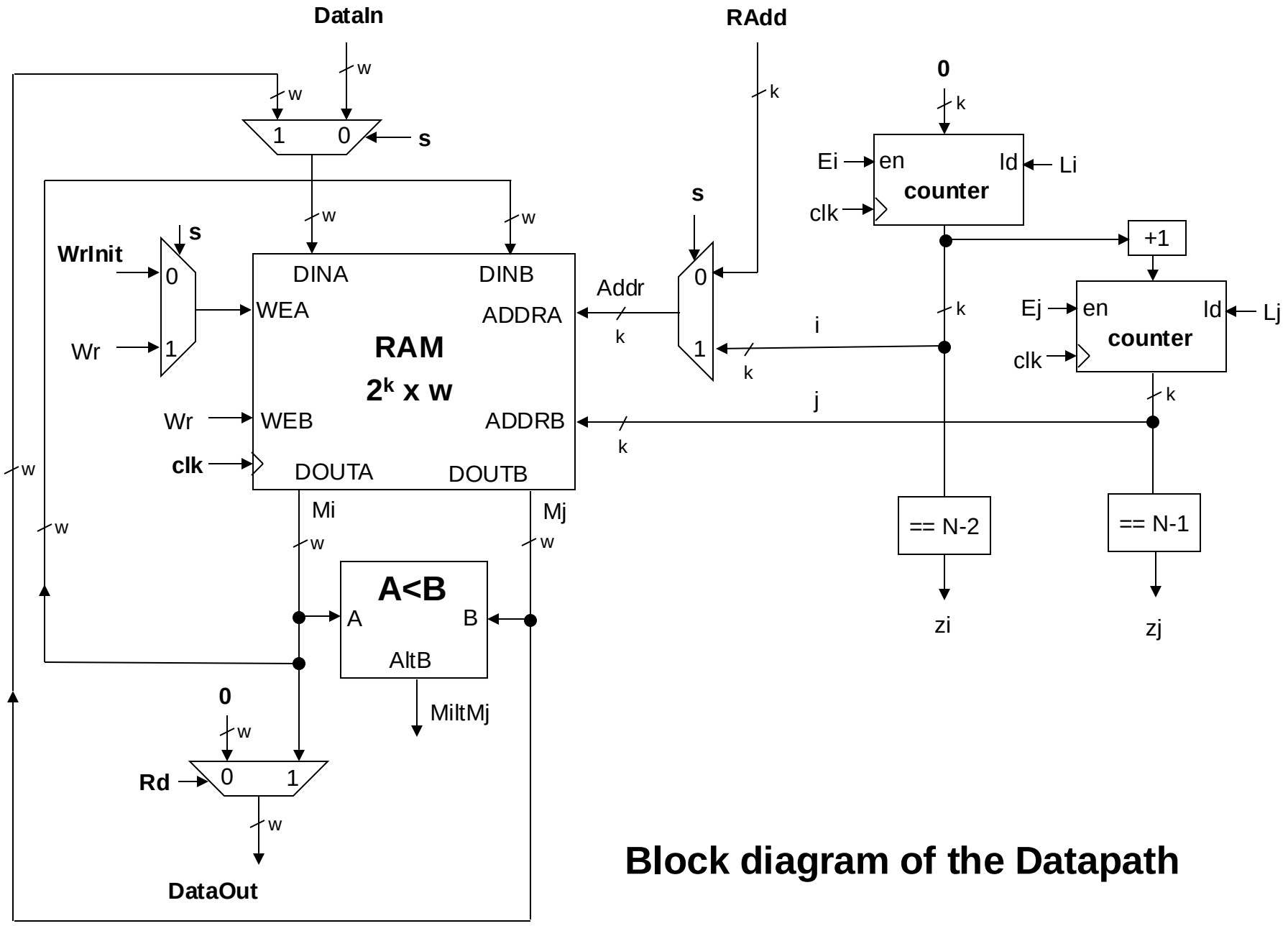
Block diagram of the Datapath

Pseudocode

wait for s=1 // read out results
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

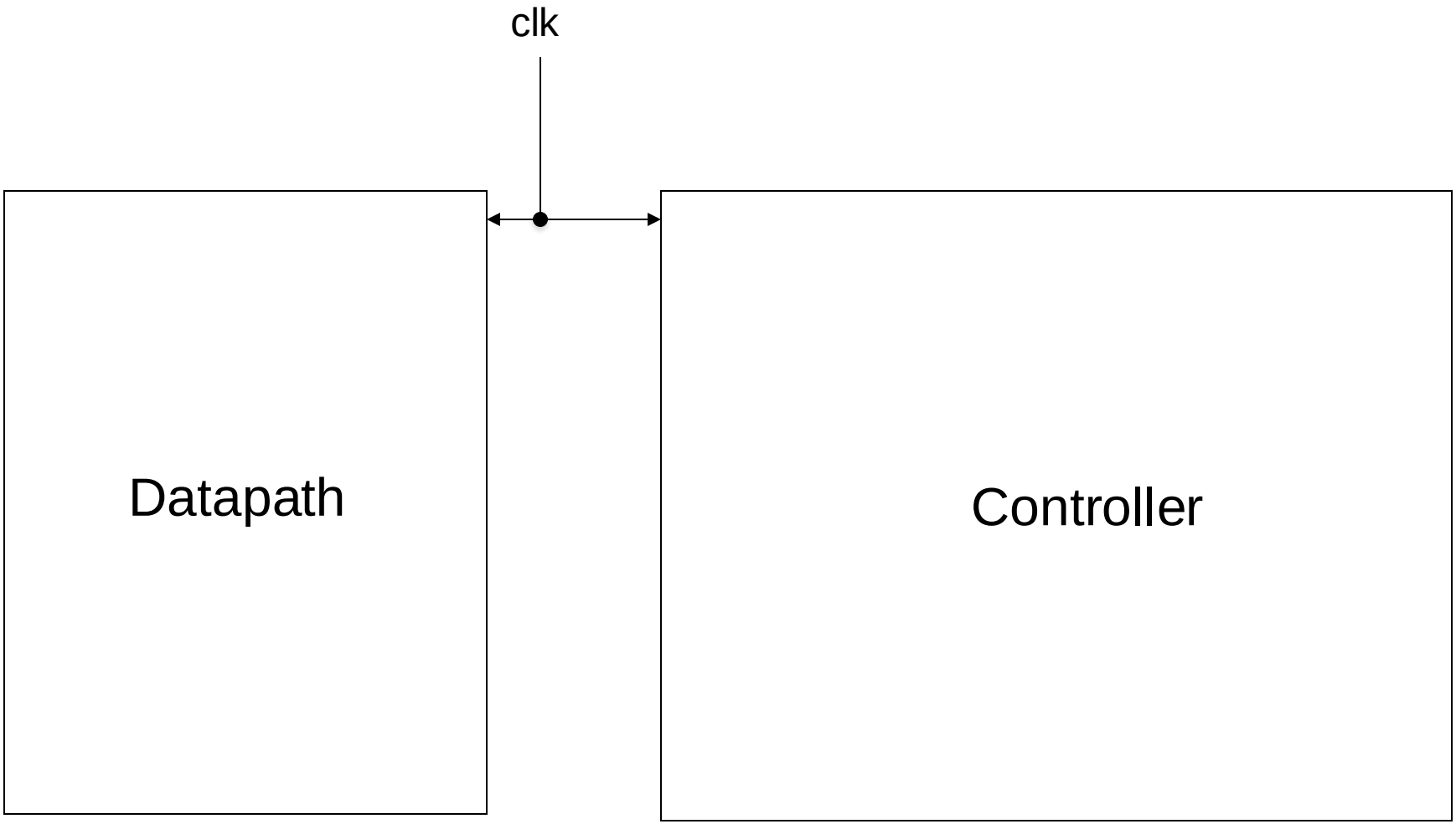
Interface


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

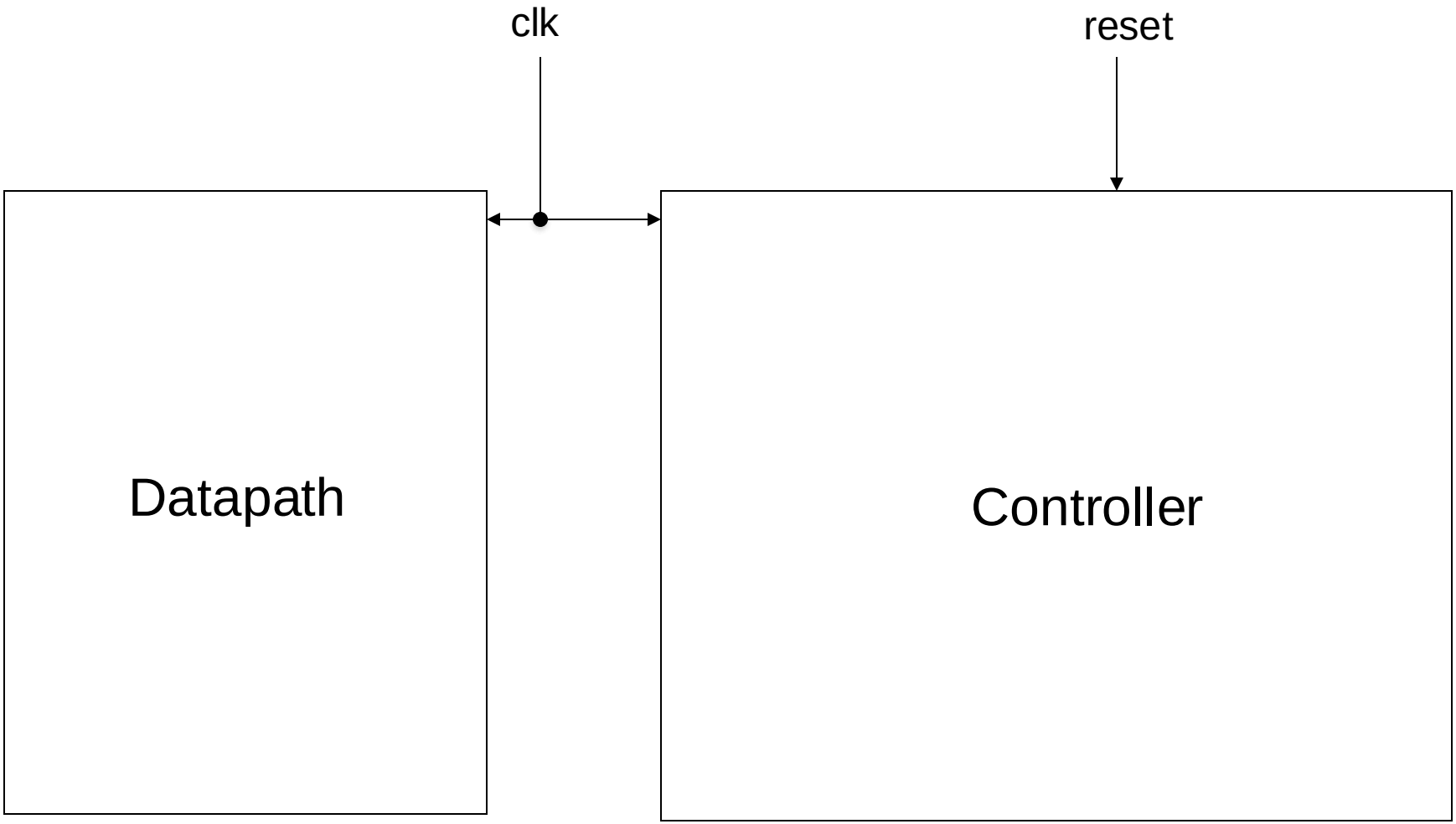


Block diagram of the Datapath

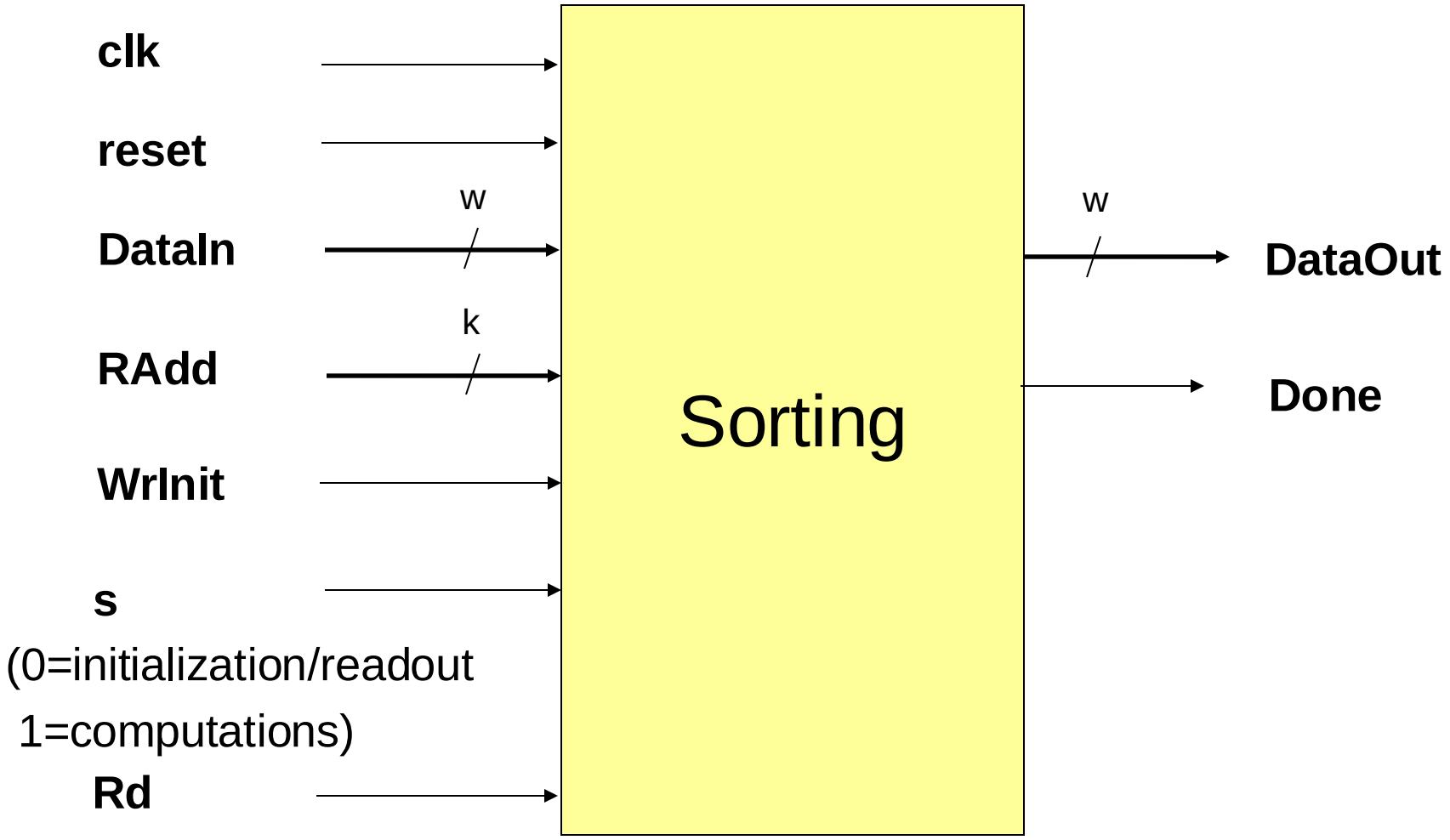
Interface with the division into the Datapath and Controller



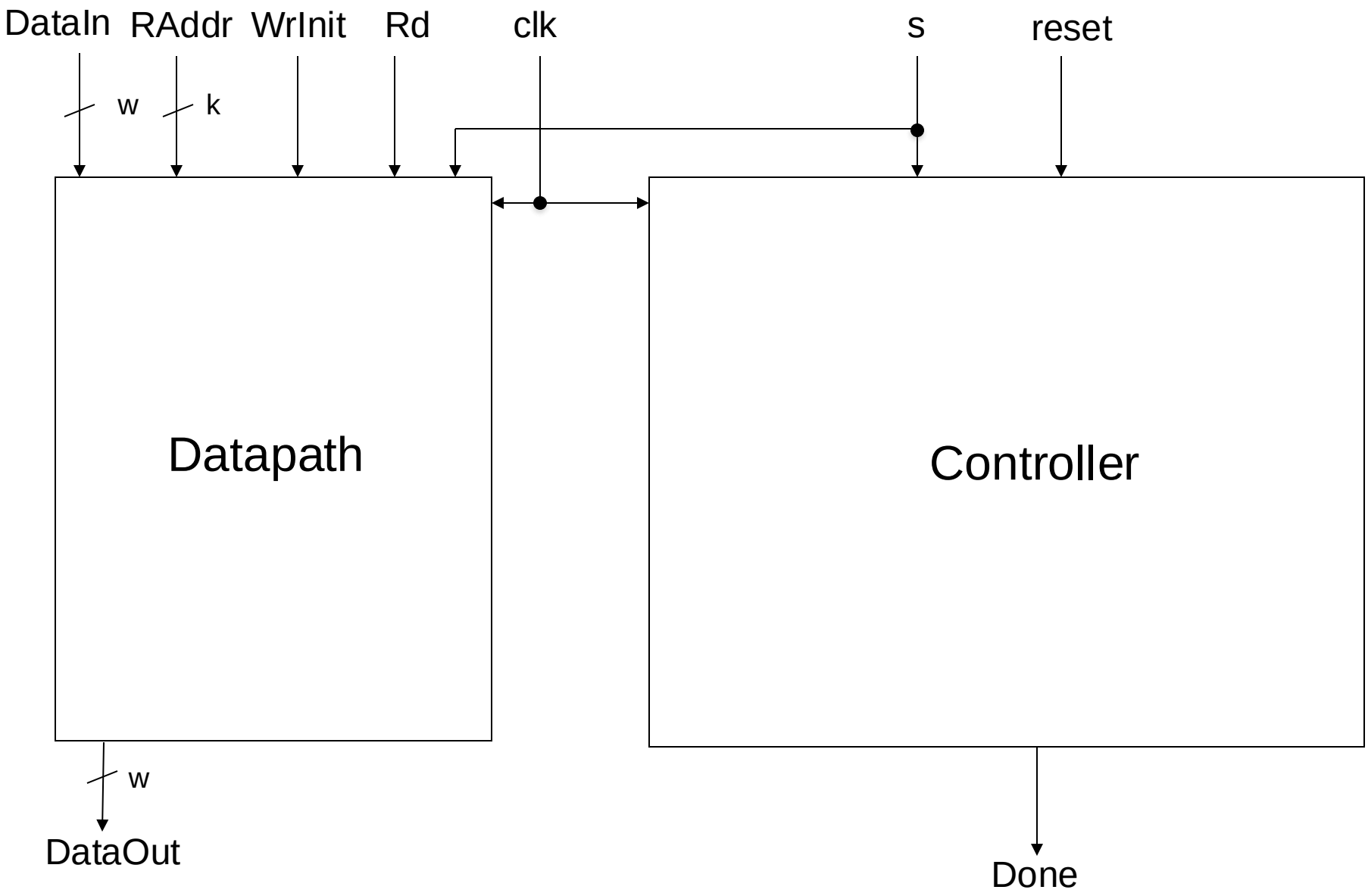
Interface with the division into the Datapath and Controller

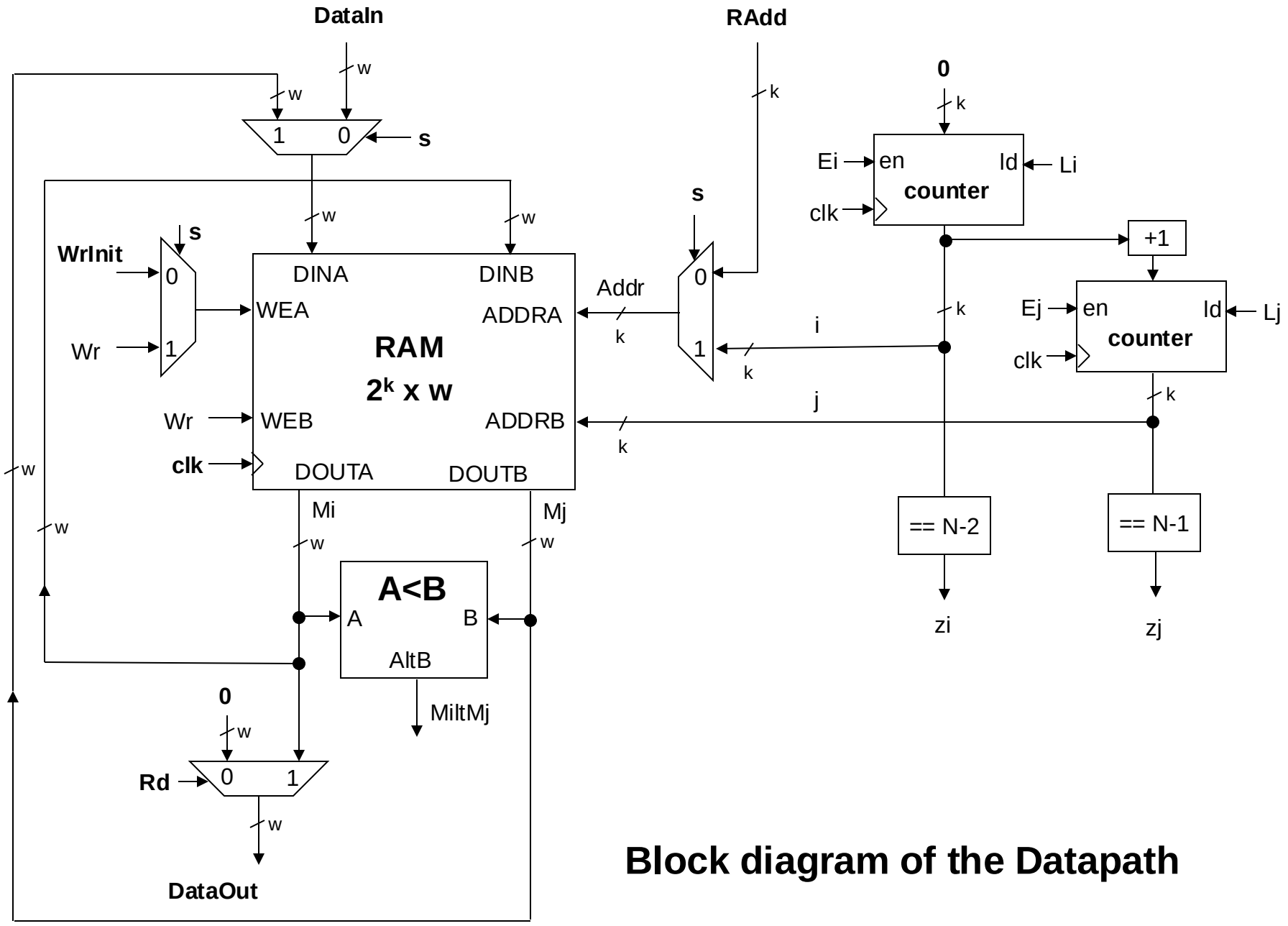


Interface


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

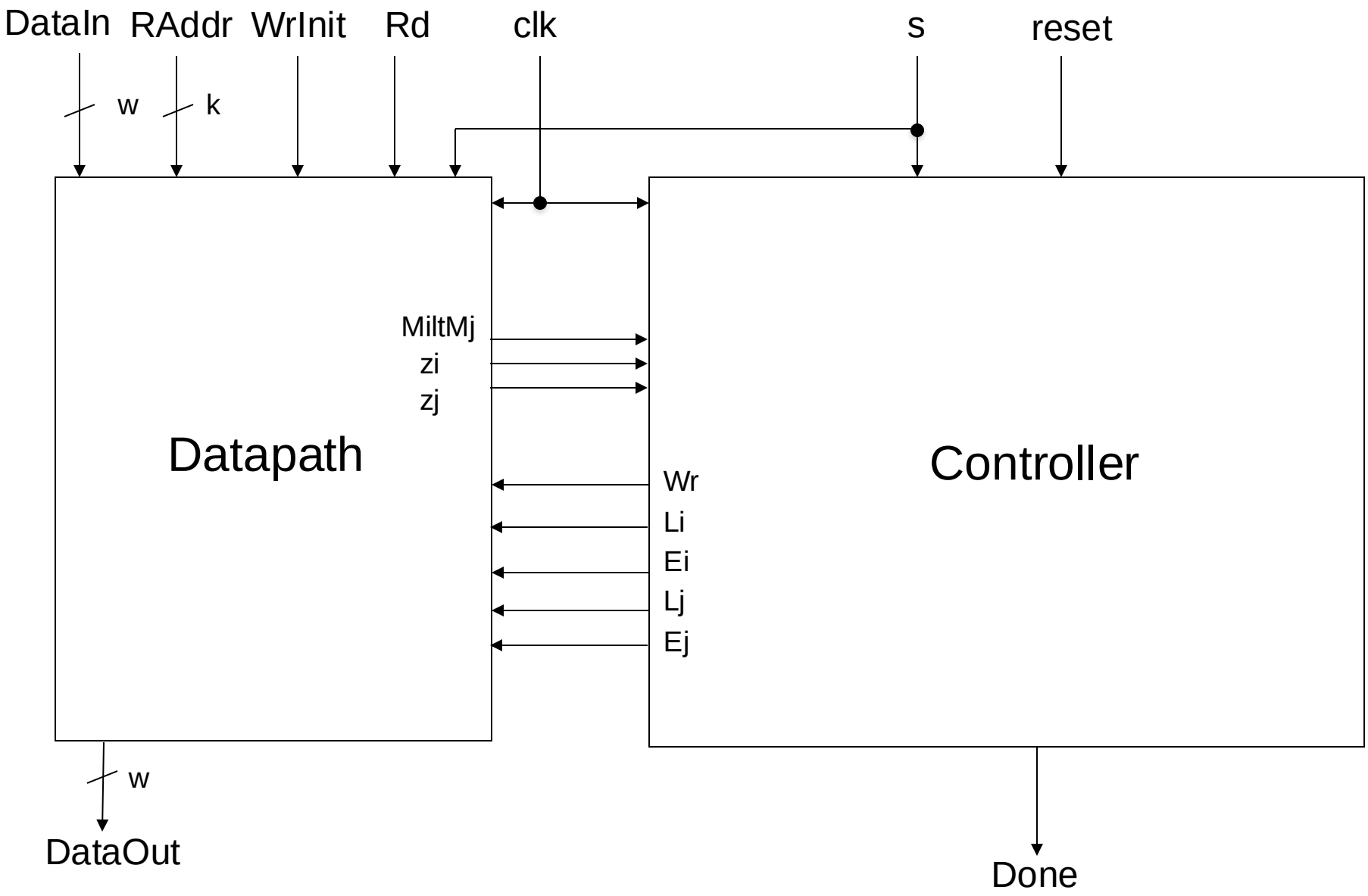
Interface with the division into the Datapath and Controller





Block diagram of the Datapath

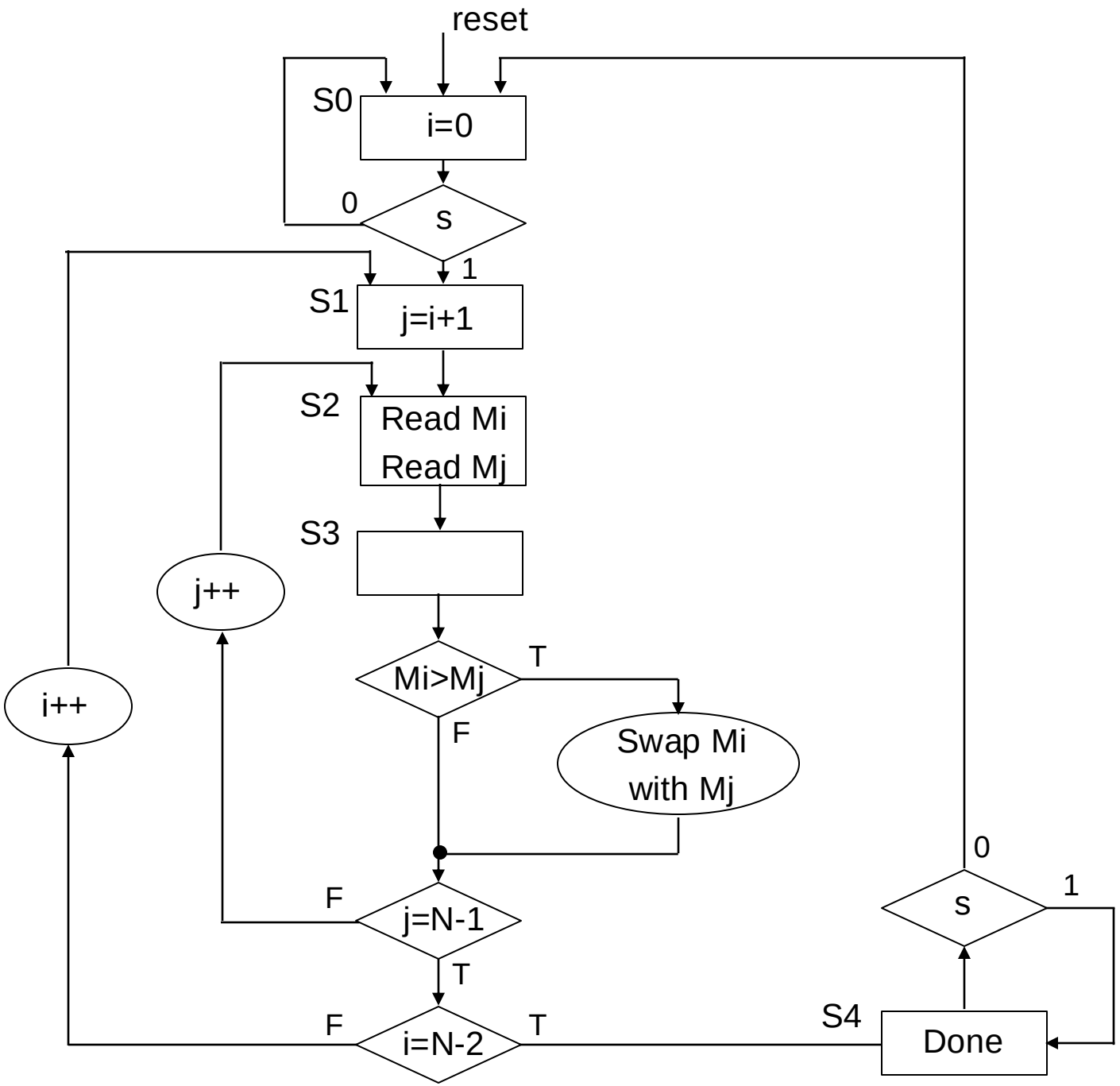
Interface with the division into the Datapath and Controller

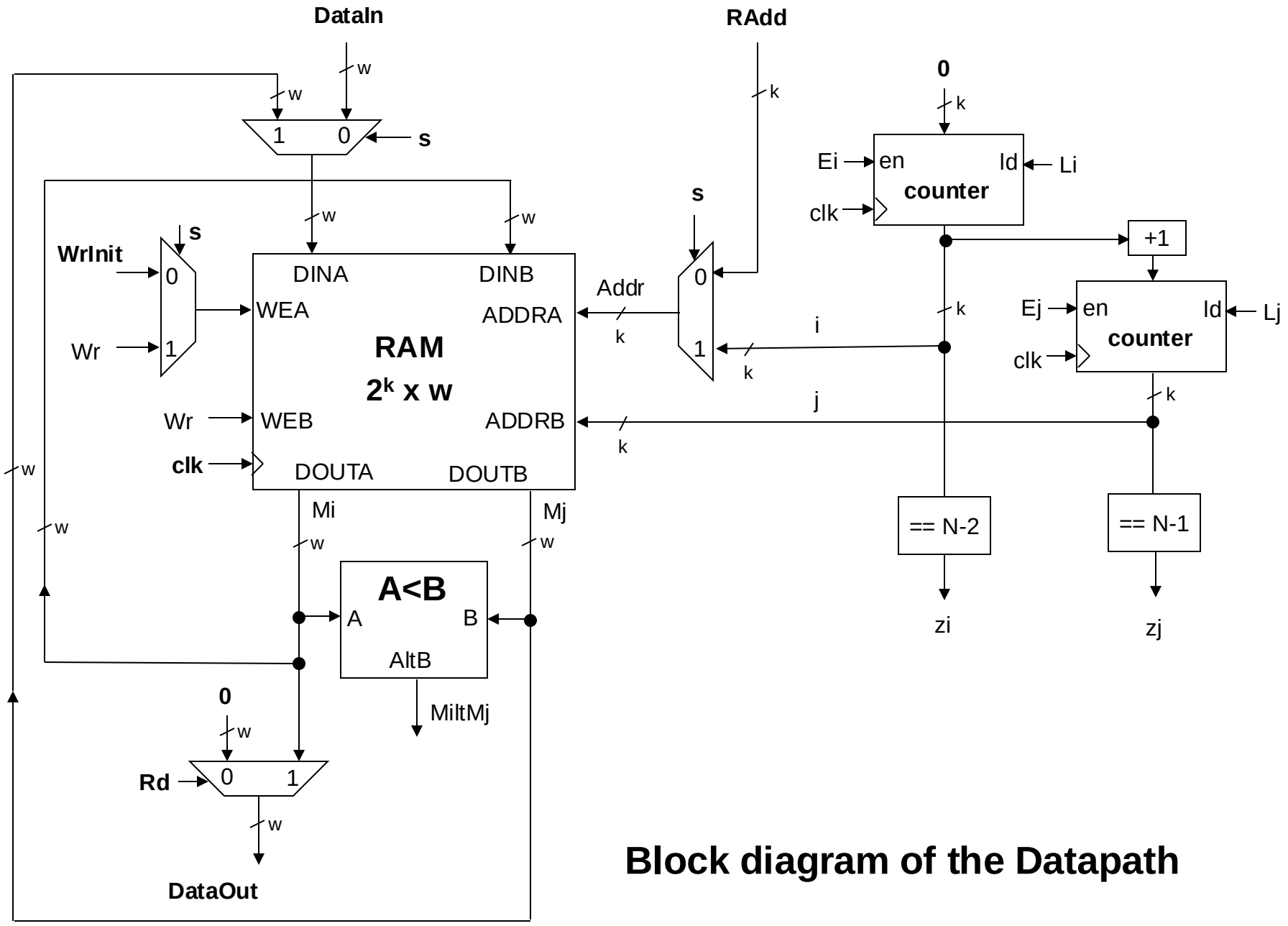


Pseudocode

wait for s=1 // read out results
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

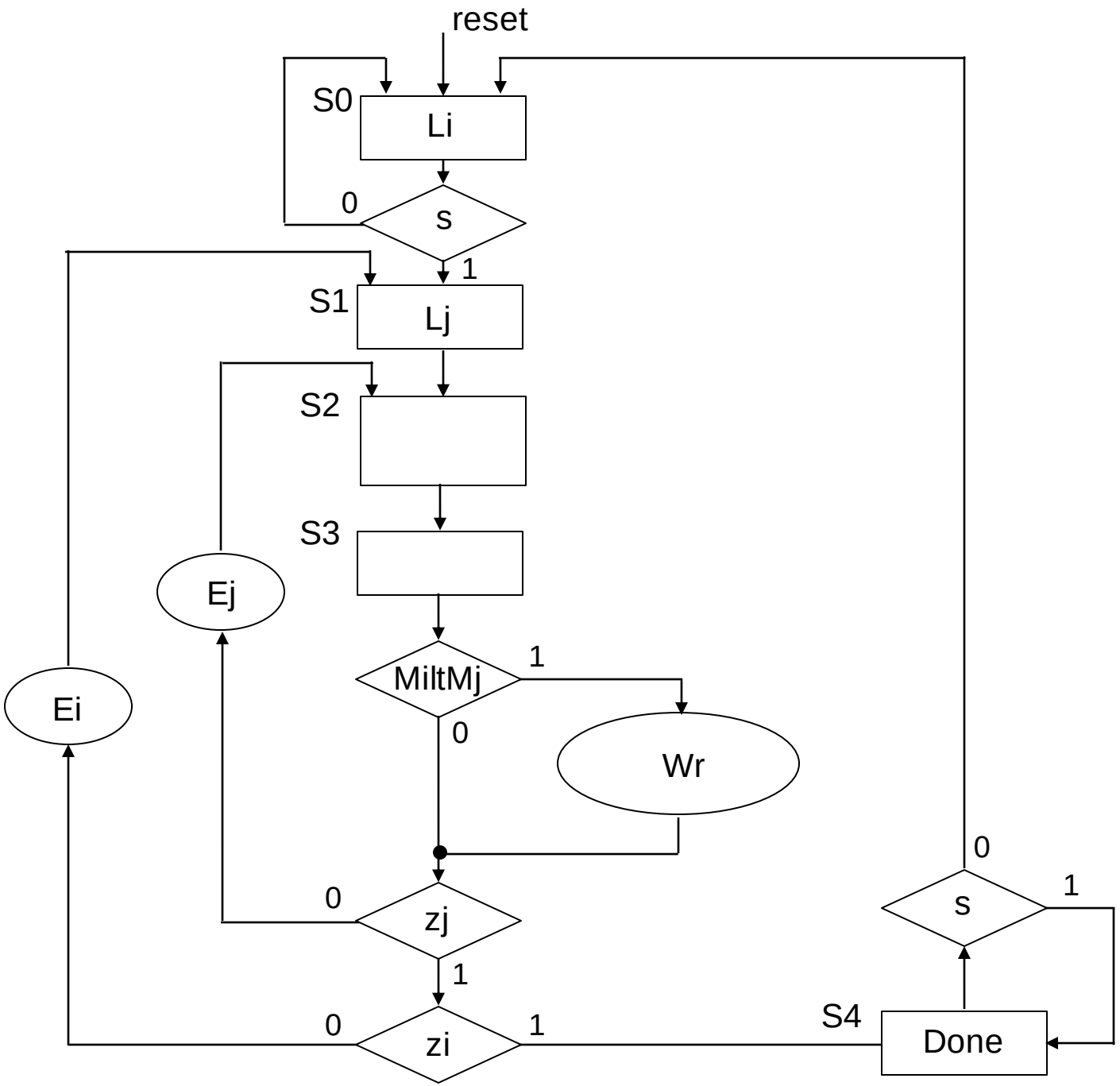
ASM Chart


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
MiltMj
1
0
Ei
Wr
0
zj
1
0
zi
1
S4
Done
0
s
1

Timing Analysis

Execution Time(N) =

Timing Analysis

For RAM with synchronous read

Execution Time(N) =

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

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

….

$1 + 1 \cdot 2 =$

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

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

$= N^2 - 1$

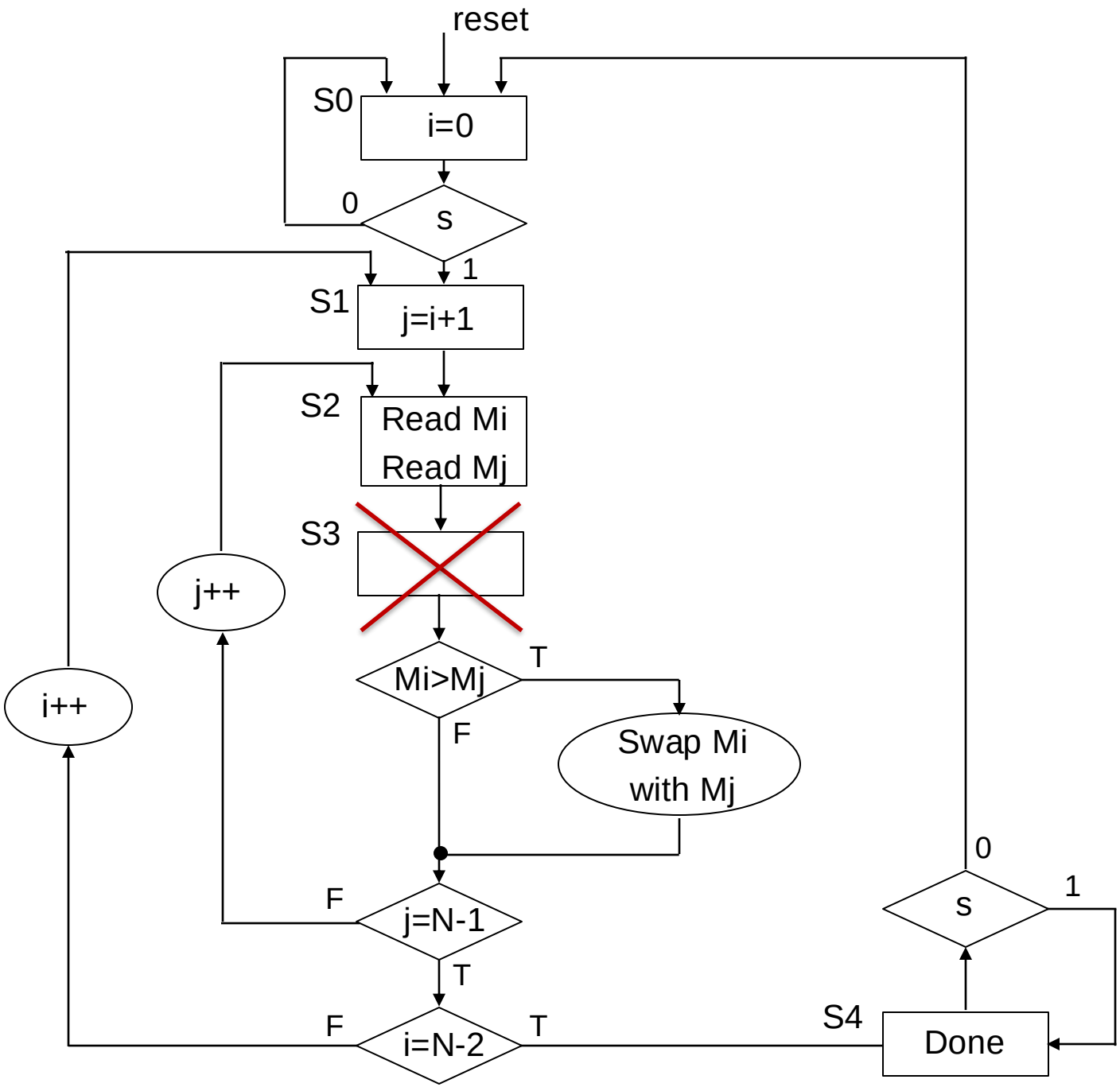
Timing Analysis

For RAM with synchronous read

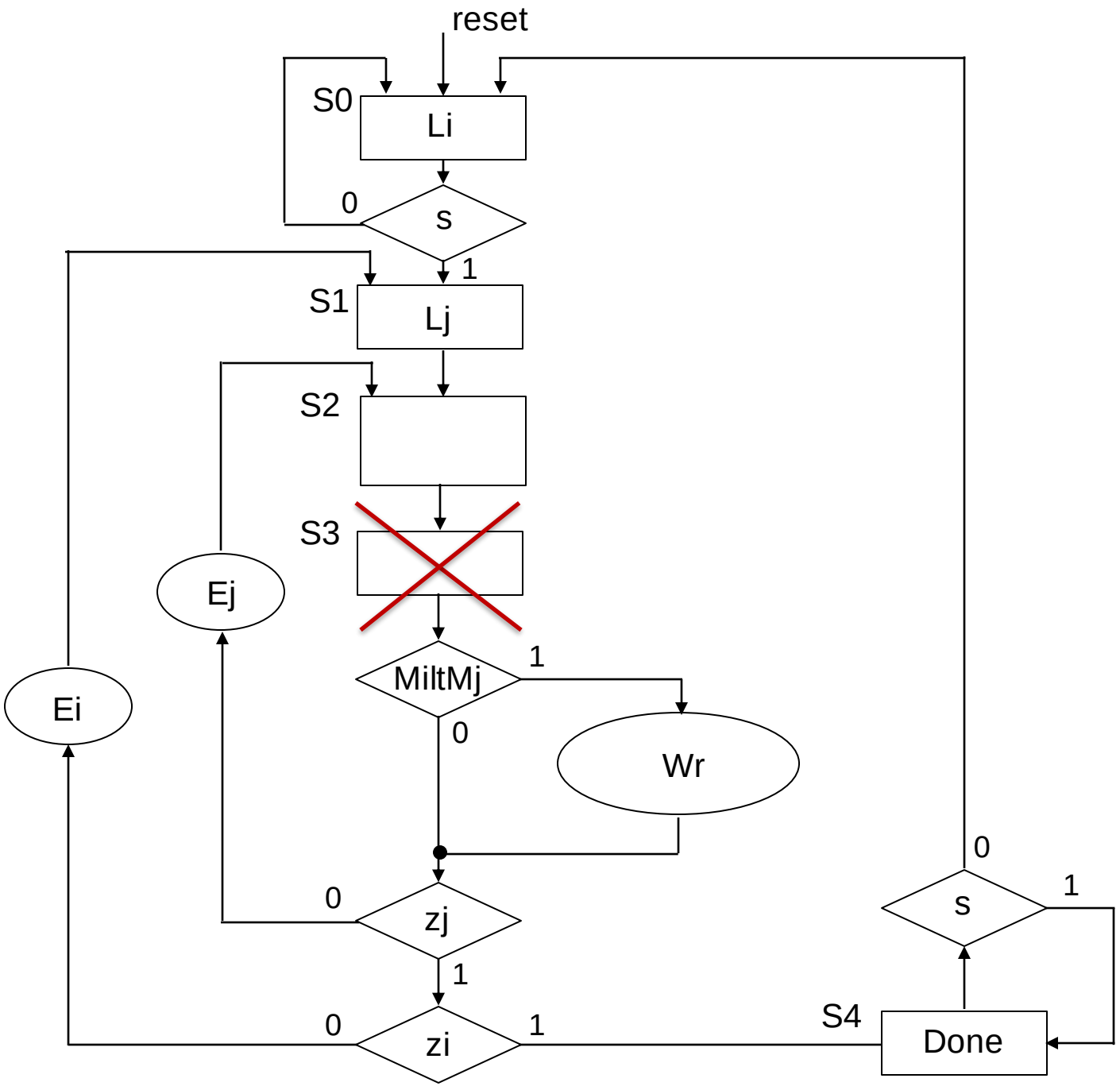
Timing Analysis

For RAM with asynchronous read

ASM Chart


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

ASM Chart


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

Timing Analysis

For RAM with asynchronous read

Timing Analysis

For RAM with asynchronous read

Execution Time(N) =

1 + (N-1) +

1 + (N-2) +

….

1 + 1 =

= (N-1) + (N-1)(N/2) =

= $N^2/2 + N/2 - 1$