ECE 448
Lecture 20
Software/Hardware Co-design Using the FPro System
Part 2:
I/O Register Map & a Wrapper of the Sorting Core

Developing a software/hardware implementation using an FPro system

Conceptual Design:

C1. Software/hardware partitioning

C2. I/O register map of the IP core

Hardware Design:

H1. Basic circuit performing the required functionality

* datapath * controller * top-level * functional verification

H2. A wrapper matching the interface of an MMIO core

*** design adjustments * coding * functional verification**

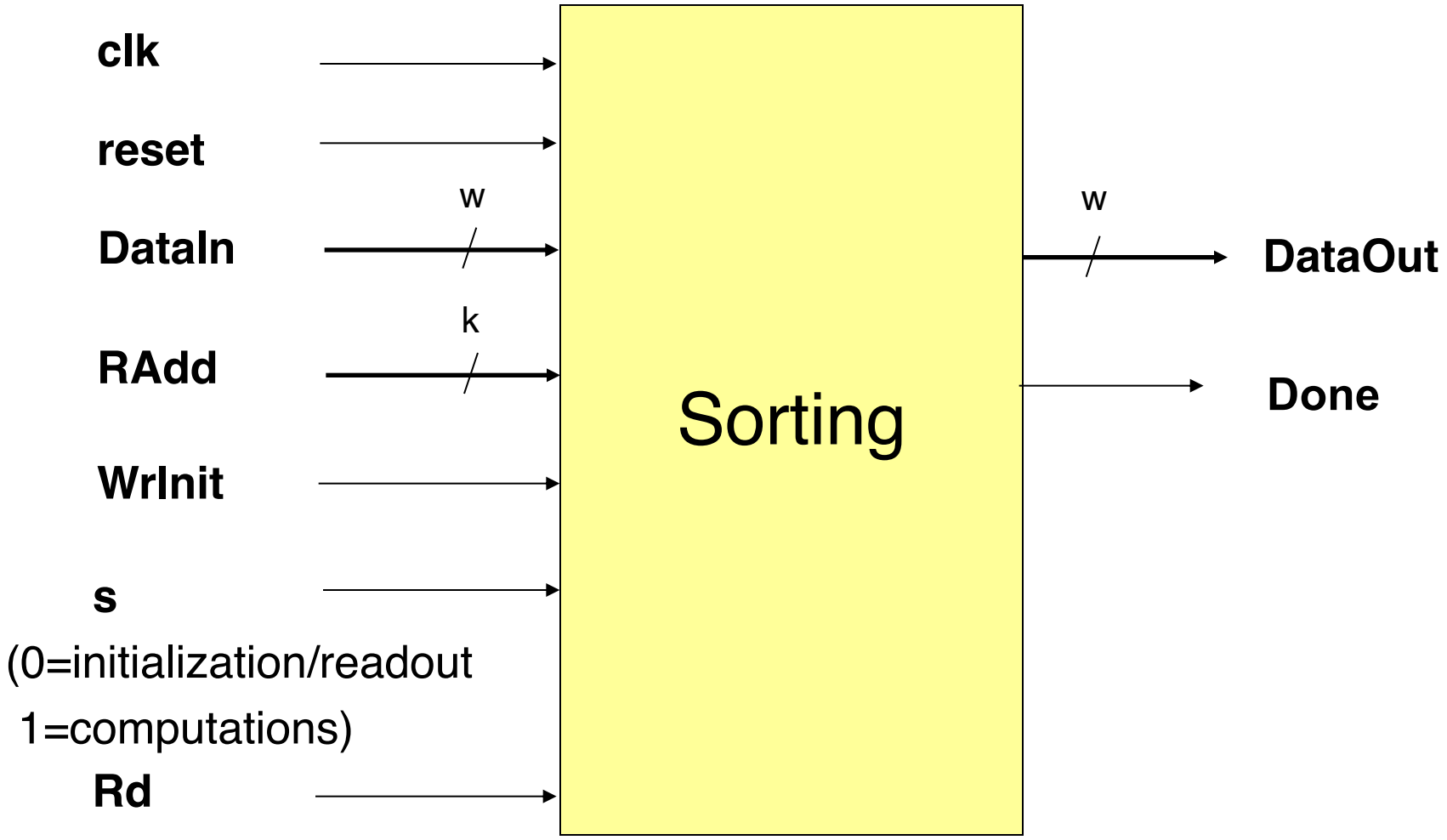
Software Design:

S1. Software driver

* declarations (.h) * implementations (.cpp) * testing

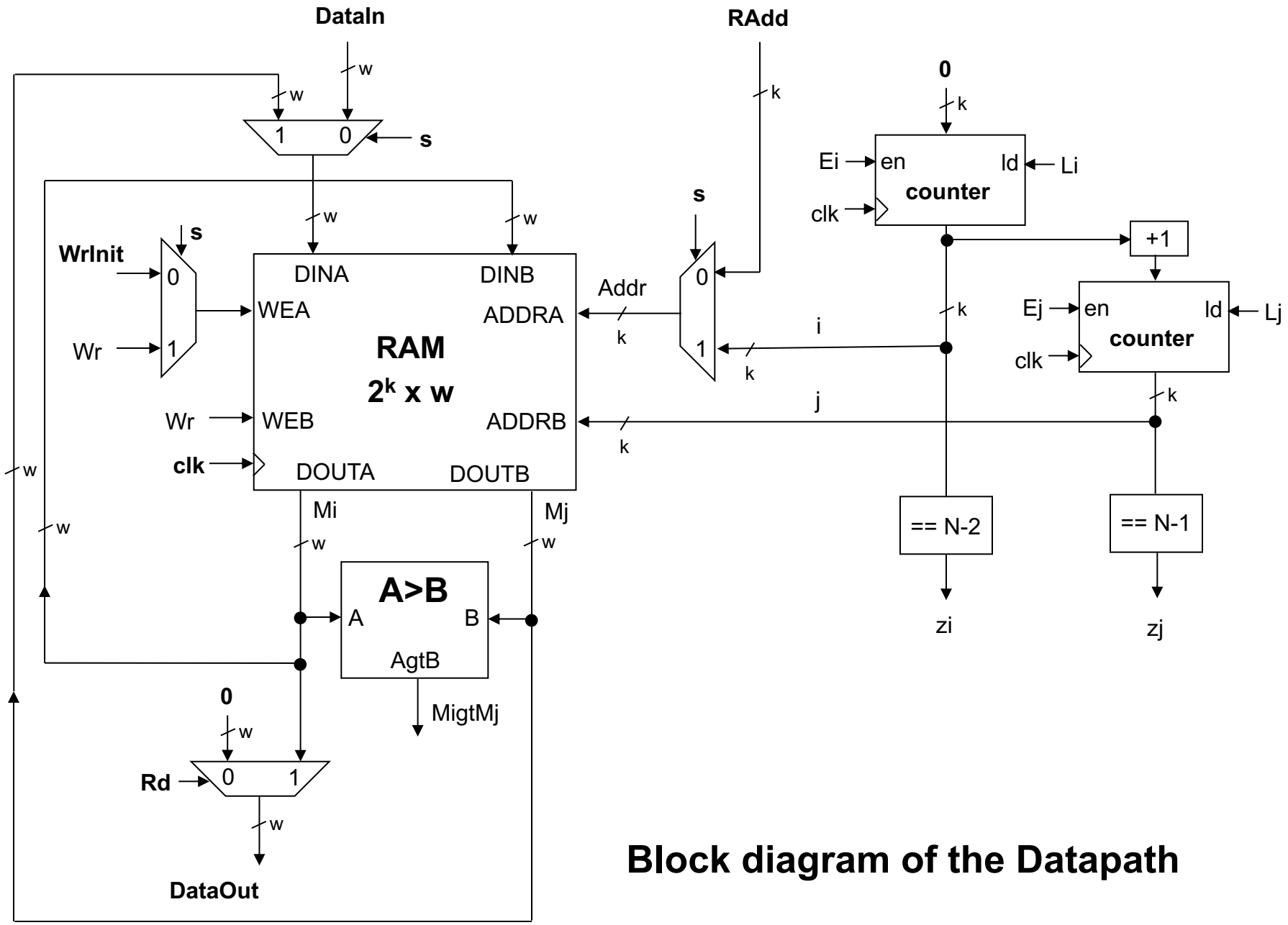
S2. Application based on functions of custom and standard cores

Sorting


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

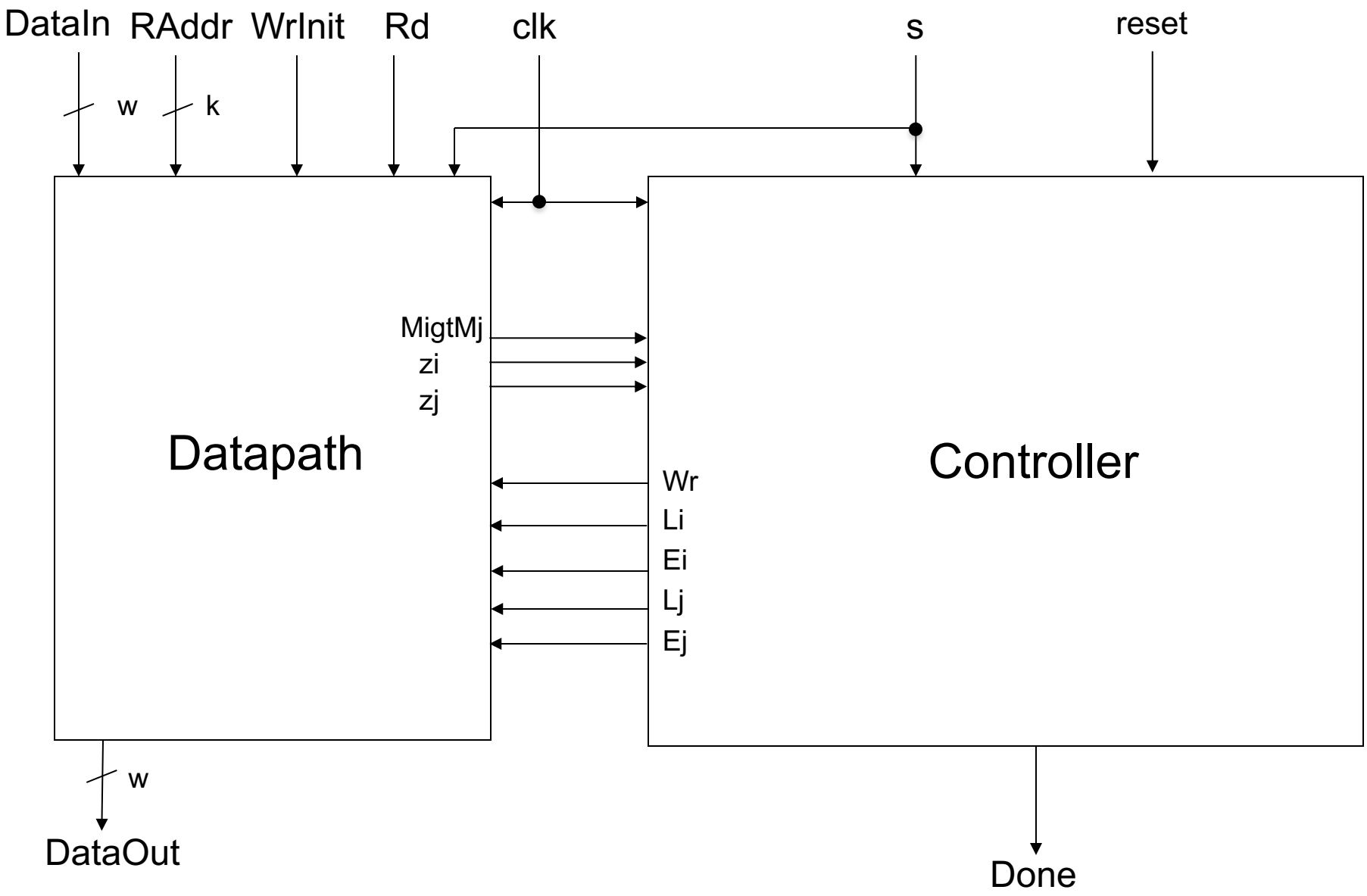
Pseudocode

```
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

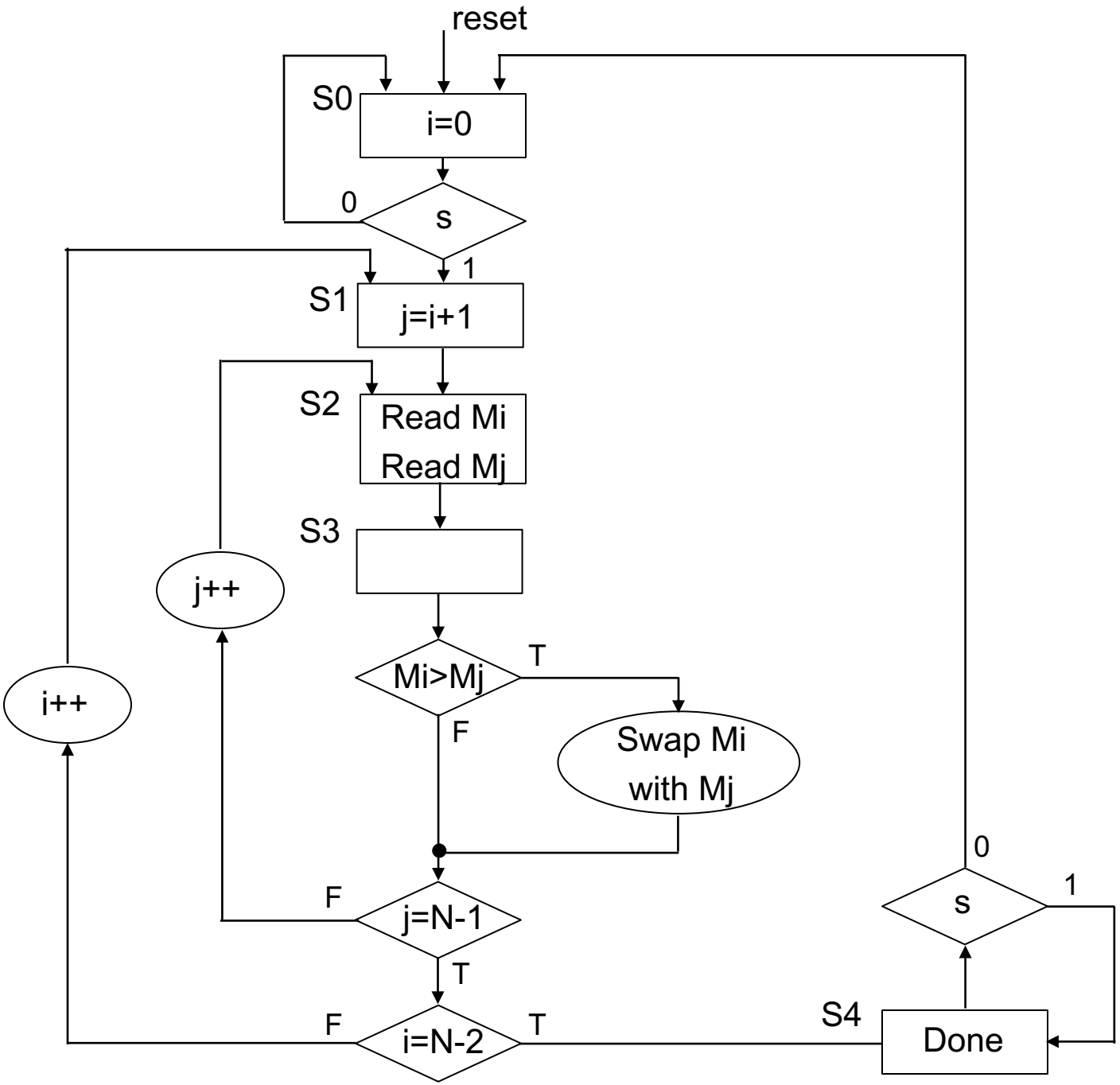
Interface with the division into the Datapath and Controller

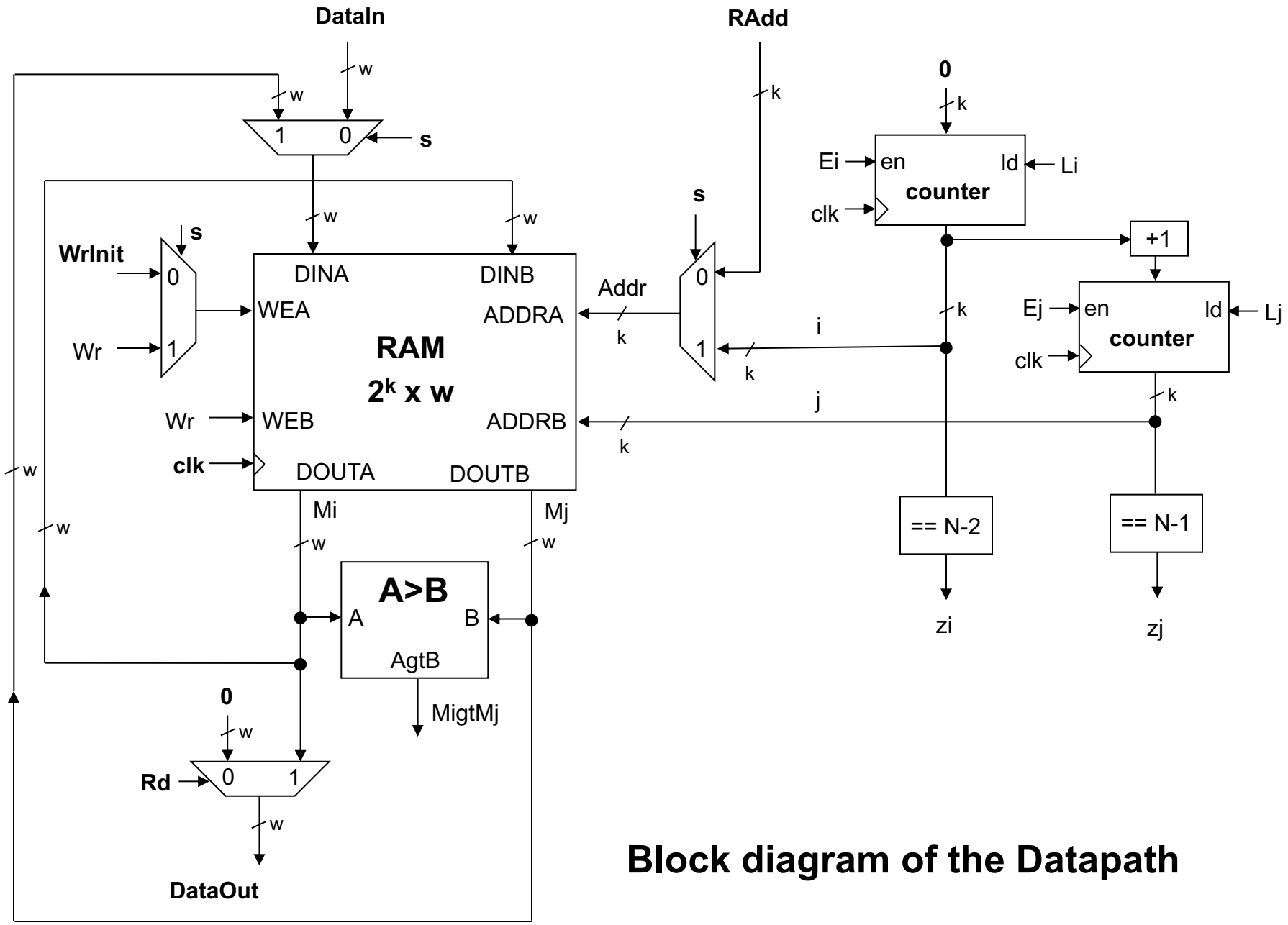


Pseudocode

```
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
```

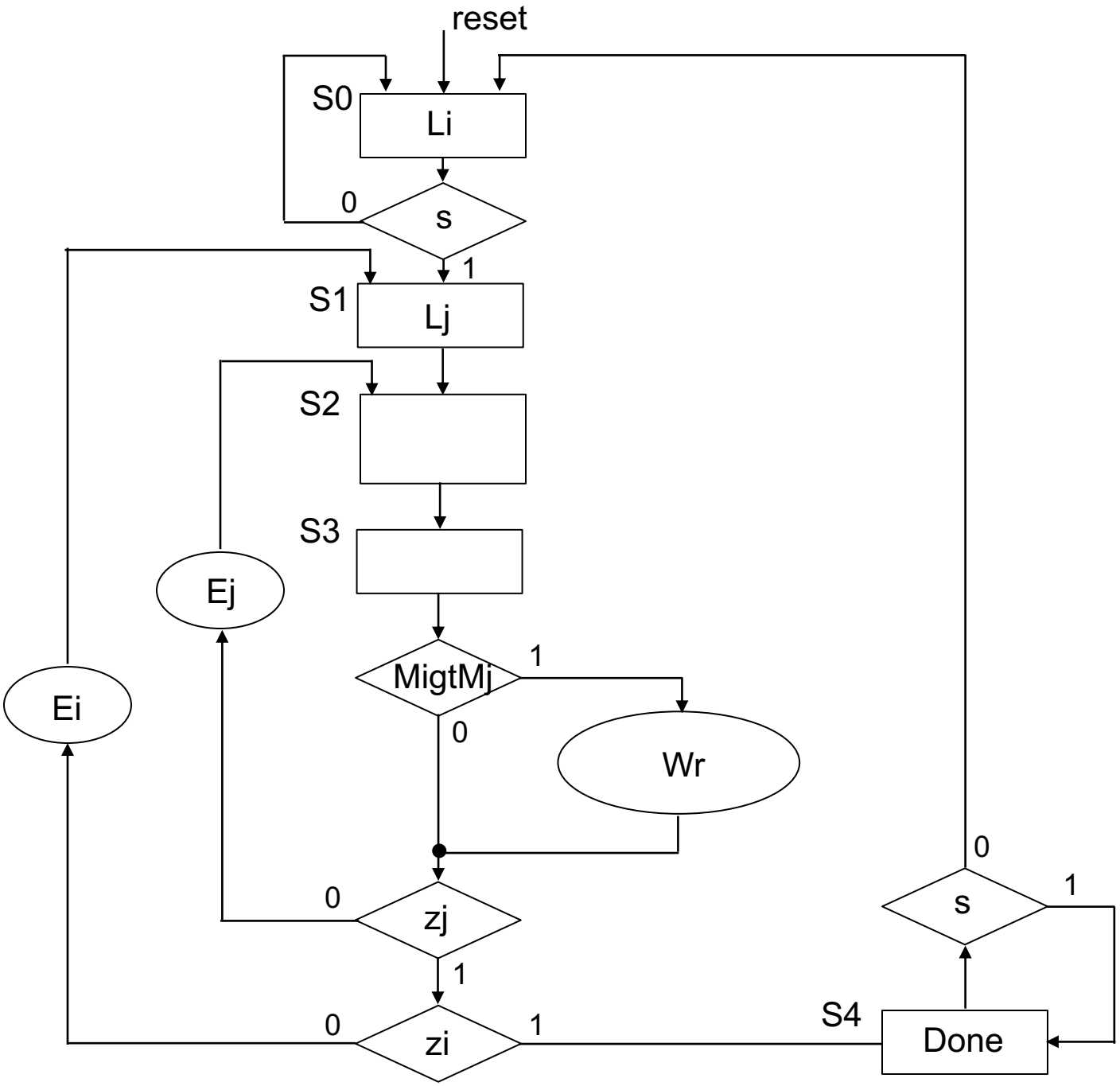
ASM Chart


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

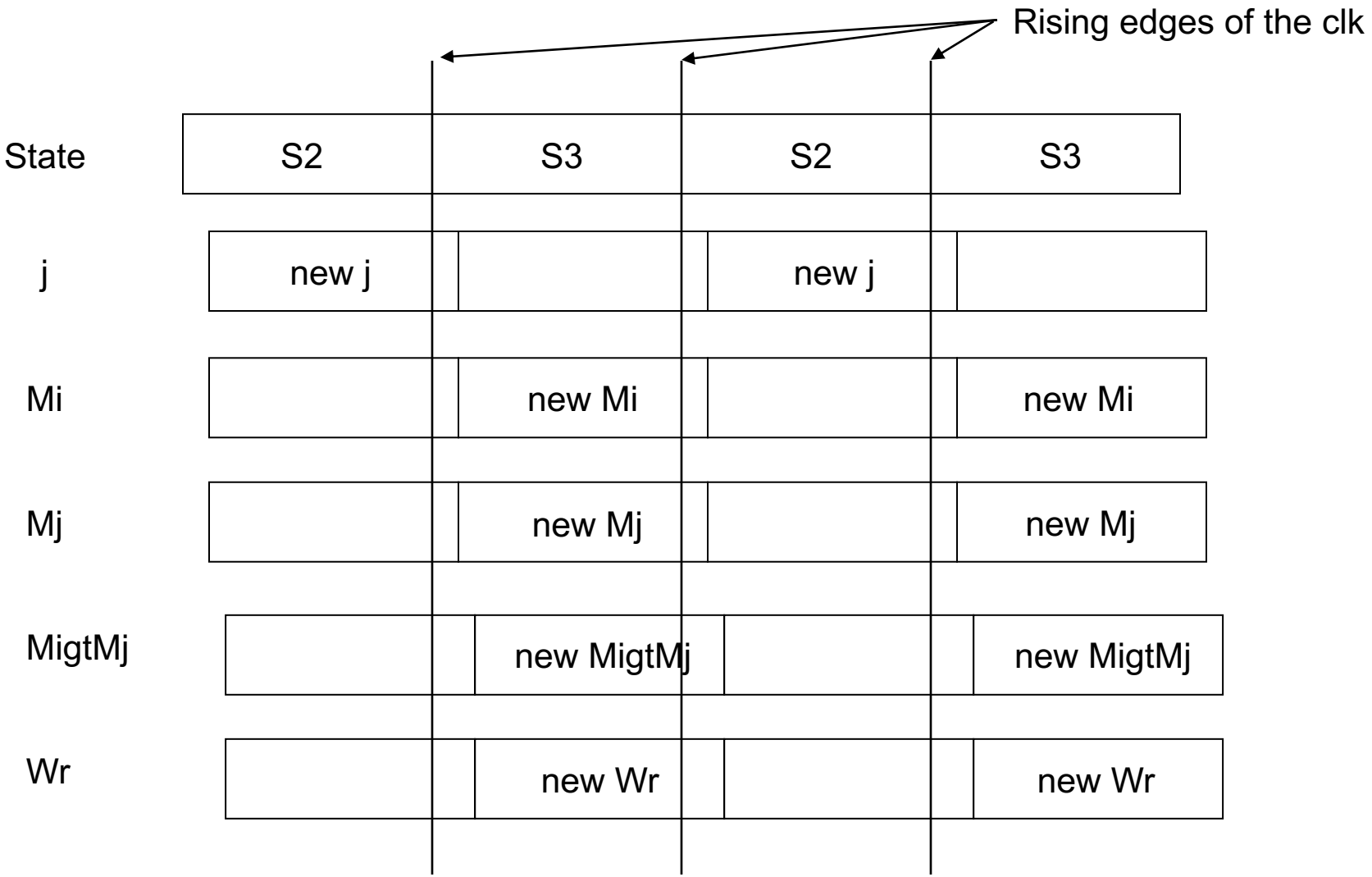


Block diagram of the Datapath

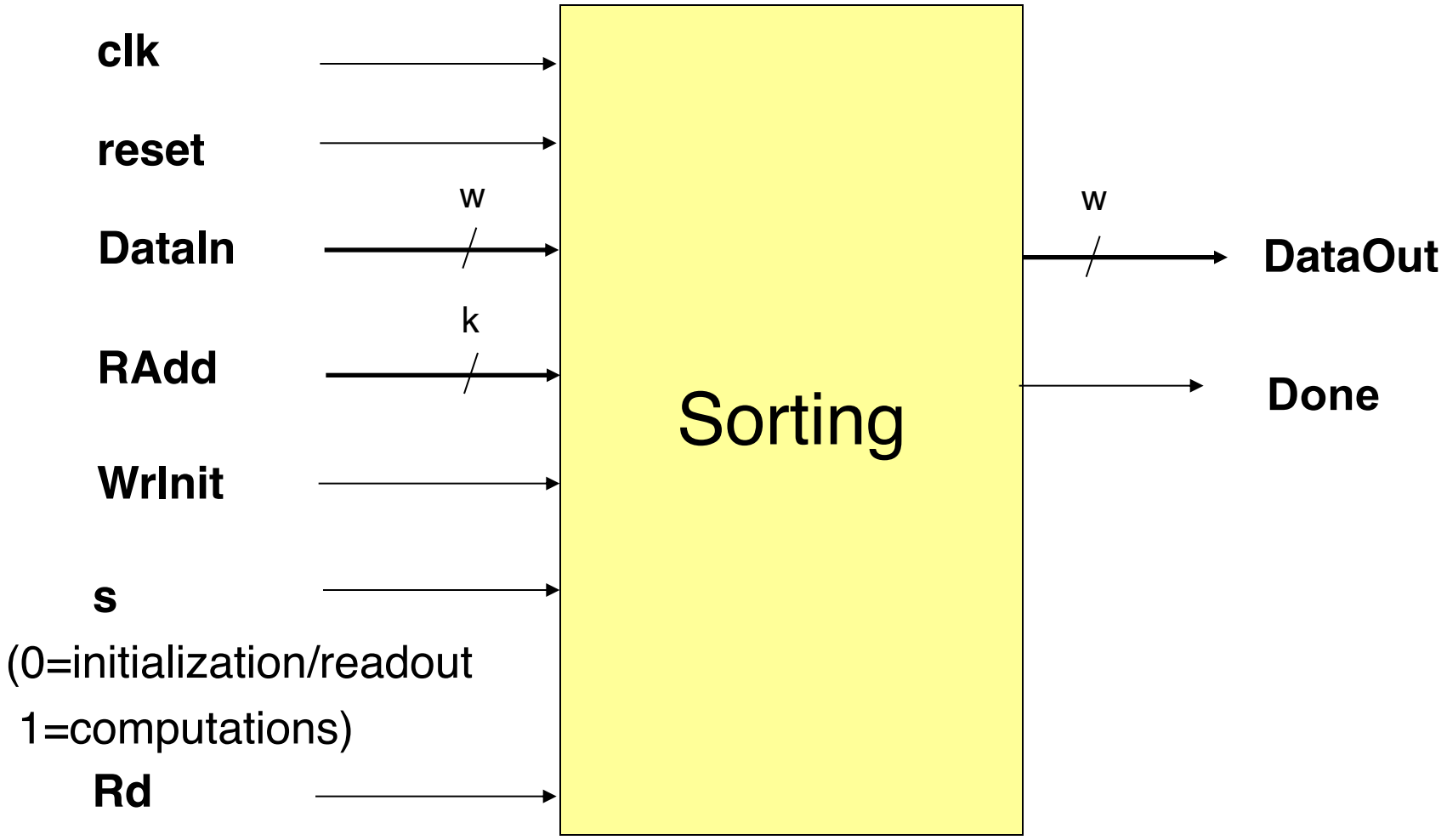
ASM Chart


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



Sorting


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

Developing a software/hardware implementation using an FPro system

Conceptual Design:

C1. Software/hardware partitioning

C2. I/O register map of the IP core

Hardware Design:

H1. Basic circuit performing the required functionality

* datapath * controller * top-level * functional verification

H2. A wrapper matching the interface of an MMIO core

*** design adjustments * coding * functional verification**

Software Design:

S1. Software driver

* declarations (.h) * implementations (.cpp) * testing

S2. Application based on functions of custom and standard cores

Setting Size of Memory
to Initialize & Sort

SW3..SW0=k=$\log_2$ N,
where N= number of elements to initialize and sort

k	N	w
4	16	8
6	64	8
9	512	16
10	1024	16
11	2048	16
12	4096	16
13	8192	16

Setting Size of Memory
to Initialize & Sort

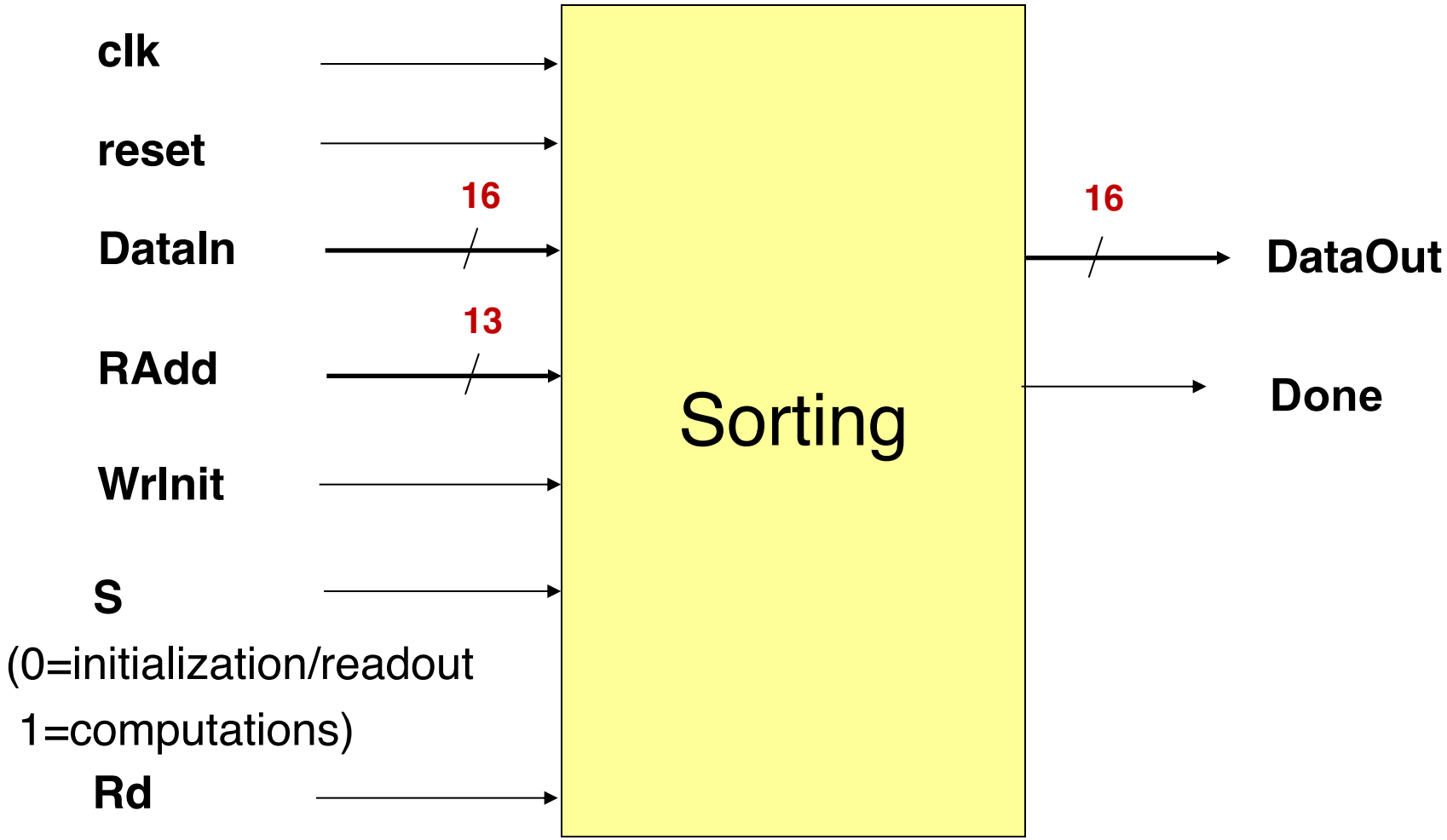
Suggested values:

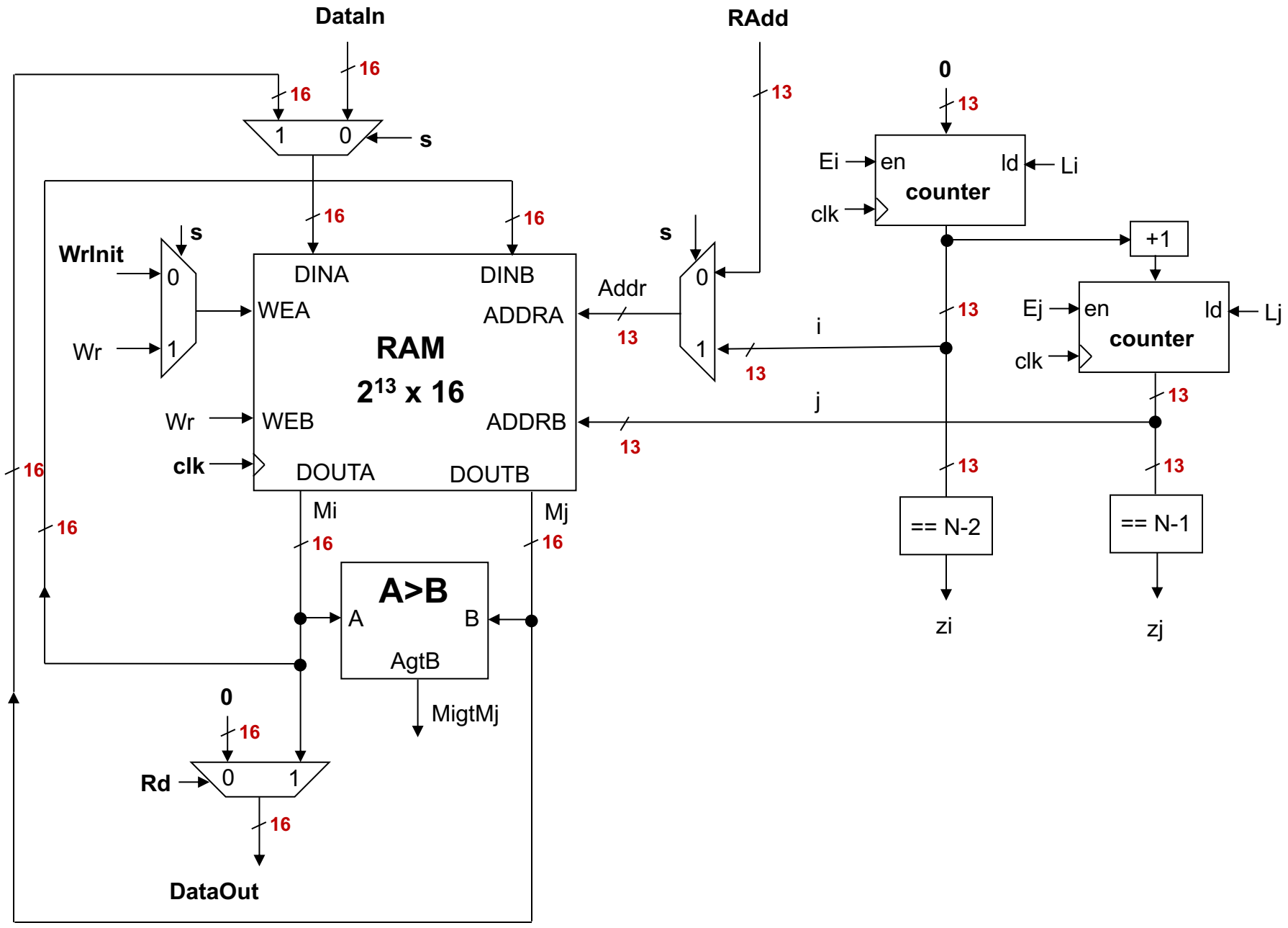
1. Debugging: k=4
   N=16 elements of the width w=8 bits

2. Demo: k=4 and k=6
   N=16 and N=64 elements of the width w=8 bits

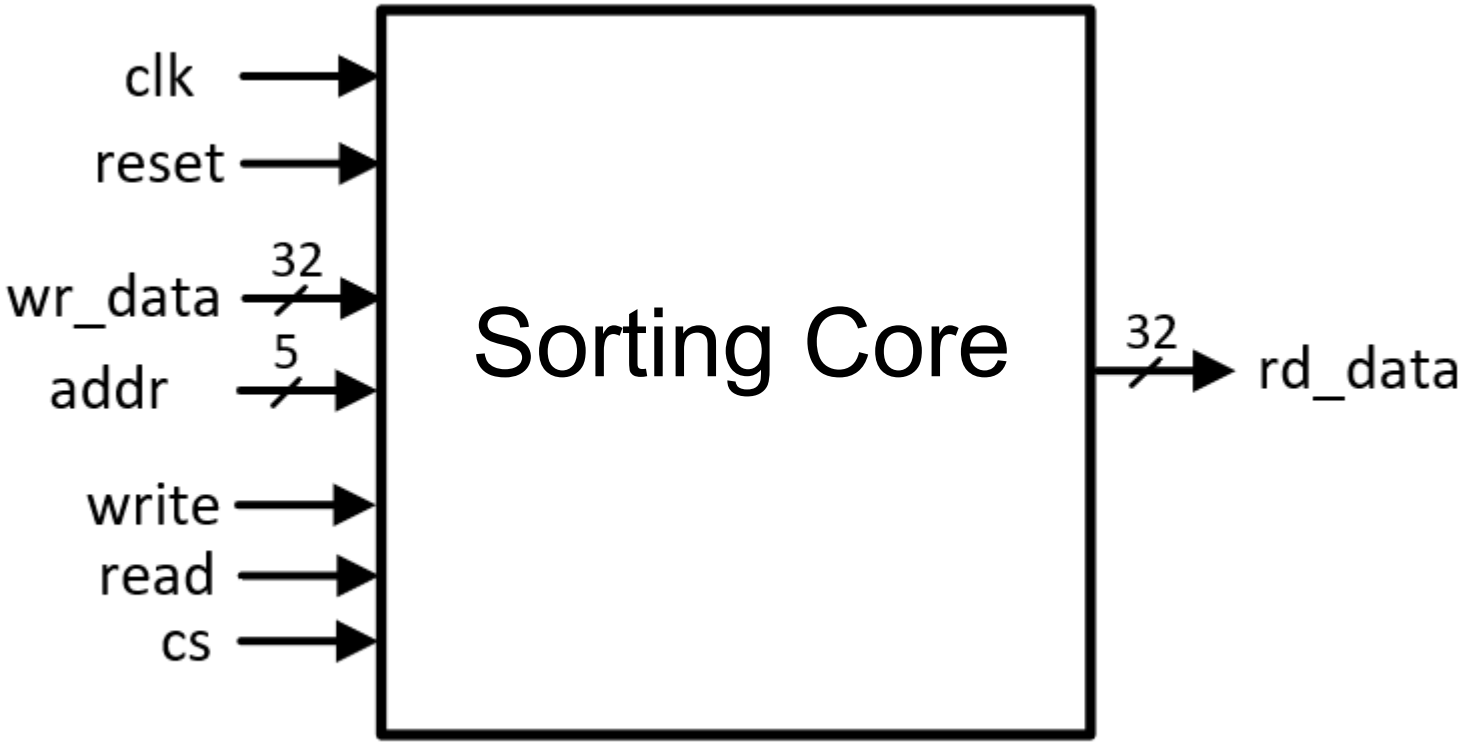
3. Timing measurements: k=9..13
   from N=512 to N=8192 elements of the width w=16 bits

Sorting

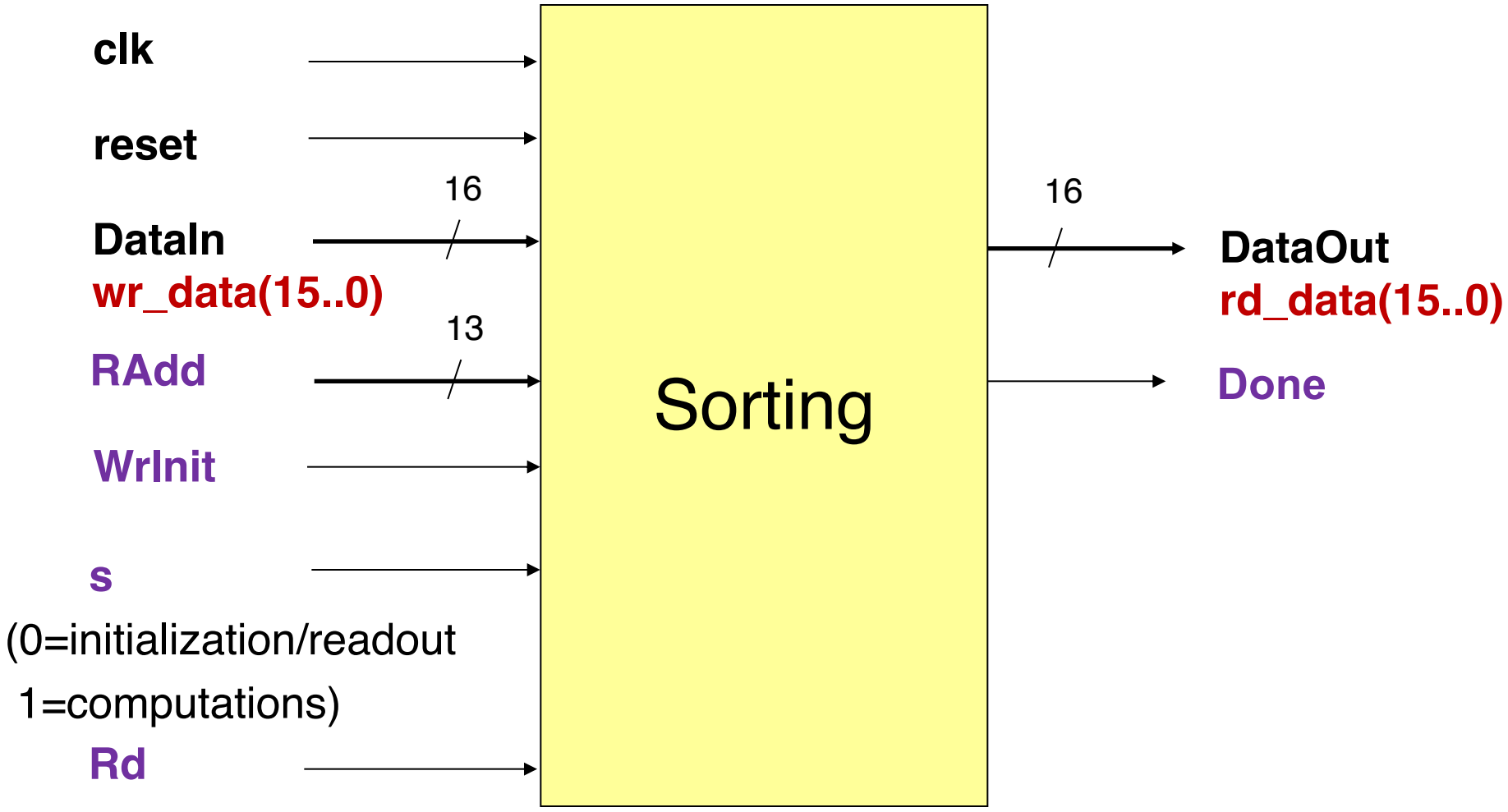

clk
reset
16
DataIn
13
RAdd
WrInit
S
(0=initialization/readout
1=computations)
Rd
Sorting
16
DataOut
Done


DataIn
16
16
1
0
s
16
16
s
WrInit
0
1
Wr
DINA
DINB
WEA
ADDRA
RAM
2^13 x 16
Wr
WEB
ADDRB
clk
DOUTA
DOUTB
16
16
Mi
Mj
16
16
A>B
A
B
AgtB
MigtMj
0
16
Rd
0
1
16
DataOut
RAdd
13
s
0
1
Addr
13
13
i
13
j
13
0
13
Ei
en
ld
Li
clk
counter
+1
Ej
en
ld
Lj
clk
counter
13
13
13
== N-2
== N-1
zi
zj

Interface of every MMIO Core



Sorting


clk
reset
16
DataIn
wr_data(15..0)
13
RAdd
WrInit
s
(0=initialization/readout
1=computations)
Rd
Sorting
16
DataOut
rd_data(15..0)
Done

Developing a software/hardware implementation using an FPro system

Conceptual Design:

C1. Software/hardware partitioning

C2. I/O register map of the IP core

Hardware Design:

H1. Basic circuit performing the required functionality

* datapath * controller * top-level * functional verification

H2. A wrapper matching the interface of an MMIO core

* design adjustments * coding * functional verification

Software Design:

S1. Software driver

* declarations (.h) * implementations (.cpp) * testing

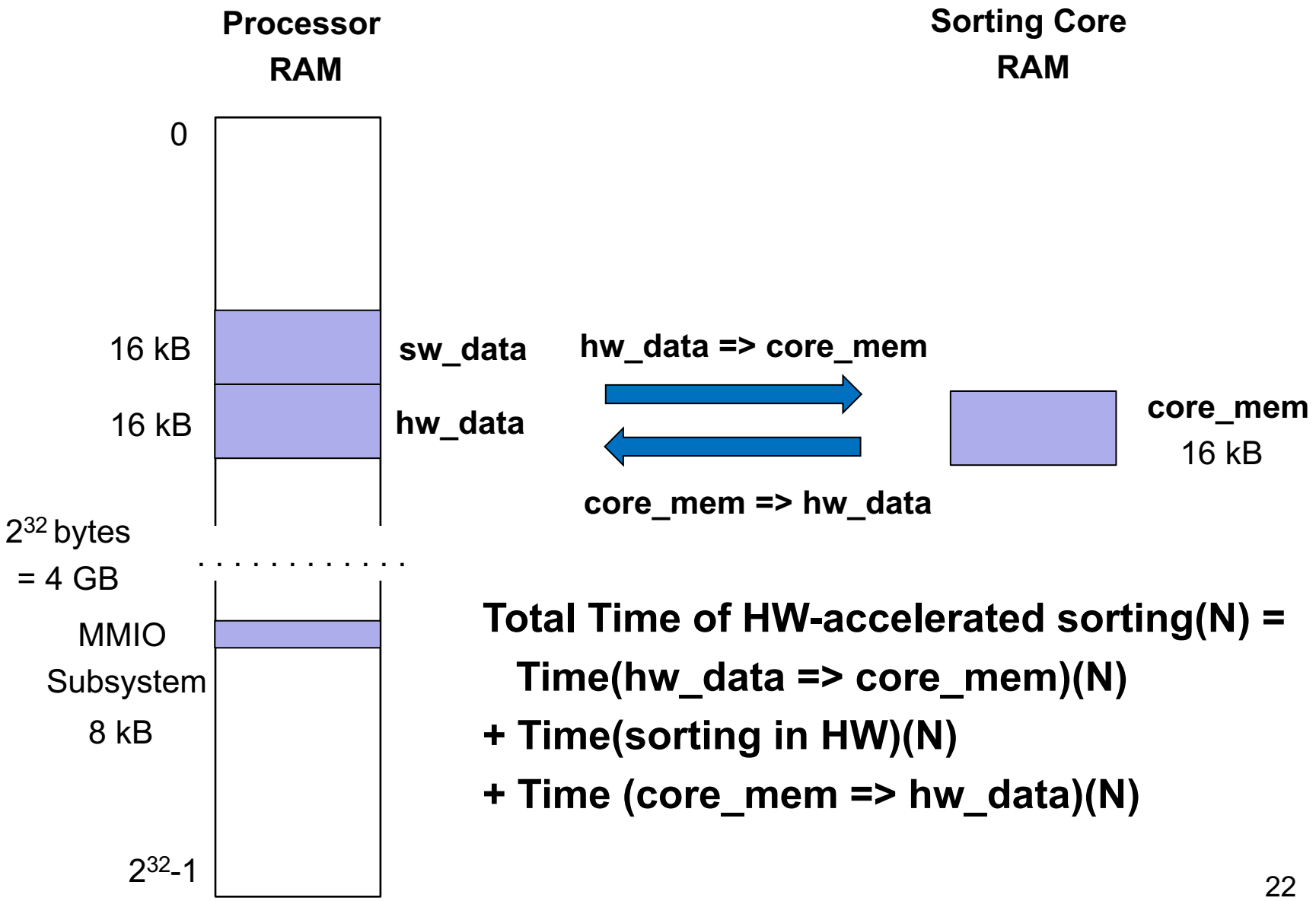
S2. Application based on functions of custom and standard cores

Setting Size of Memory
to Initialize & Sort

SW3..SW0=k=$\log_2$ N,
where N= number of elements to initialize and sort

k	N	w
4	16	8
6	64	8
9	512	16
10	1024	16
11	2048	16
12	4096	16
13	8192	16

Data Transfer



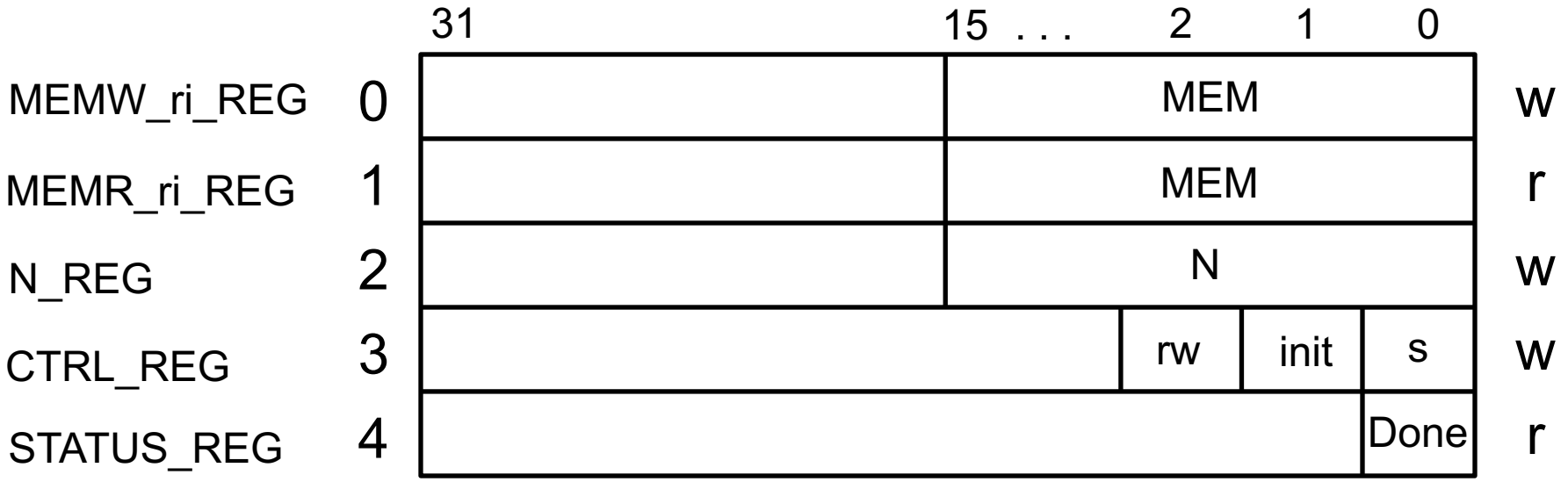
Total Time of HW-accelerated sorting(N) =

Time(hw_data => core_mem)(N)

+ Time(sorting in HW)(N)

+ Time (core_mem => hw_data)(N)

I/O Register Map of the Sorting Core



Writing to MEMW_ri_REG : Writing to MEM[ri] & ri++

Reading from MEMR_ri_REG : Reading from MEM[ri] & ri++

Writing to NREG_REG : Initializing N

Reading from STATUS_REG : Reading Done

Writing to CRTL_REG:

rw	init	s	
0	0	1	Set s=1 : computations
0	0	0	Set s=0 : initialization/readout
1	1	0	Set ri=0 and WrInit_r=1 : start initialization
0	1	0	Set ri=0 and Rd_r=1 : start readout

Developing a software/hardware implementation using an FPro system

Conceptual Design:

C1. Software/hardware partitioning

C2. I/O register map of the IP core

Hardware Design:

H1. Basic circuit performing the required functionality

* datapath * controller * top-level * functional verification

H2. A wrapper matching the interface of an MMIO core

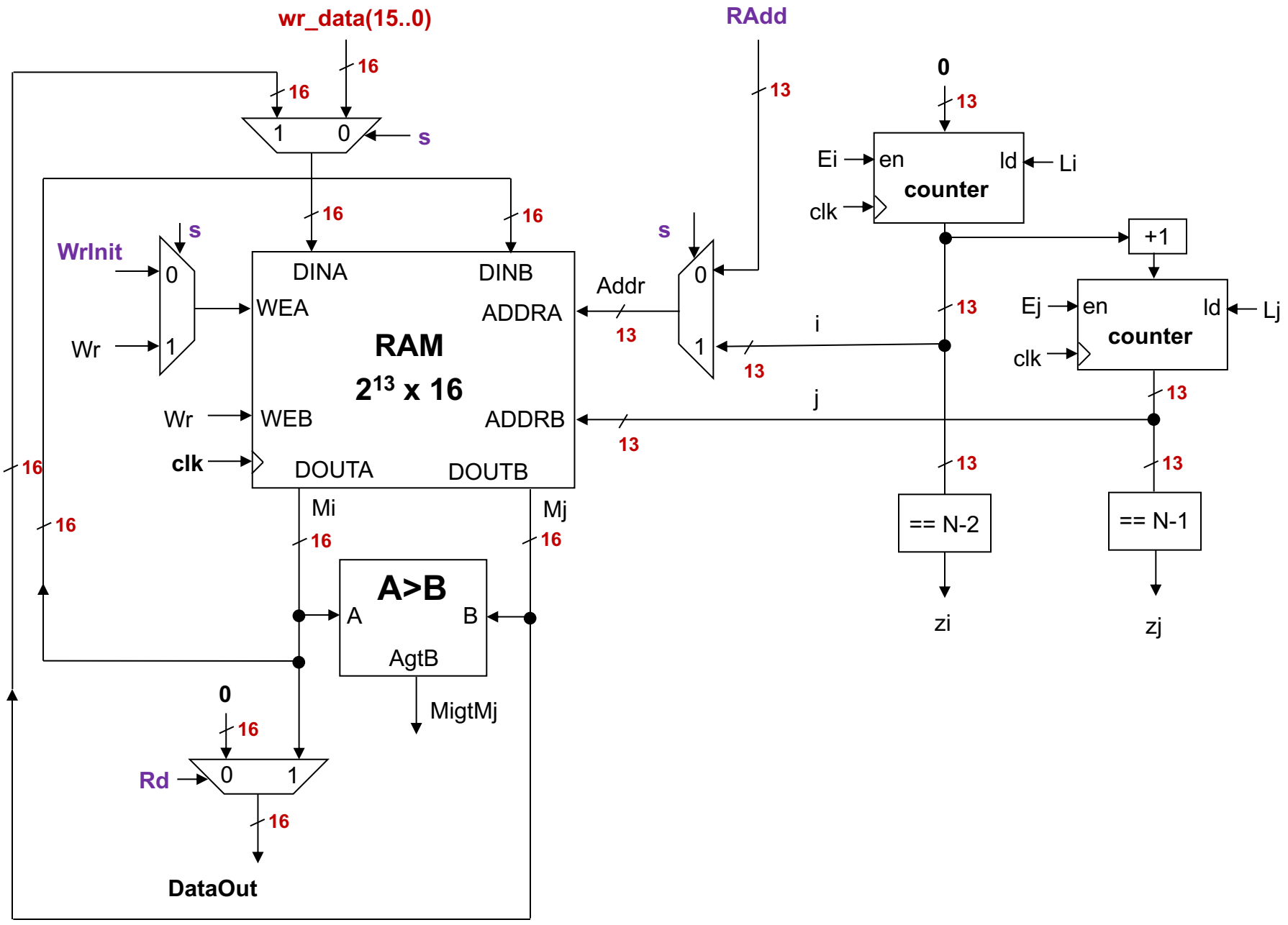
*** design adjustments * coding * functional verification**

Software Design:

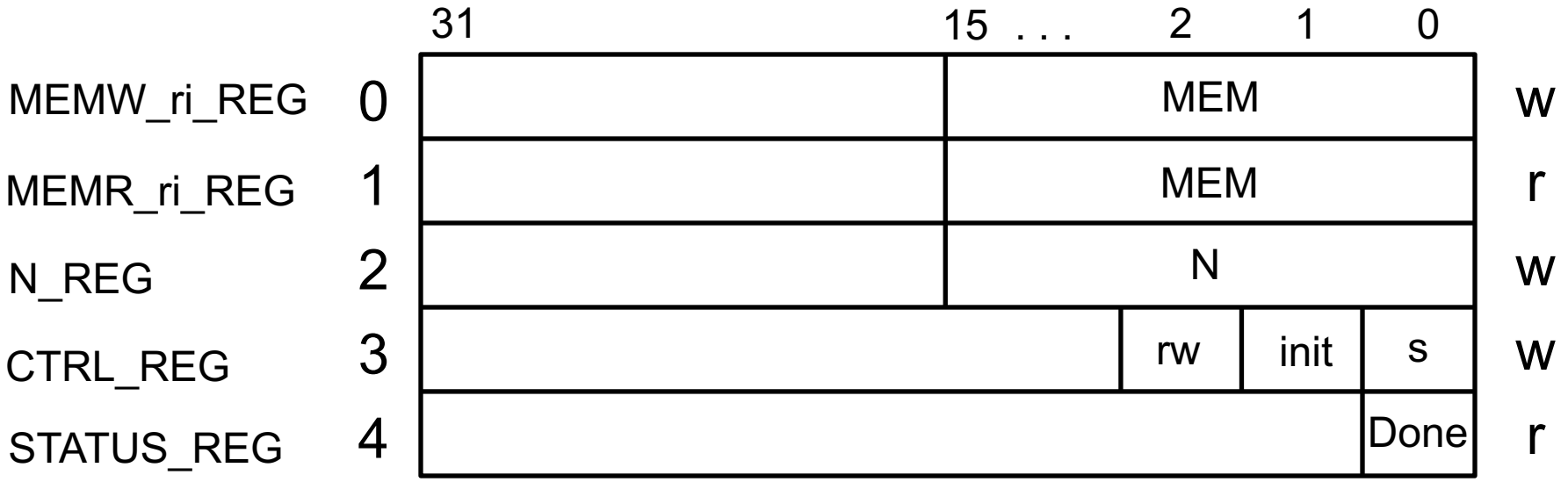
S1. Software driver

* declarations (.h) * implementations (.cpp) * testing

S2. Application based on functions of custom and standard cores


wr_data(15..0)
RAdd
16
13
0
1
s
Ei
en
ld
Li
counter
clk
WrInit
DINA
DINB
WEA
ADDRA
RAM
2^13 x 16
Wr
WEB
ADDRB
clk
DOUTA
DOUTB
Addr
i
j
+1
Ej
Lj
Mi
Mj
A>B
A
B
AgtB
MigtMj
Rd
DataOut
== N-2
== N-1
zi
zj

IO Register Map of the Sorting Core



Writing to MEMW_ri_REG : Writing to MEM[ri] & ri++

Reading from MEMR_ri_REG : Reading from MEM[ri] & ri++

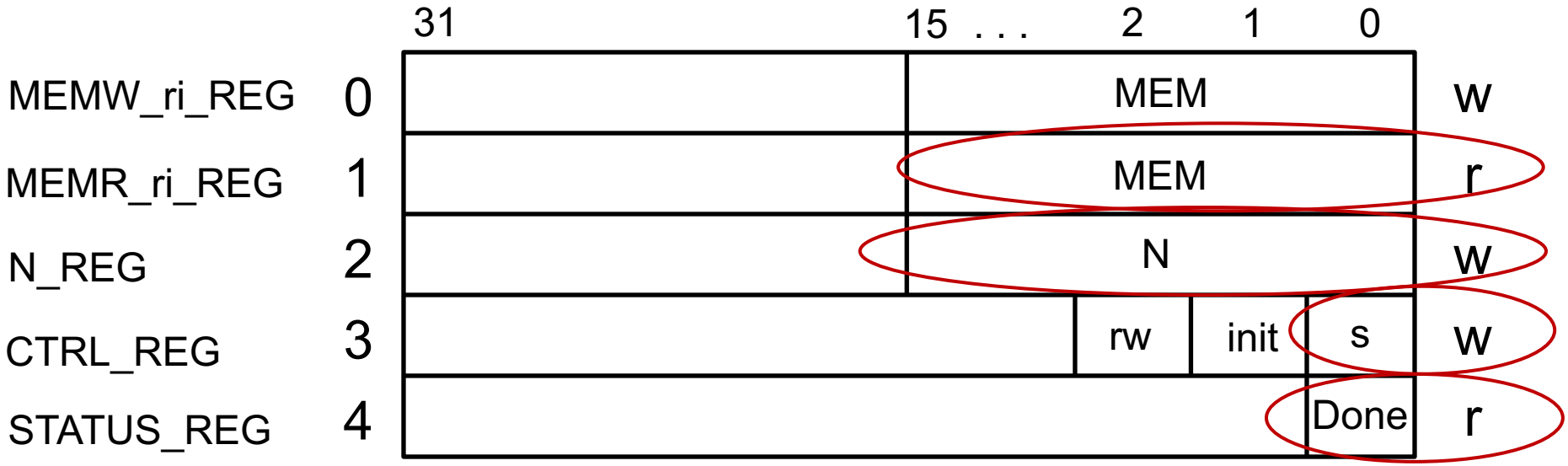
Writing to NREG_REG : Initializing N

Reading from STATUS_REG : Reading Done

Writing to CRTL_REG:

rw	init	s	
0	0	1	Set s=1 : computations
0	0	0	Set s=0 : initialization/readout
1	1	0	Set ri=0 and WrInit_r=1 : start initialization
0	1	0	Set ri=0 and Rd_r=1 : start readout

IO Register Map of the Sorting Core



Writing to MEMW_ri_REG : Writing to MEM[ri] & ri++

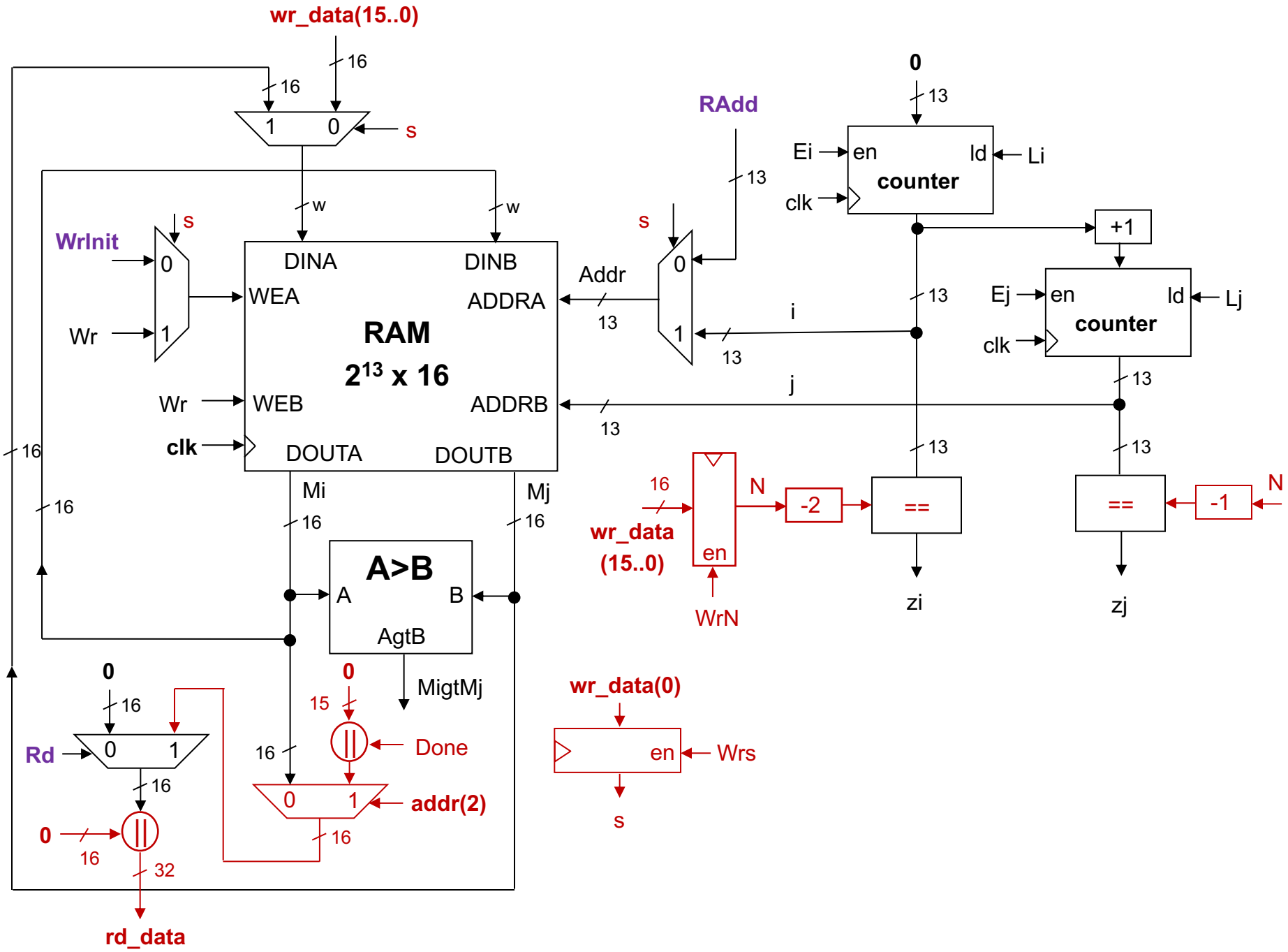
Reading from MEMR_ri_REG : Reading from MEM[ri] & ri++

Writing to NREG_REG : Initializing N

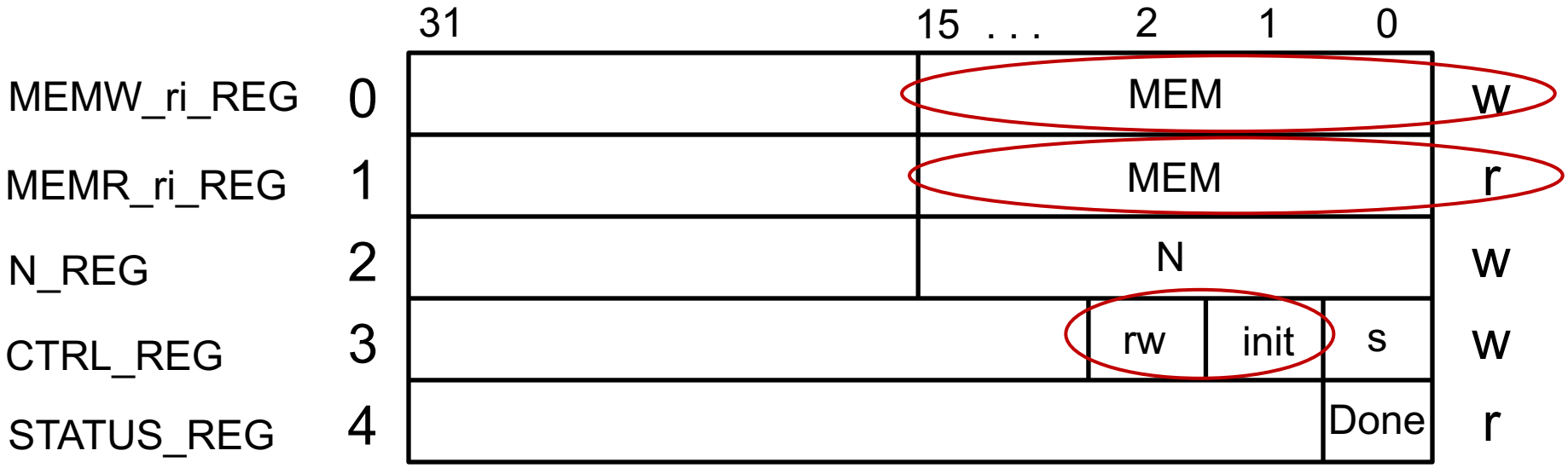
Reading from STATUS_REG : Reading Done

Writing to CRTL_REG:

rw	init	s	
0	0	1	Set s=1 : computations
0	0	0	Set s=0 : initialization/readout
1	1	0	Set ri=0 and WrInit_r=1 : start initialization
0	1	0	Set ri=0 and Rd_r=1 : start readout


wr_data(15..0)
16
16
1
0
s
w
w
WrInit
s
0
1
Wr
DINA
DINB
WEA
ADDRA
RAM
2^13 x 16
Wr
WEB
ADDRB
clk
DOUTA
DOUTB
16
16
Mi
Mj
16
16
A>B
A
B
AgtB
MigtMj
0
16
Rd
0
1
16
0
16
32
rd_data
0
15
Done
16
0
1
addr(2)
16
RAdd
13
s
0
1
Addr
13
13
i
13
j
13
0
13
Ei
en
ld
Li
counter
clk
+1
Ej
en
ld
Lj
counter
clk
13
13
16
wr_data
(15..0)
en
WrN
N
-2
==
zi
13
13
==
-1
N
zj
wr_data(0)
en
Wrs
s

IO Register Map of the Sorting Core



Writing to MEMW_ri_REG : Writing to MEM[ri] & ri++

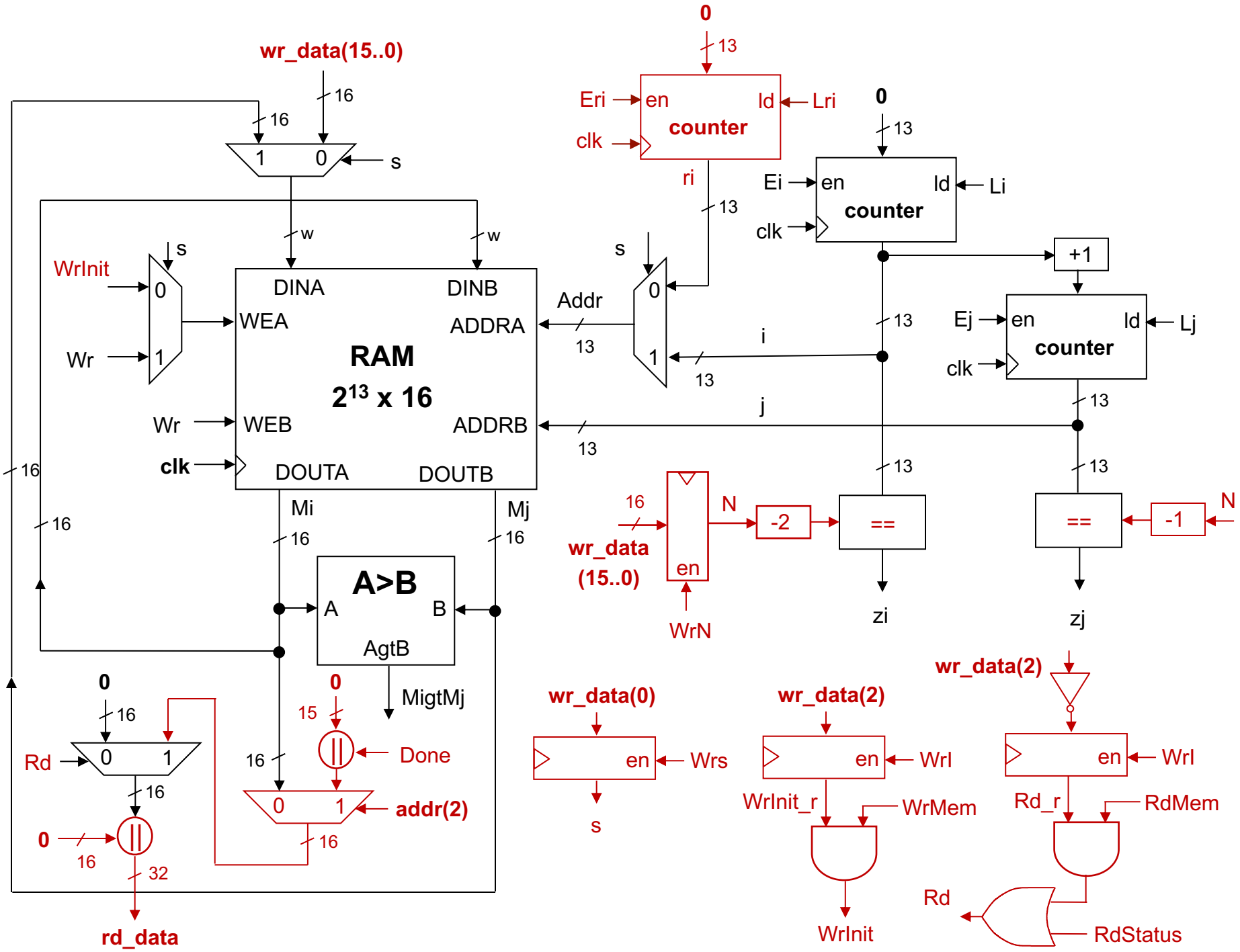
Reading from MEMR_ri_REG : Reading from MEM[ri] & ri++

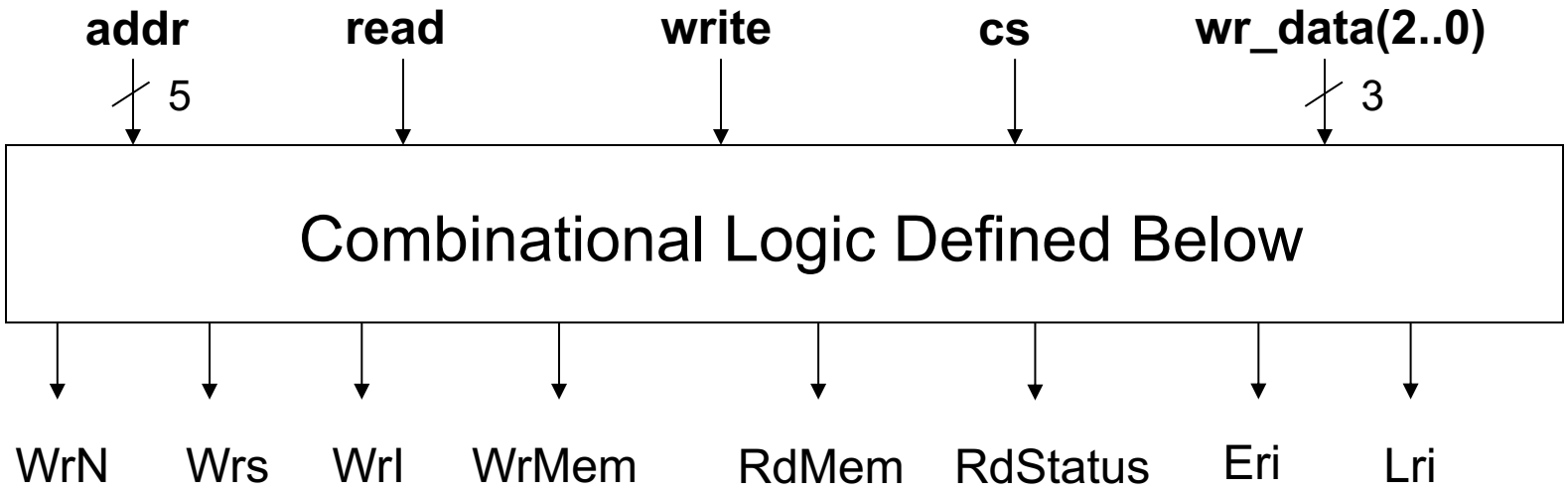
Writing to NREG_REG : Initializing N

Reading from STATUS_REG : Reading Done

Writing to CRTL_REG:

rw	init	s	
0	0	1	Set s=1 : computations
0	0	0	Set s=0 : initialization/readout
1	1	0	Set ri=0 and WrInit_r=1 : start initialization
0	1	0	Set ri=0 and Rd_r=1 : start readout


wr_data(15..0)
16
16
1
0
s
w
w
WrInit
s
0
1
Wr
DINA
DINB
WEA
ADDRA
RAM
2^13 x 16
Wr
WEB
ADDRB
clk
DOUTA
DOUTB
16
16
Mi
Mj
16
16
A>B
A
B
AgtB
MigtMj
0
16
Rd
0
1
16
0
16
32
0
15
Done
16
0
1
addr(2)
16
rd_data
0
13
Eri
en
ld
Lri
clk
counter
ri
13
0
13
Ei
en
ld
Li
clk
counter
+1
s
0
1
Addr
13
i
13
13
Ej
en
ld
Lj
clk
counter
13
j
13
13
13
16
wr_data
(15..0)
en
N
-2
==
N
==
-1
WrN
zi
zj
wr_data(0)
en
Wrs
s
wr_data(2)
en
WrI
WrInit_r
WrMem
WrInit
wr_data(2)
en
WrI
Rd_r
RdMem
Rd
RdStatus

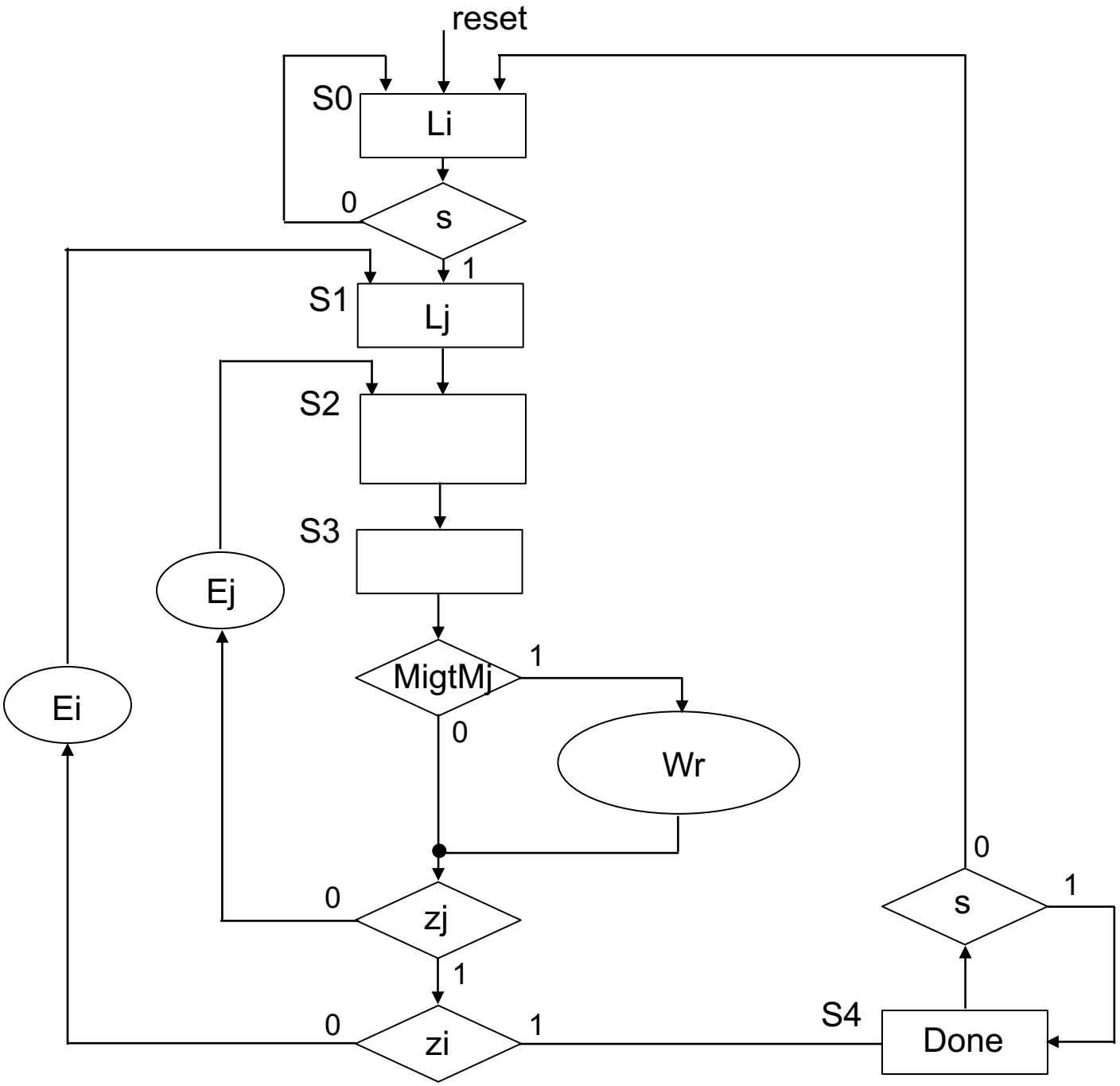


Output	cs	write	read	addr	wr_data(2..0)
WrN=1	1	1	0	2	–––
Wrs=1	1	1	0	3	–0–
Wrl=1	1	1	0	3	–10
WrMem=1	1	1	0	0	–––
RdMem=1	1	0	1	1	–––
RdStatus=1	1	0	1	4	–––

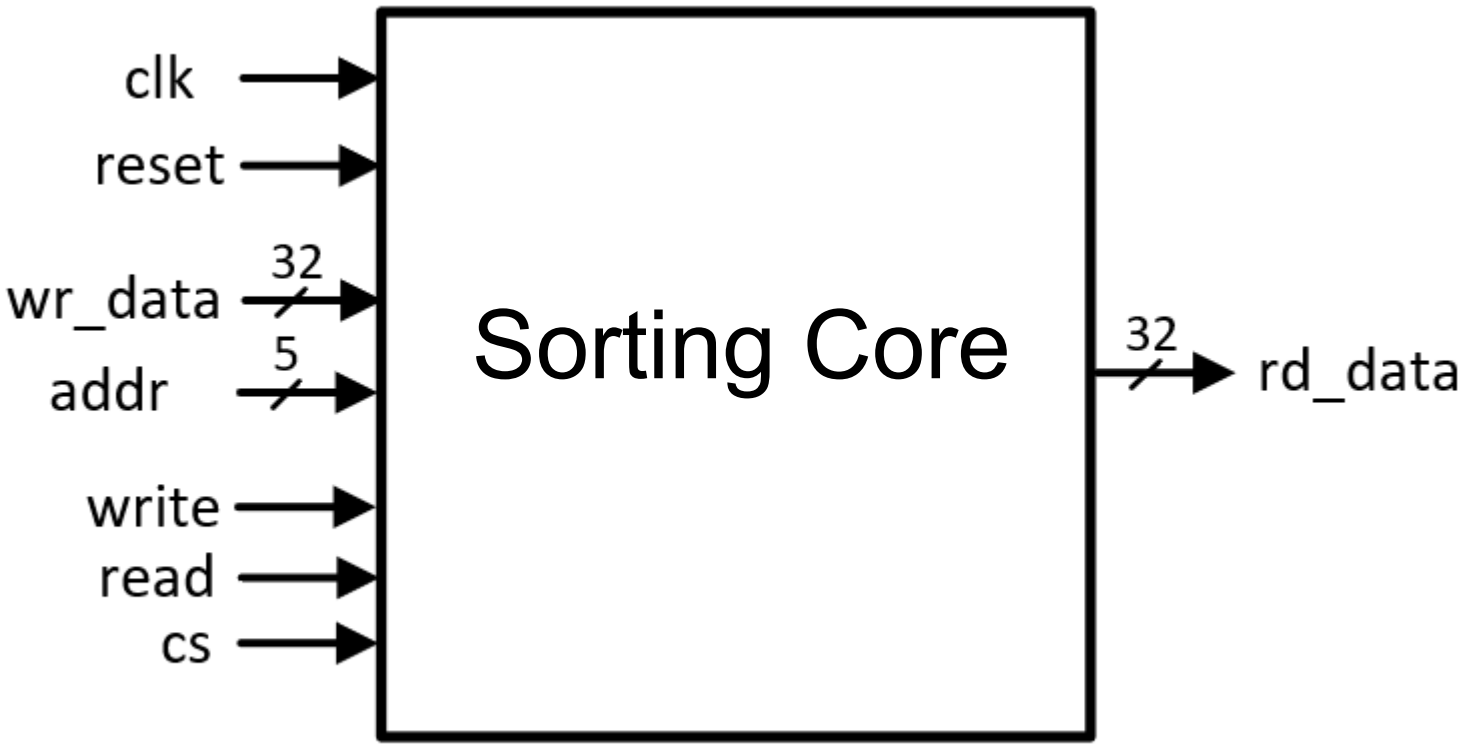
Eri = WrMem or RdMem

Lri = Wrl

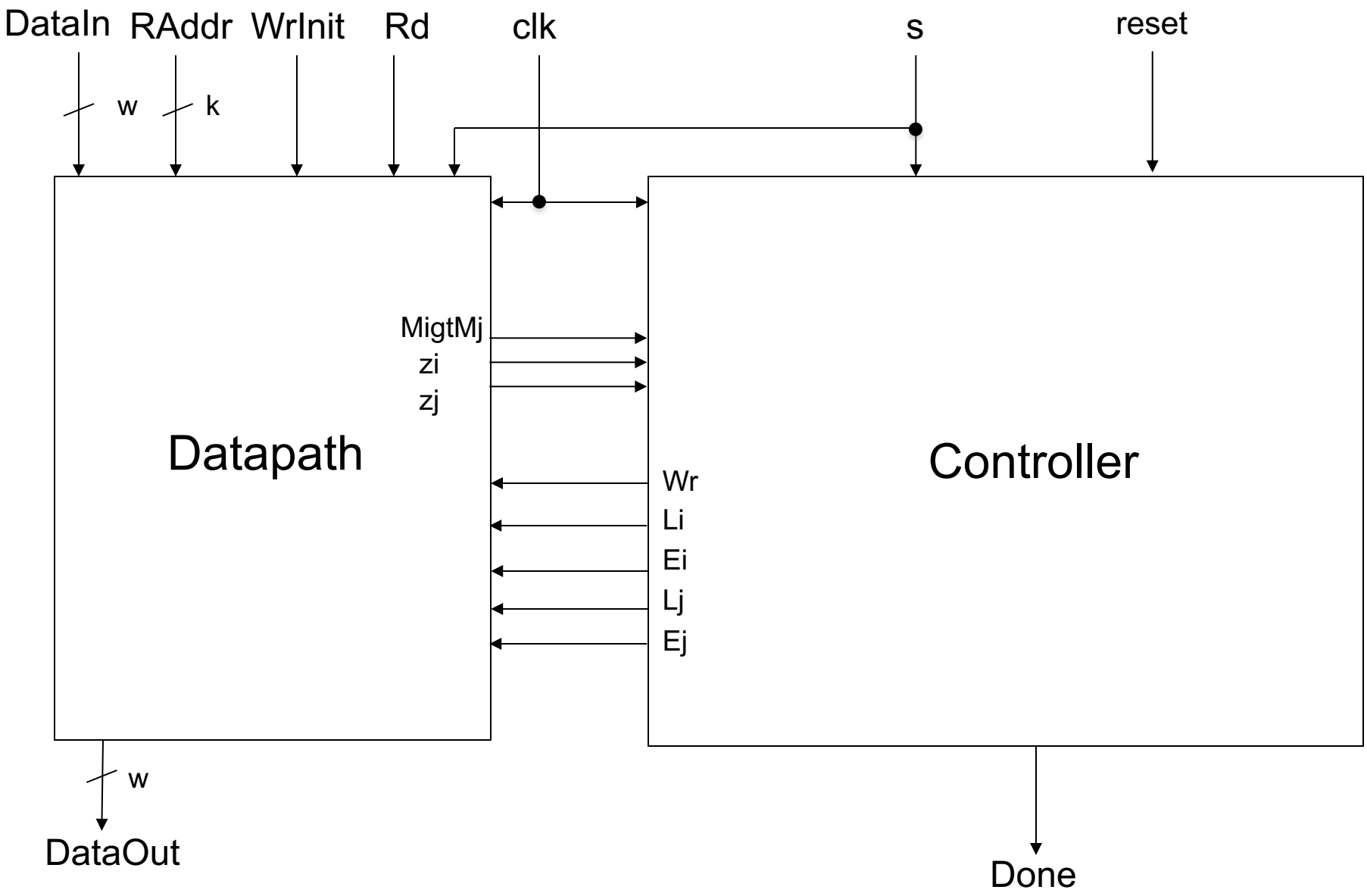
ASM Chart


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

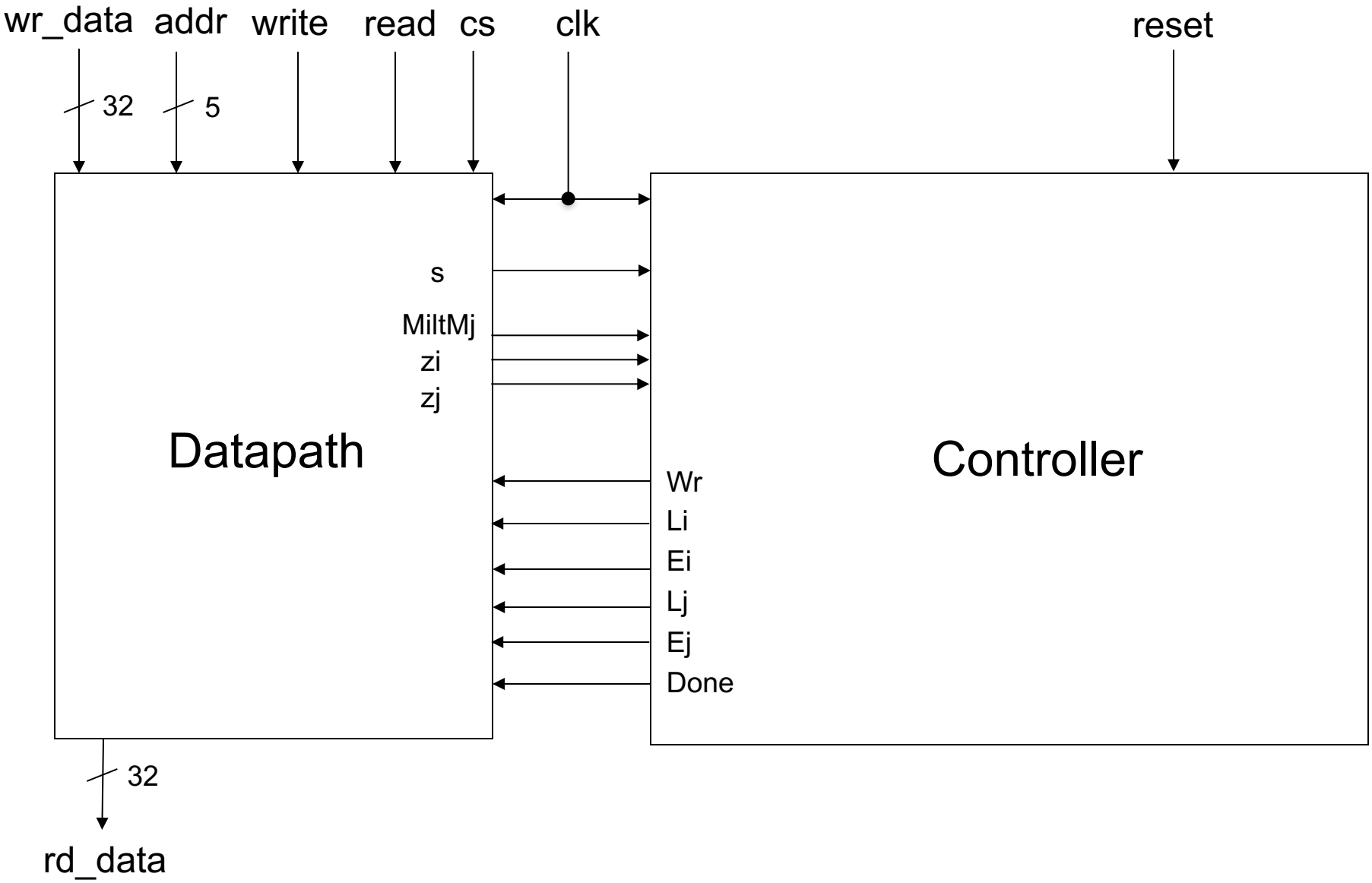
Interface

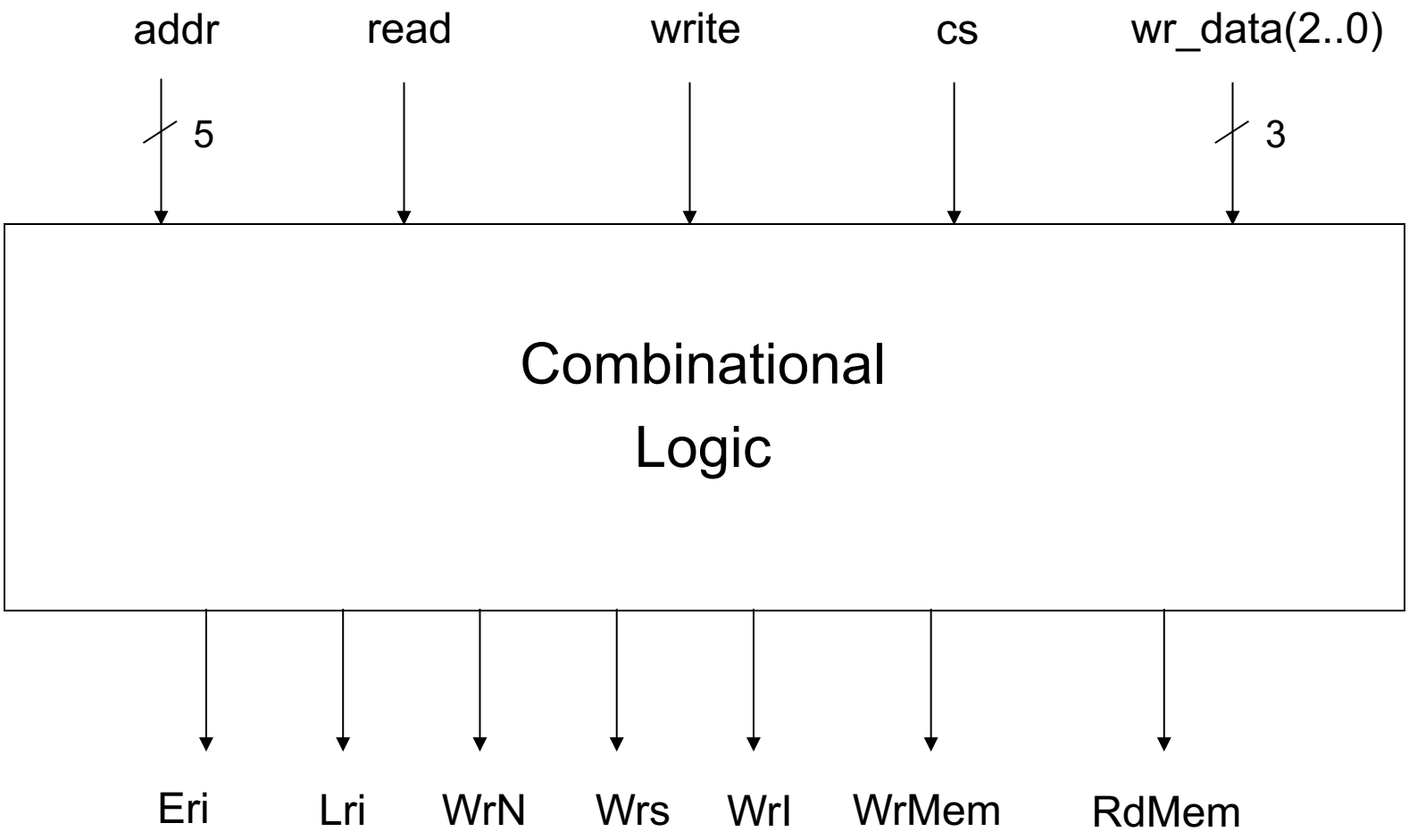

clk
reset
32
wr_data
5
addr
write
read
cs
Sorting Core
32
rd_data

Interface with the division into the Datapath and Controller w/o a Wrapper



Interface with the division into the Datapath and Controller with a Wrapper




addr
read
write
cs
wr_data(2..0)
5
3
Combinational
Logic
Eri
Lri
WrN
Wrs
Wrl
WrMem
RdMem