ECE 448
Lecture 18

Software/Hardware Co-design Using the FPro System
Part 2: 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

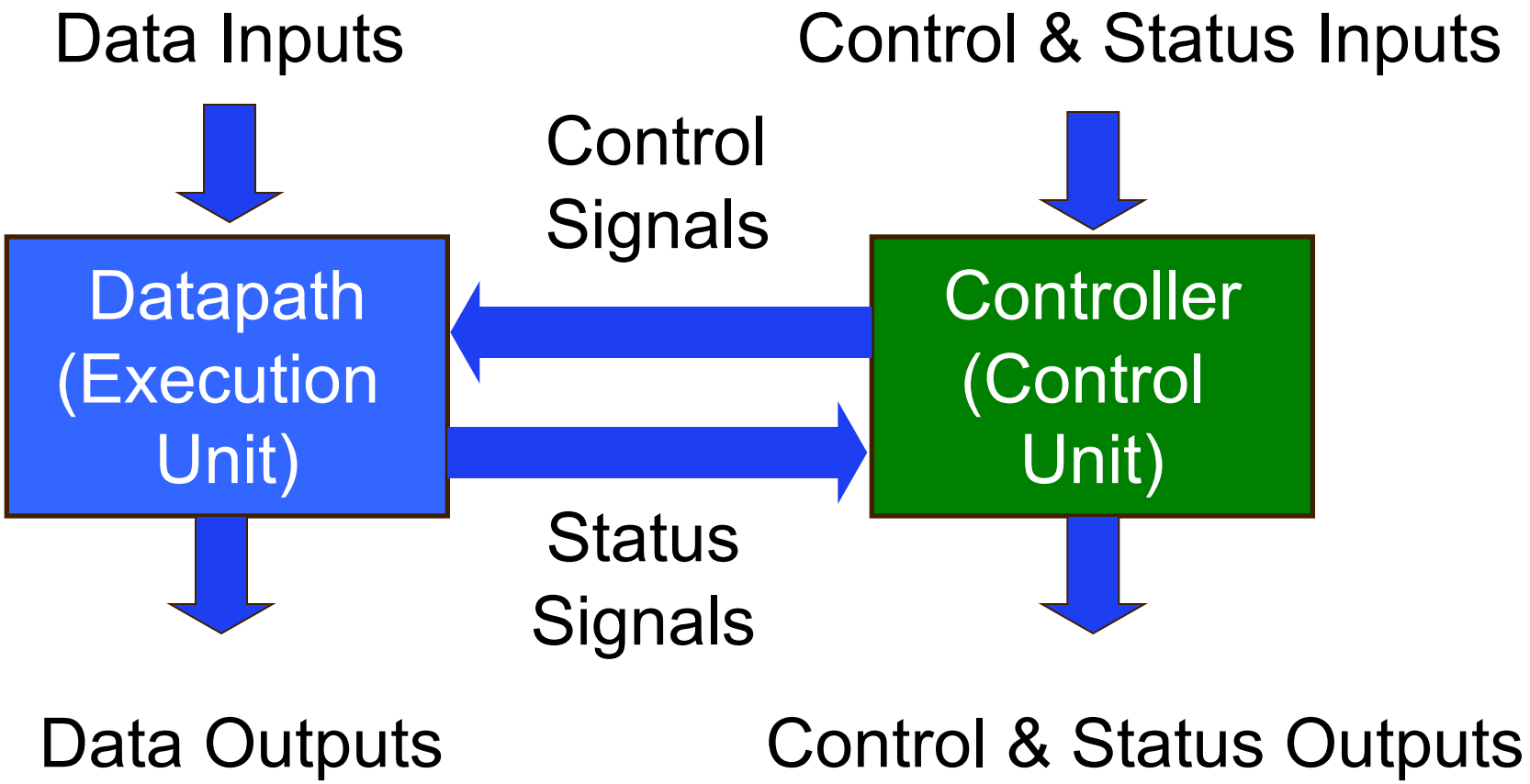
Software Design:

S1. Software driver

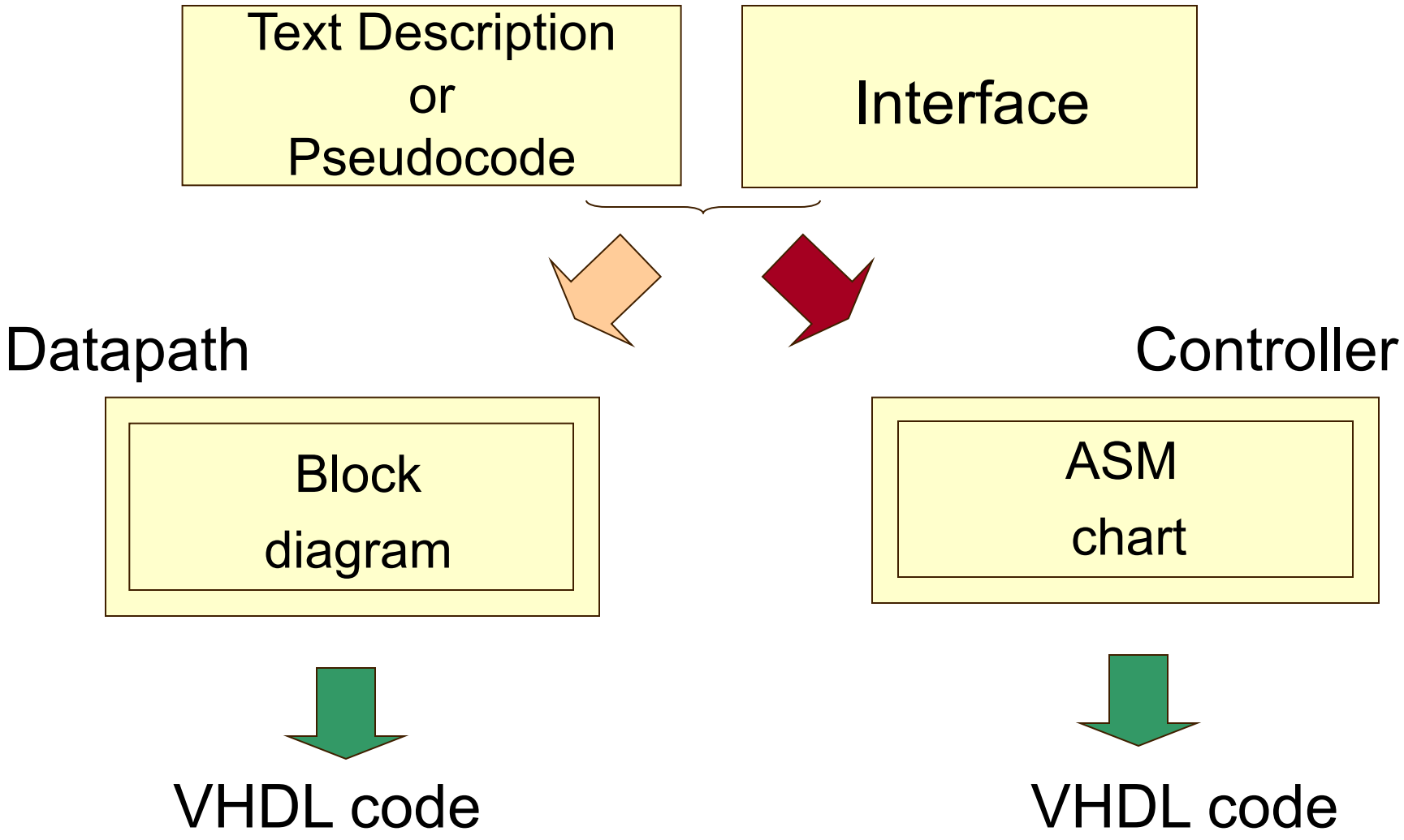
* declarations (.h) * implementations (.cpp) * testing

S2. Application based on functions of custom and standard cores

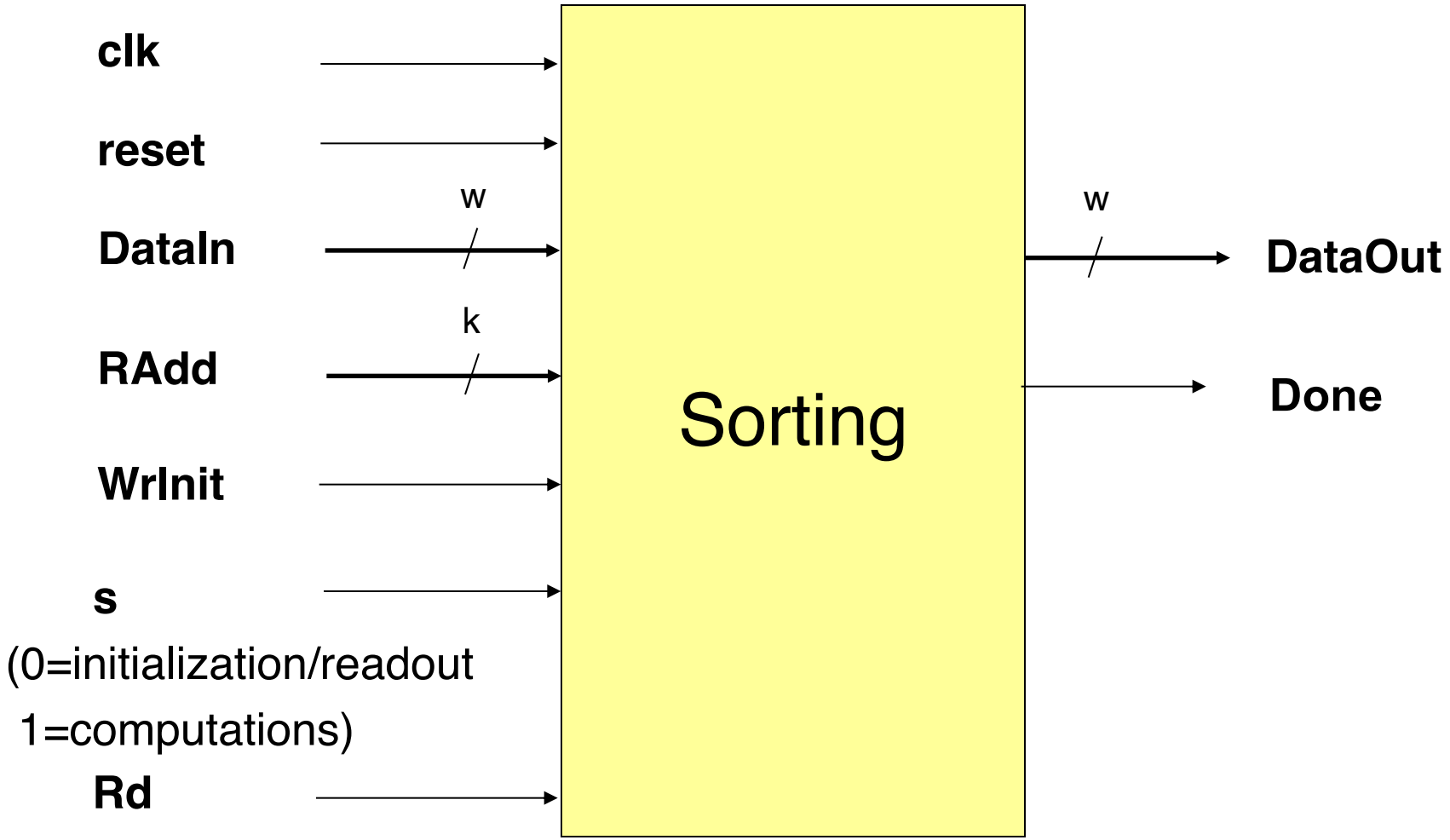
Structure of a Typical Digital System



Hardware Design with RTL VHDL

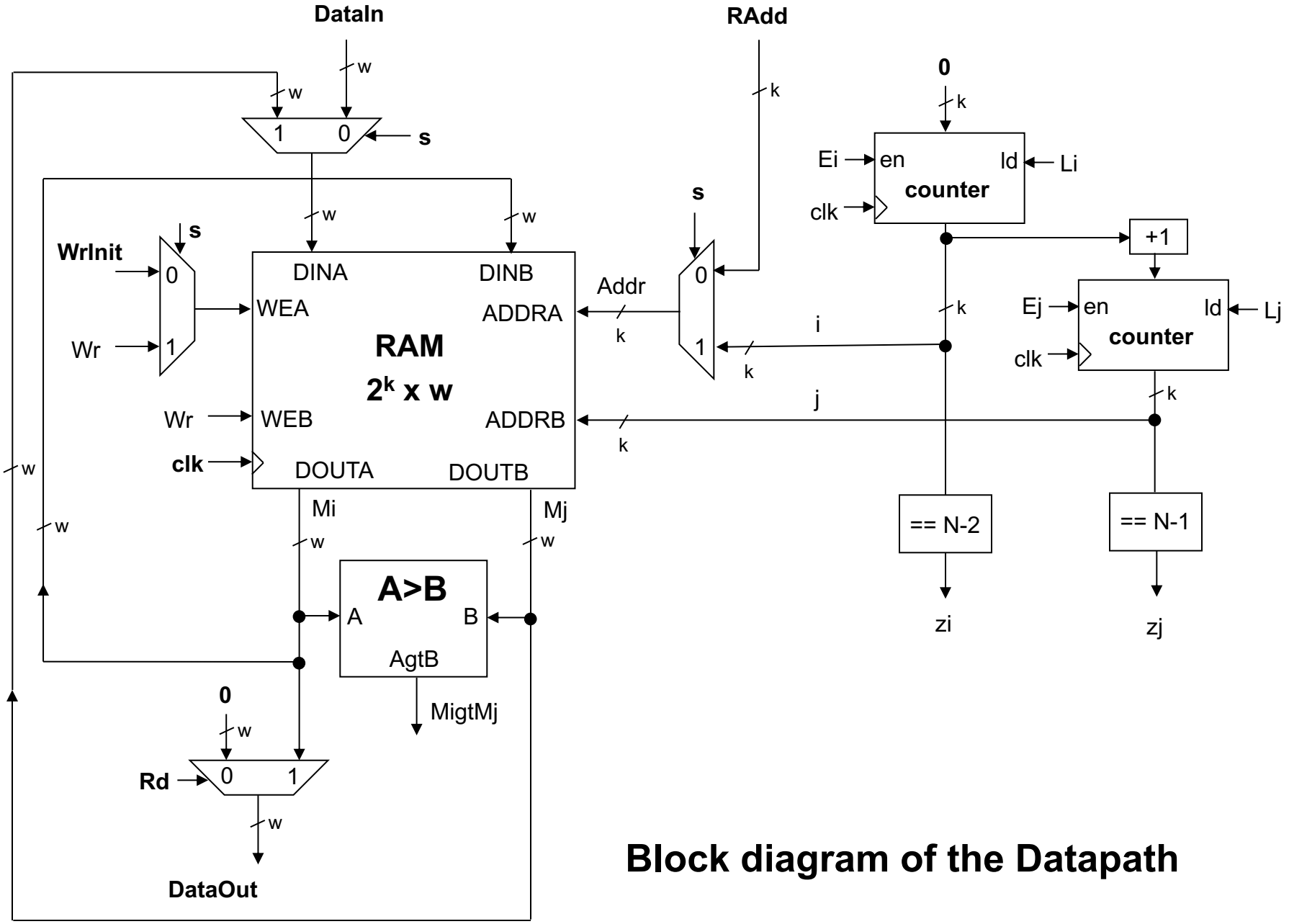


Sorting


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

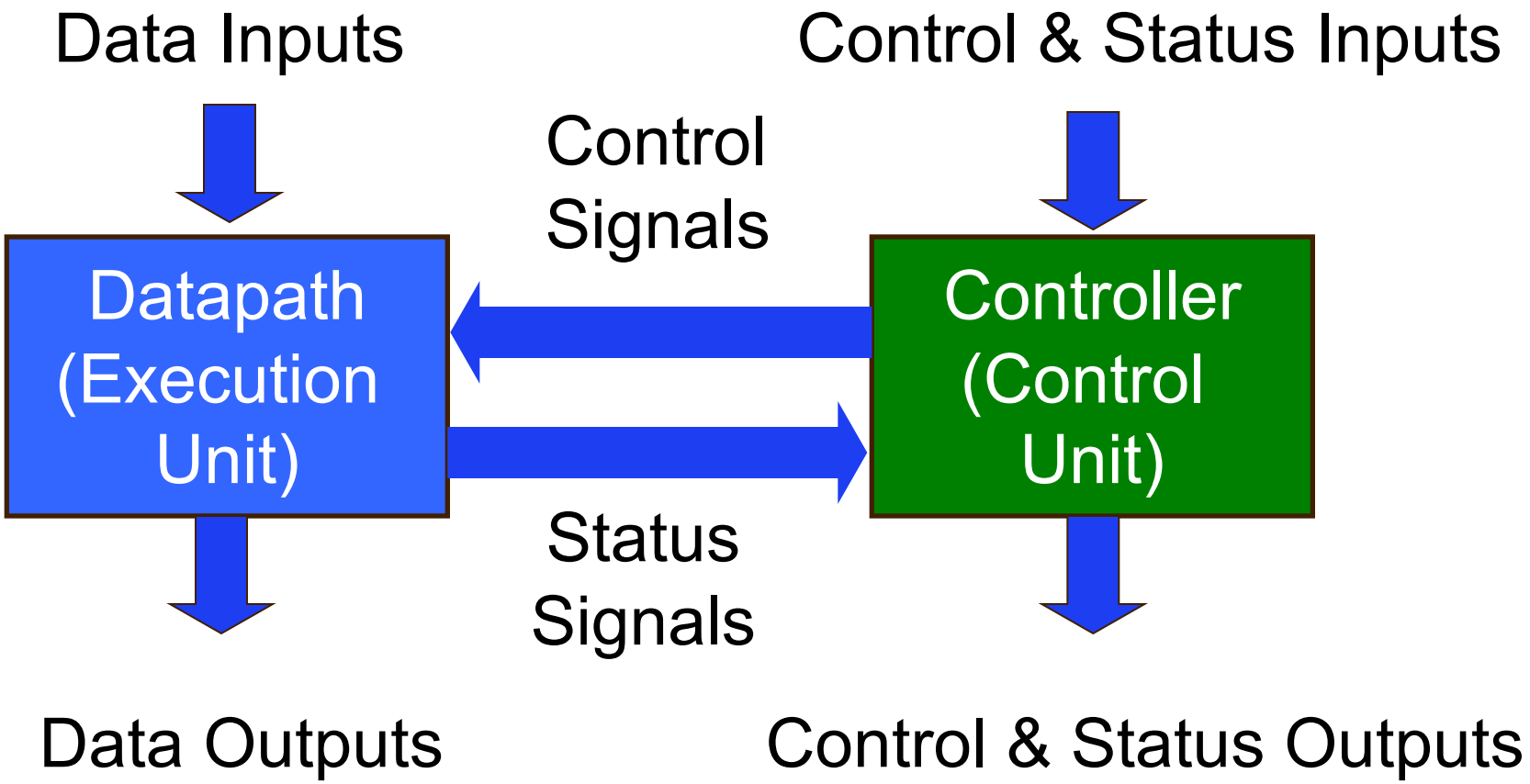
Pseudocode

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

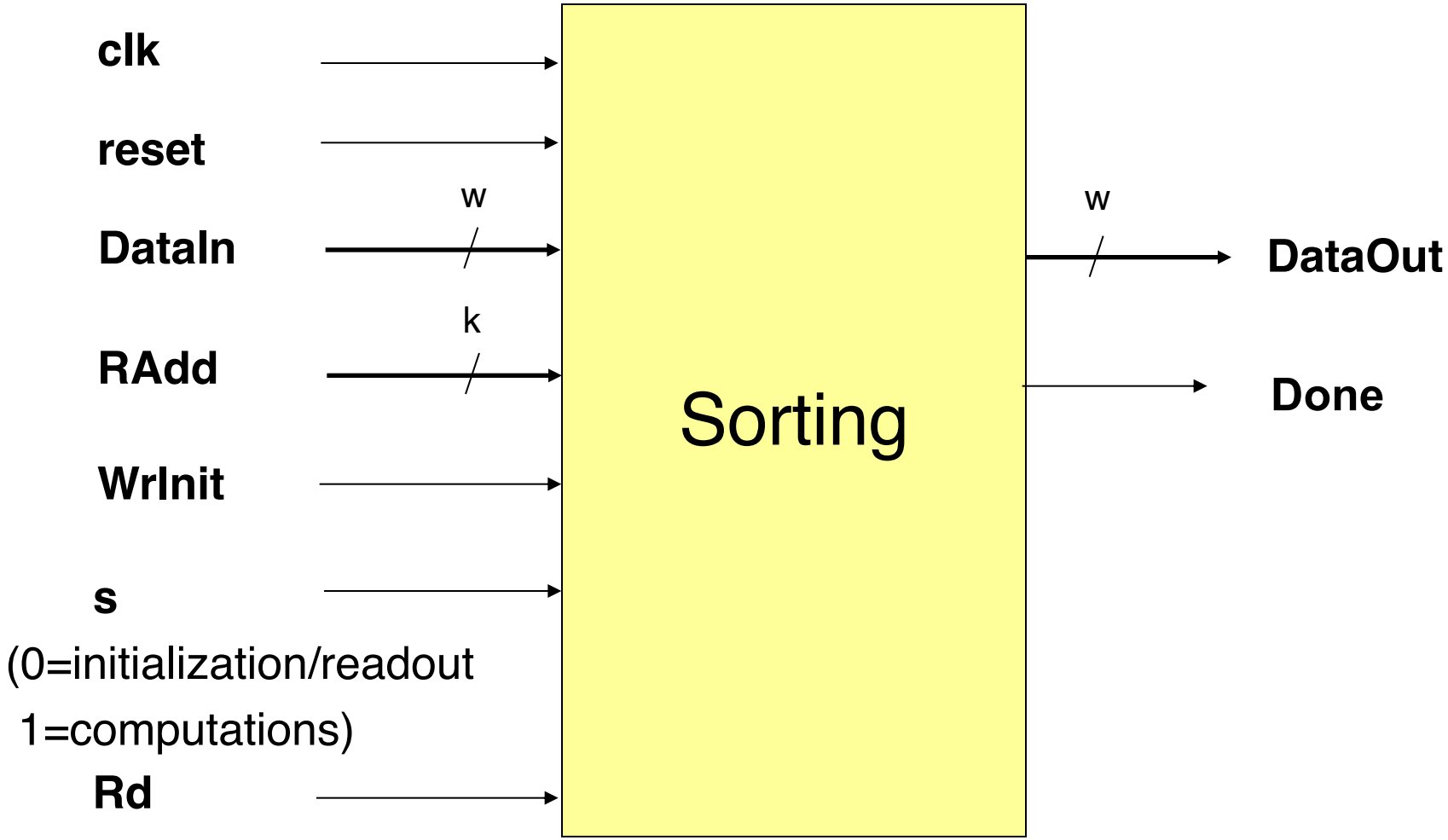


Block diagram of the Datapath

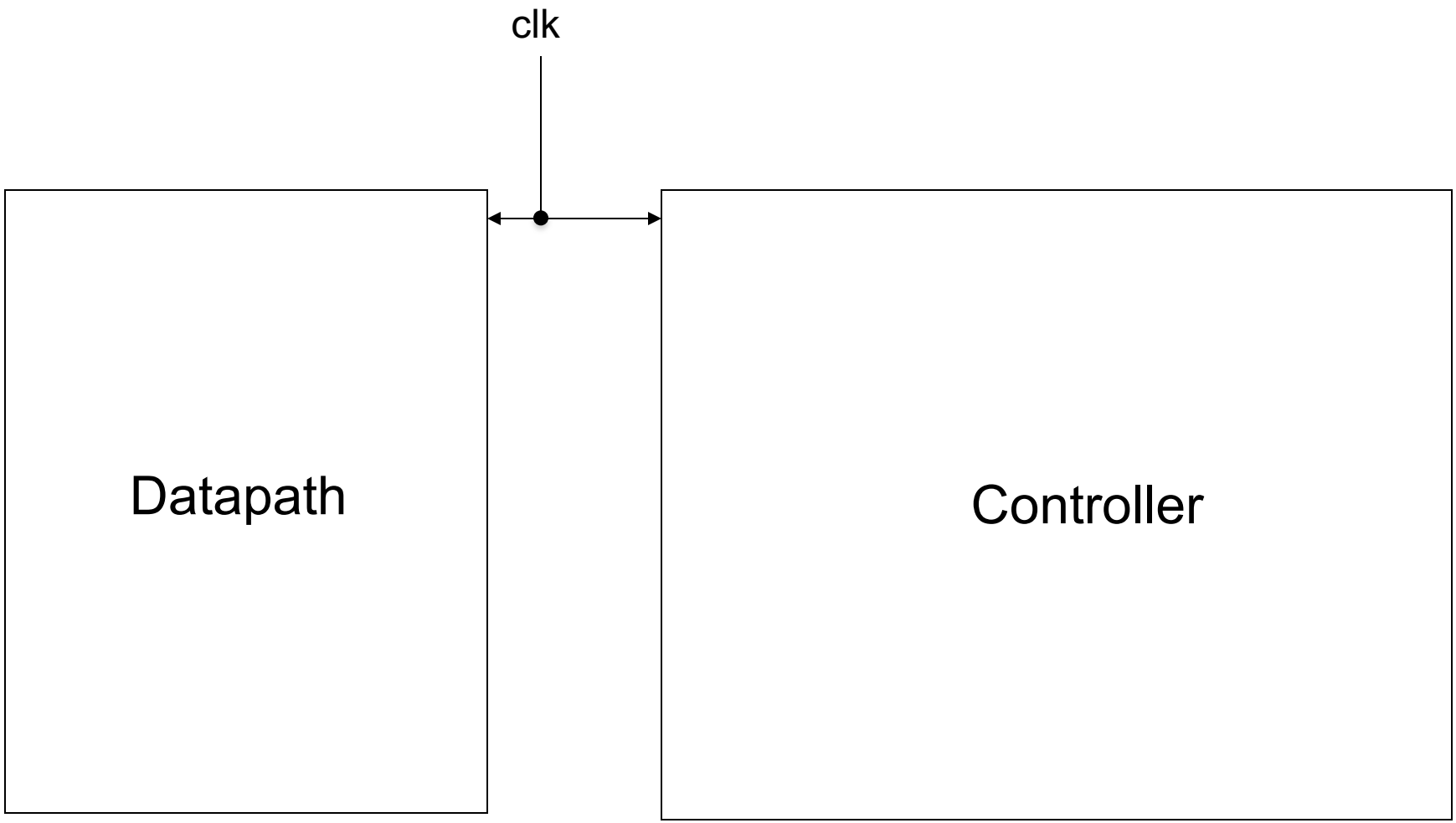
Structure of a Typical Digital System



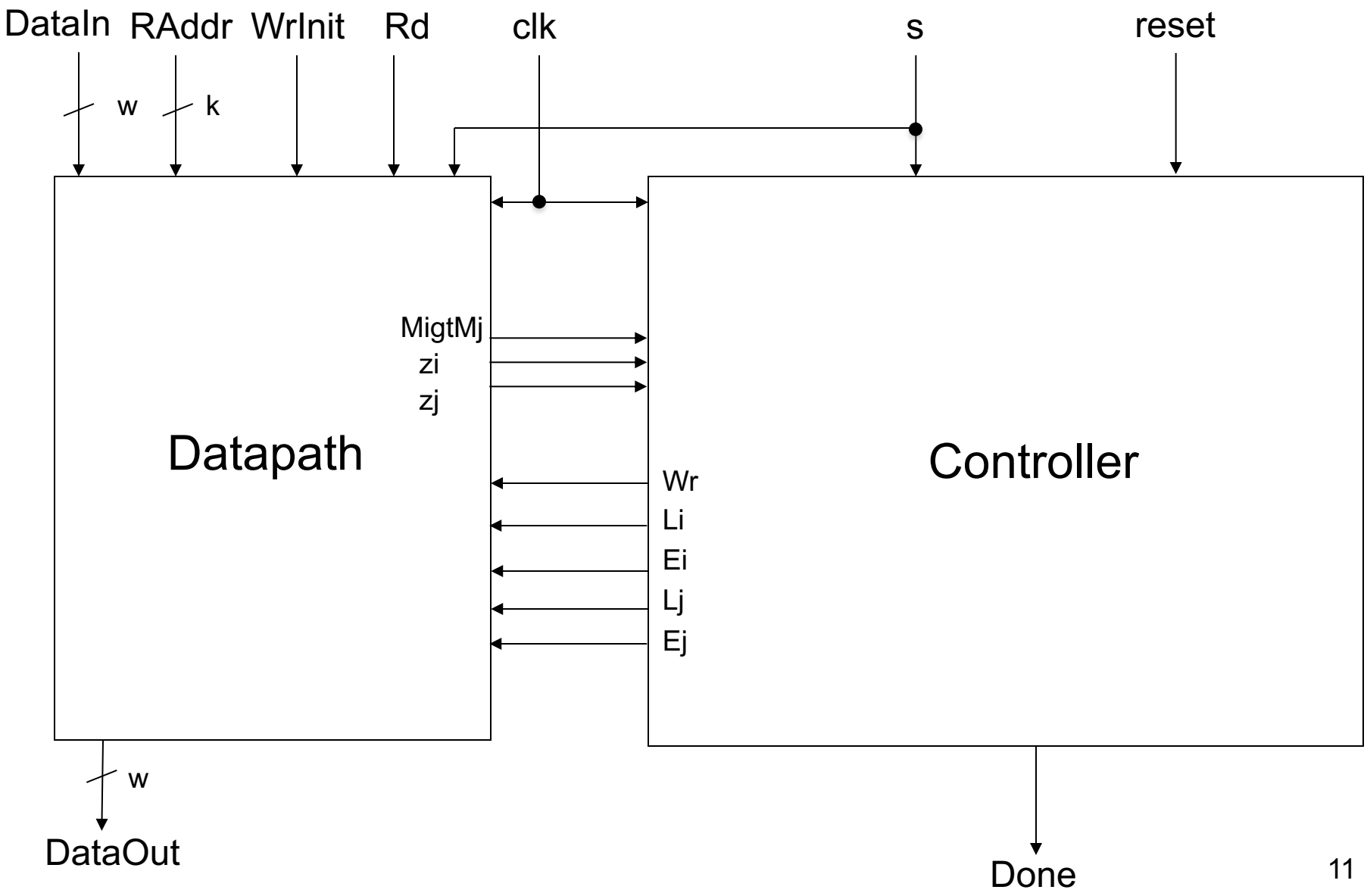
Sorting


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



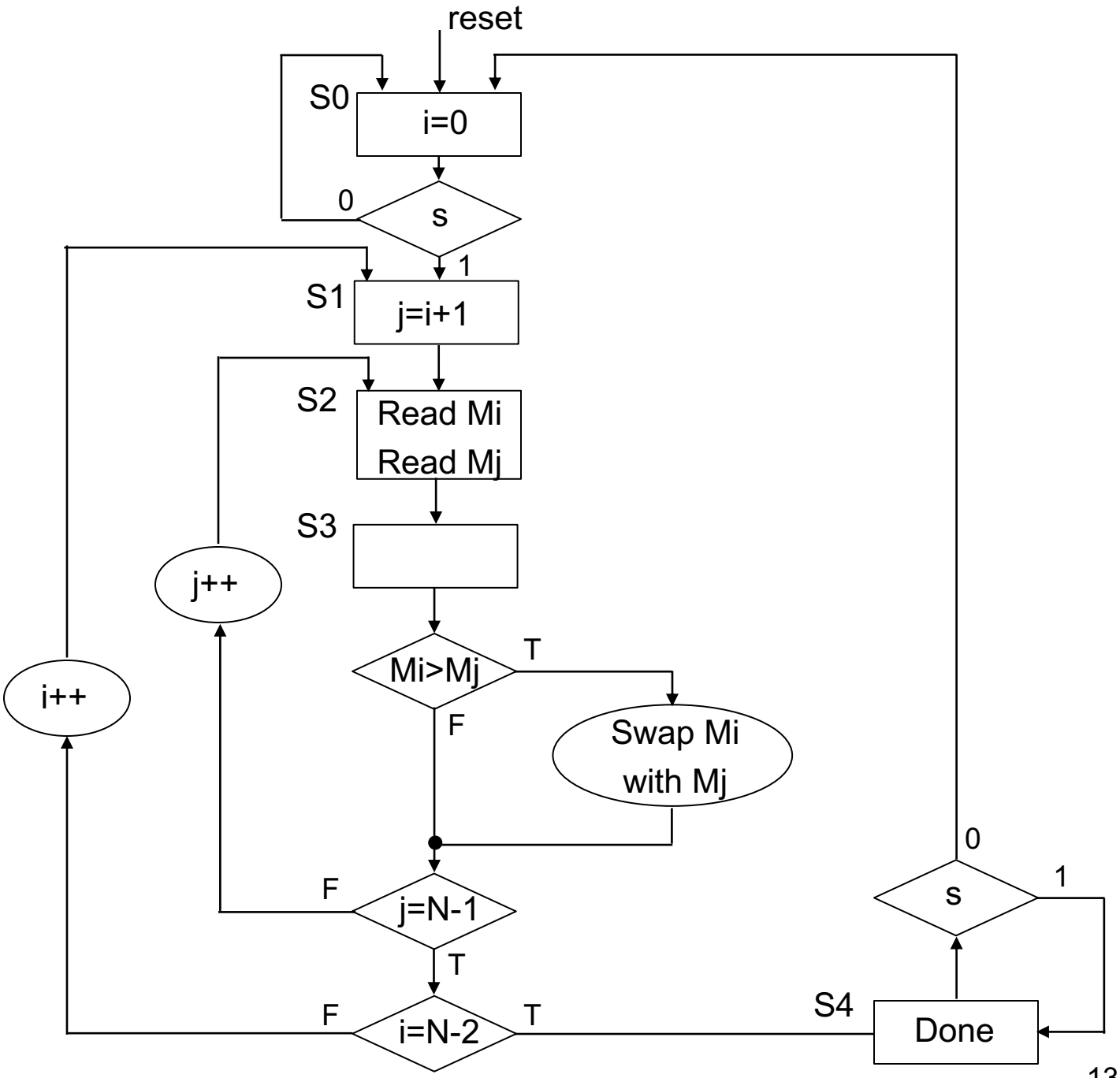
Interface with the division into the Datapath and Controller



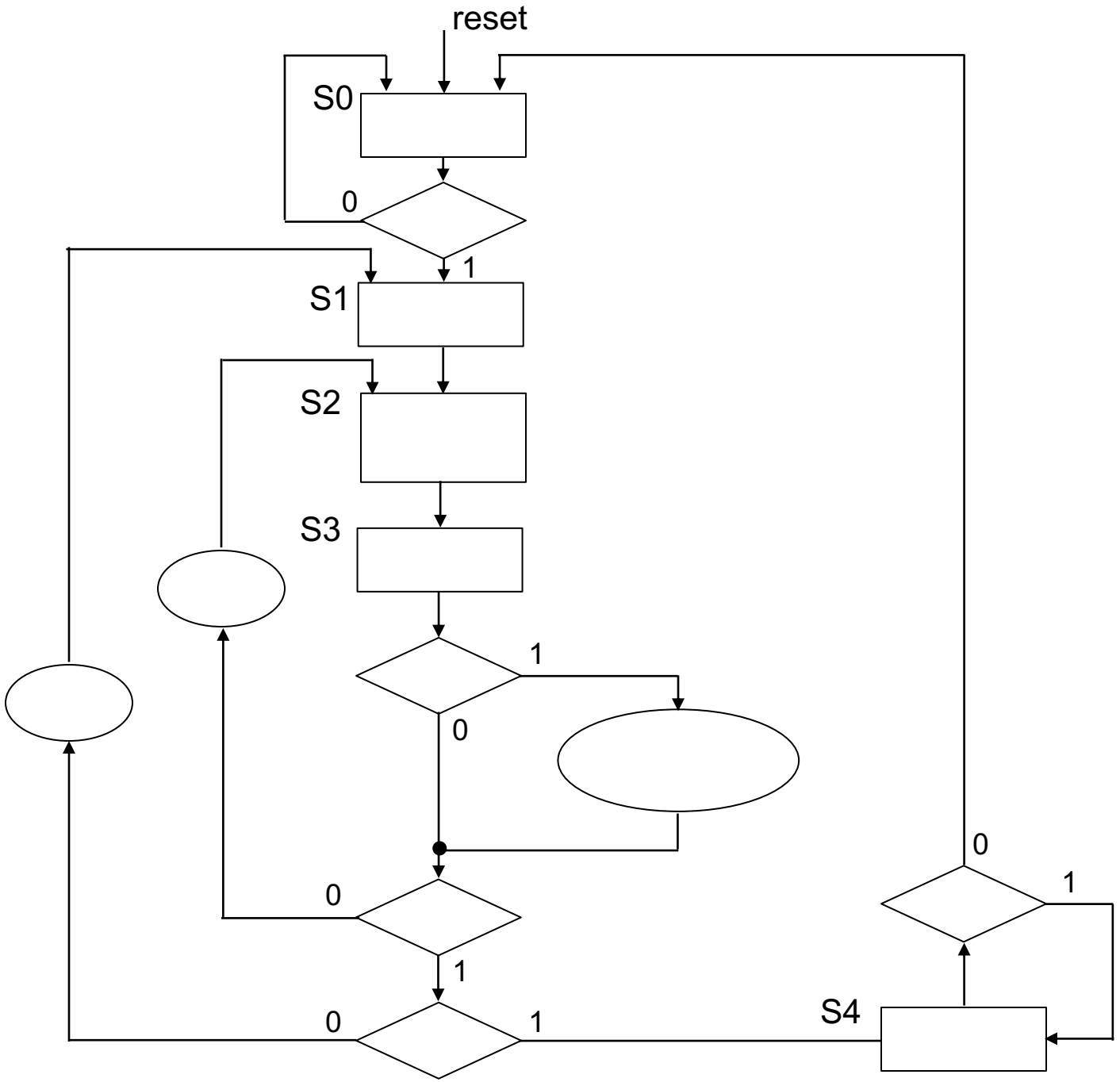
Pseudocode

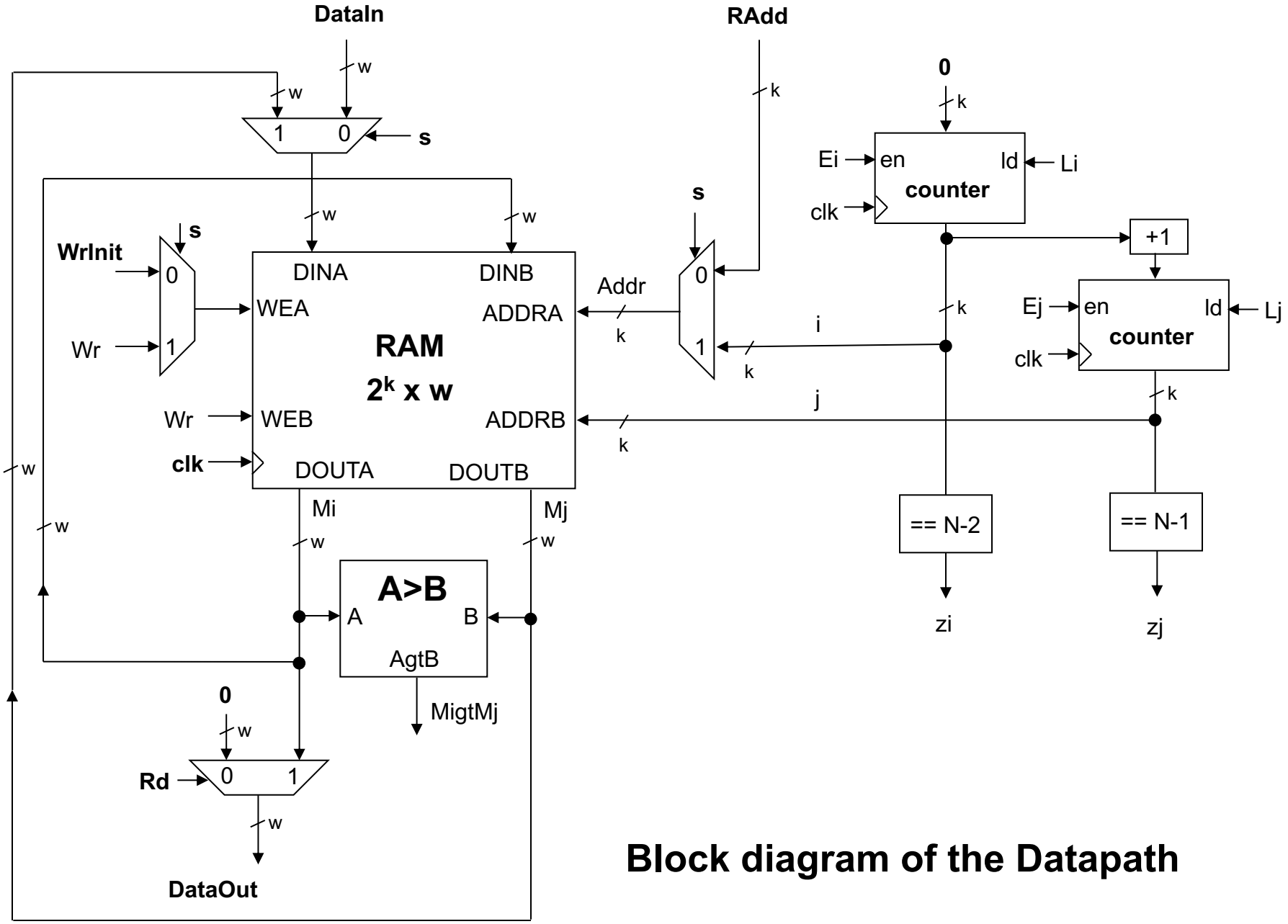
wait for s=1
for i=0 to N-2 do
    for j=i+1 to N-1 do
        if $M_i > M_j$ then
            Swap M_i with M_j
        end if
    end for
end for
Done
wait for s=0
go to the beginning

ASM Chart


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

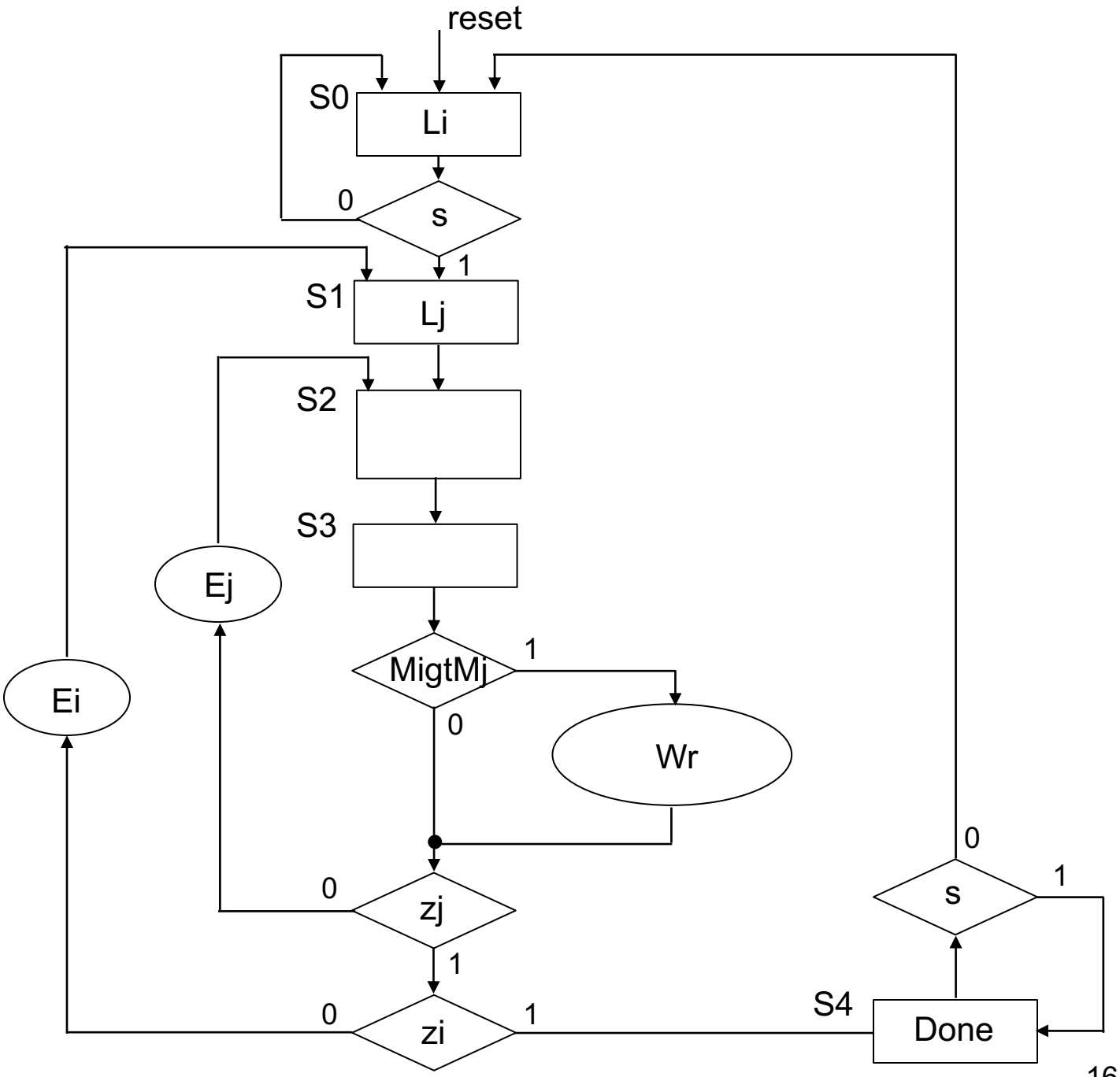
ASM Chart



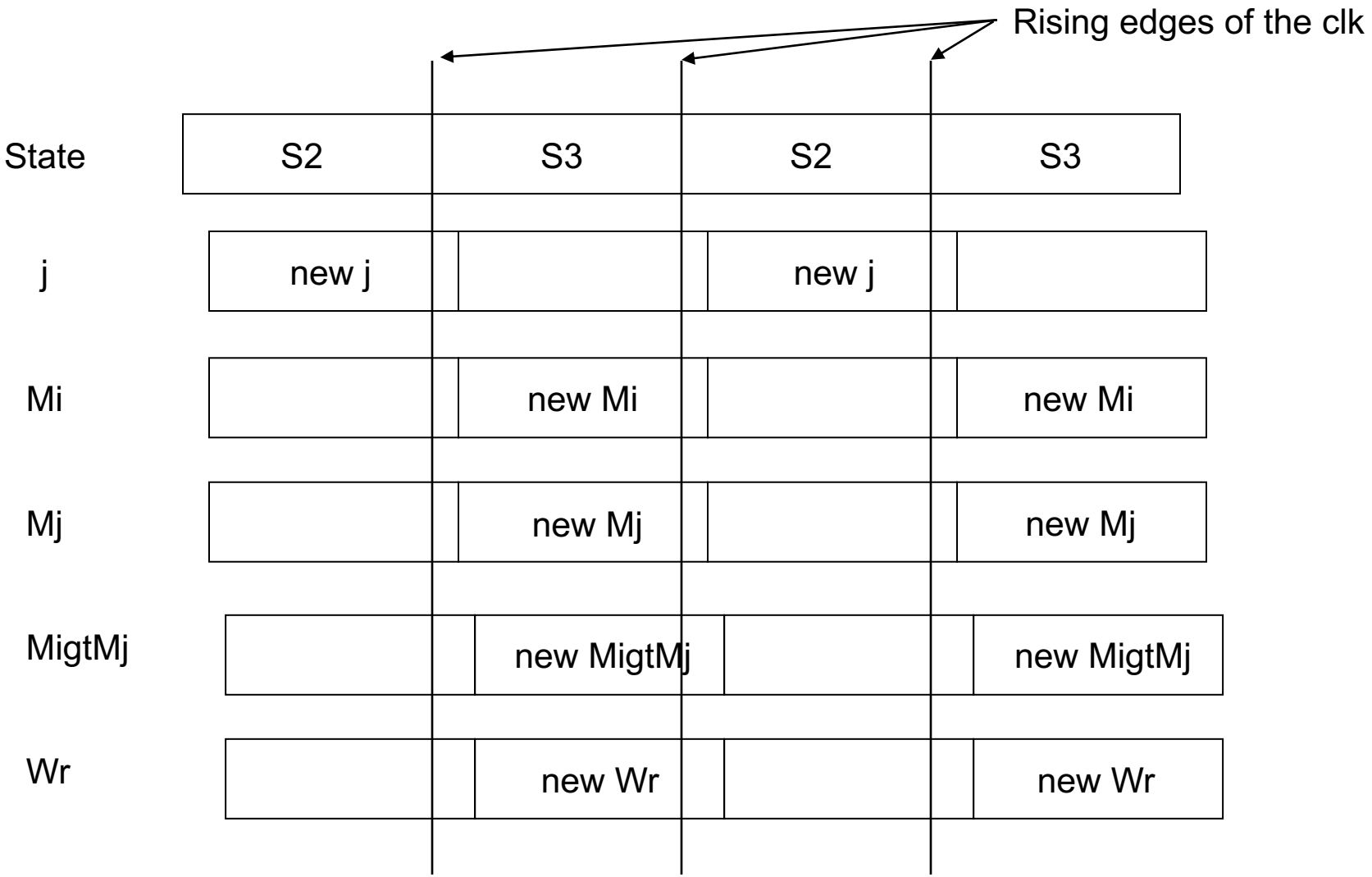


Block diagram of the Datapath

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

Timing Analysis



Lab 5
Exercise 3

Simplified Lab 5 Assignment

Task 1 – Initializing Memory

Button Left (BTNL) =

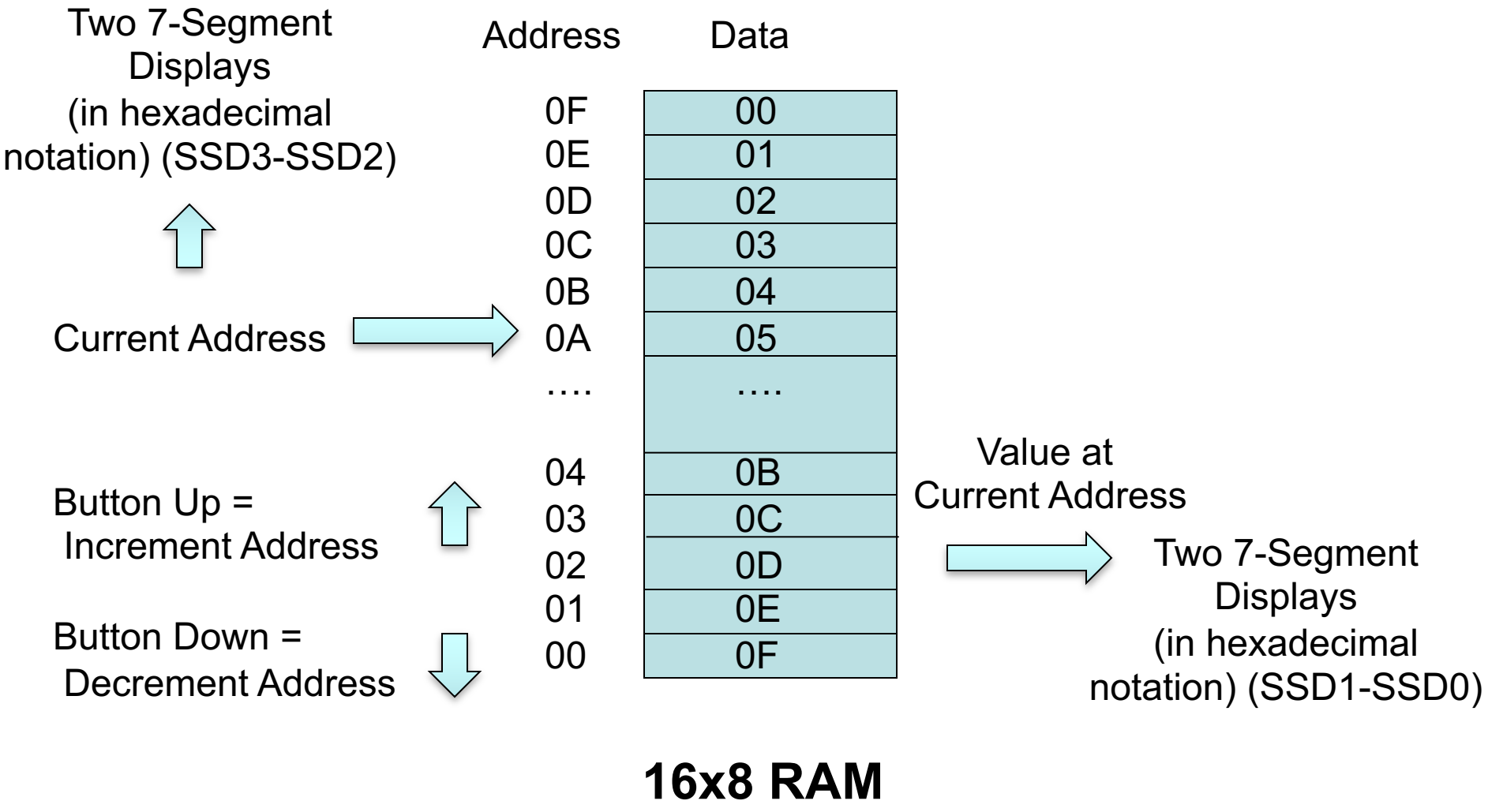
Initialize memory with
Data = 0x0F-Addr

Current Address

Address	Data
0F	00
0E	01
0D	02
0C	03
0B	04
0A	05
….	….
04	0B
03	0C
02	0D
01	0E
00	0F

16x8 RAM

Task 2 – Display Mode



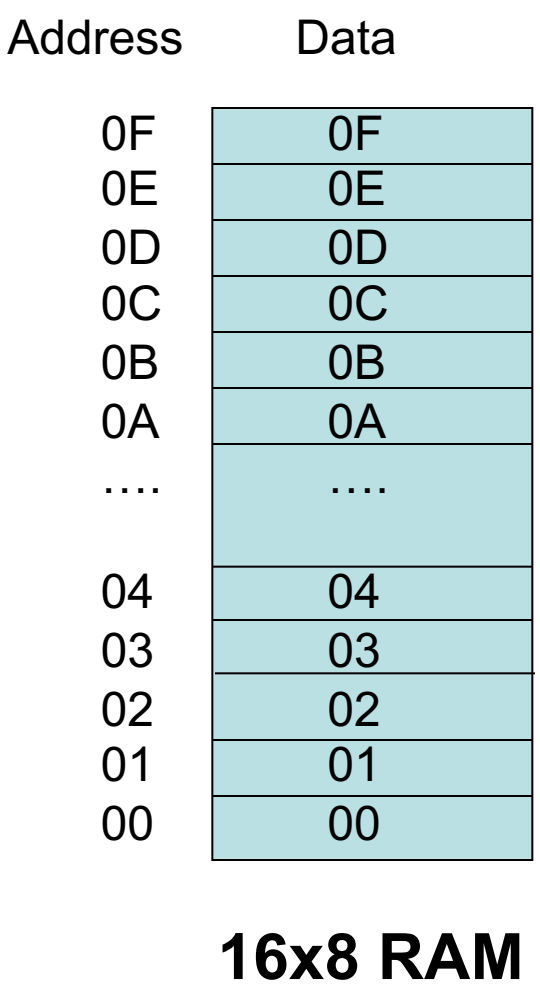
16x8 RAM

Task 3 – Sorting

Started by pressing BTNC.

Sorting unsigned numbers in the ascending order

During Sorting display: “----”
on the Seven Segment Displays.



16x8 RAM

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**

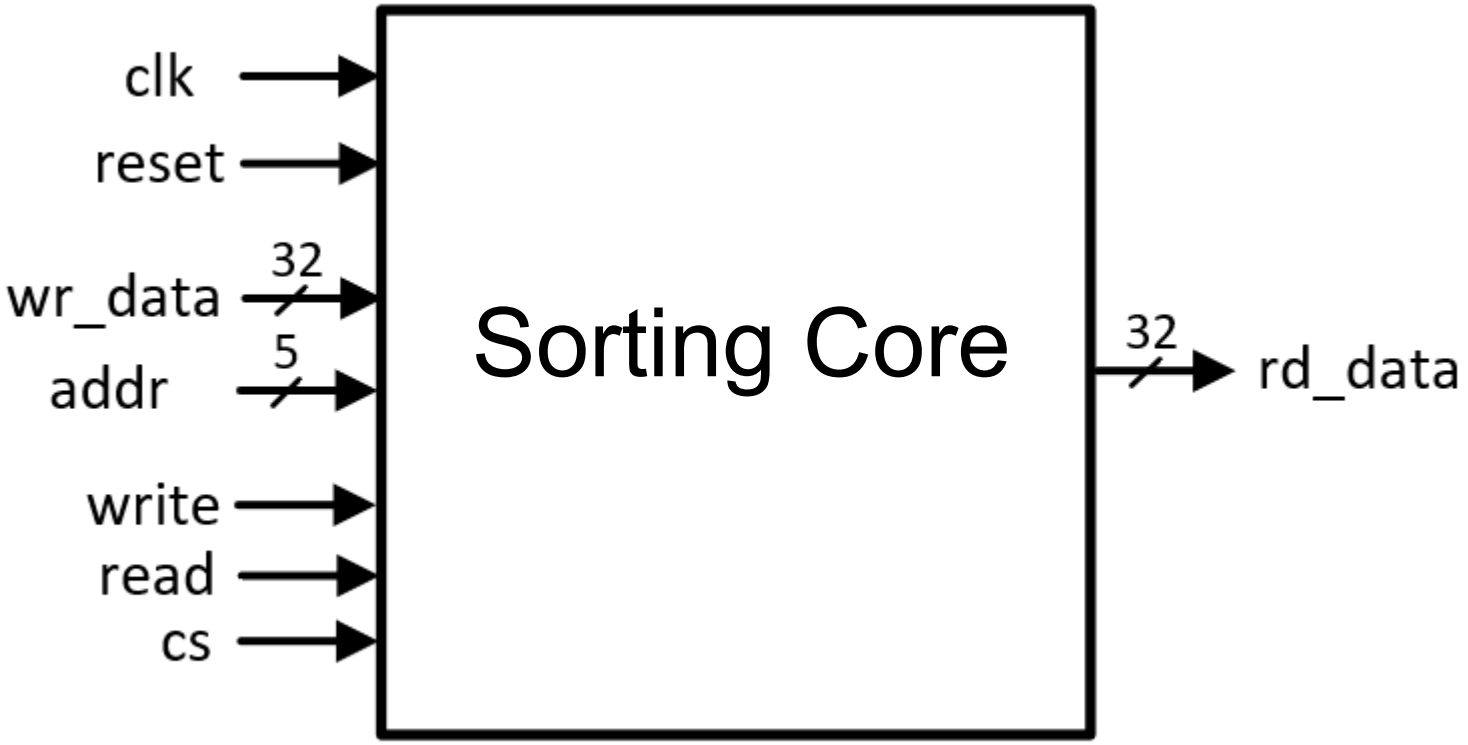
Software Design:

S1. Software driver

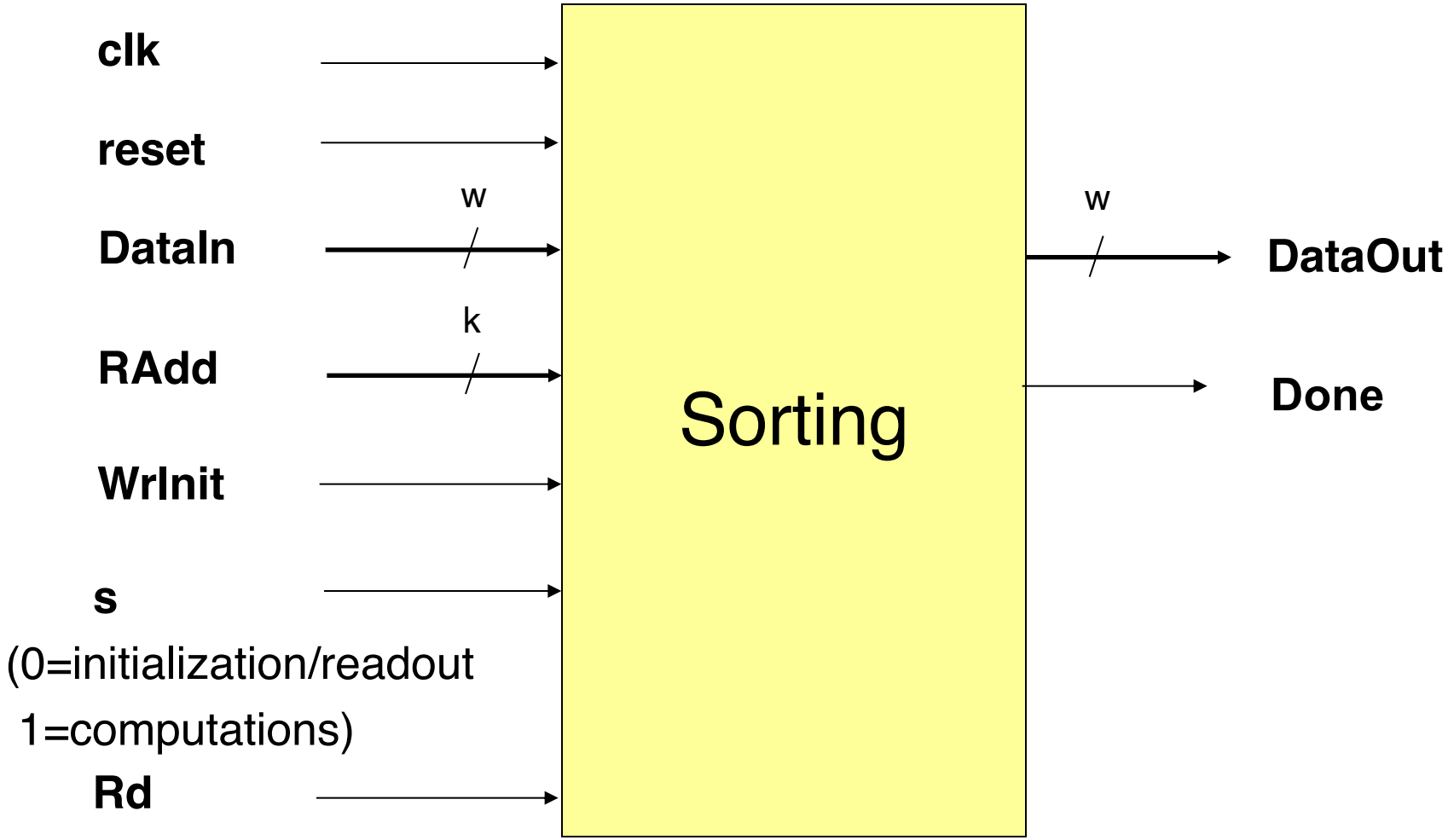
* declarations (.h) * implementations (.cpp) * testing

S2. Application based on functions of custom and standard cores

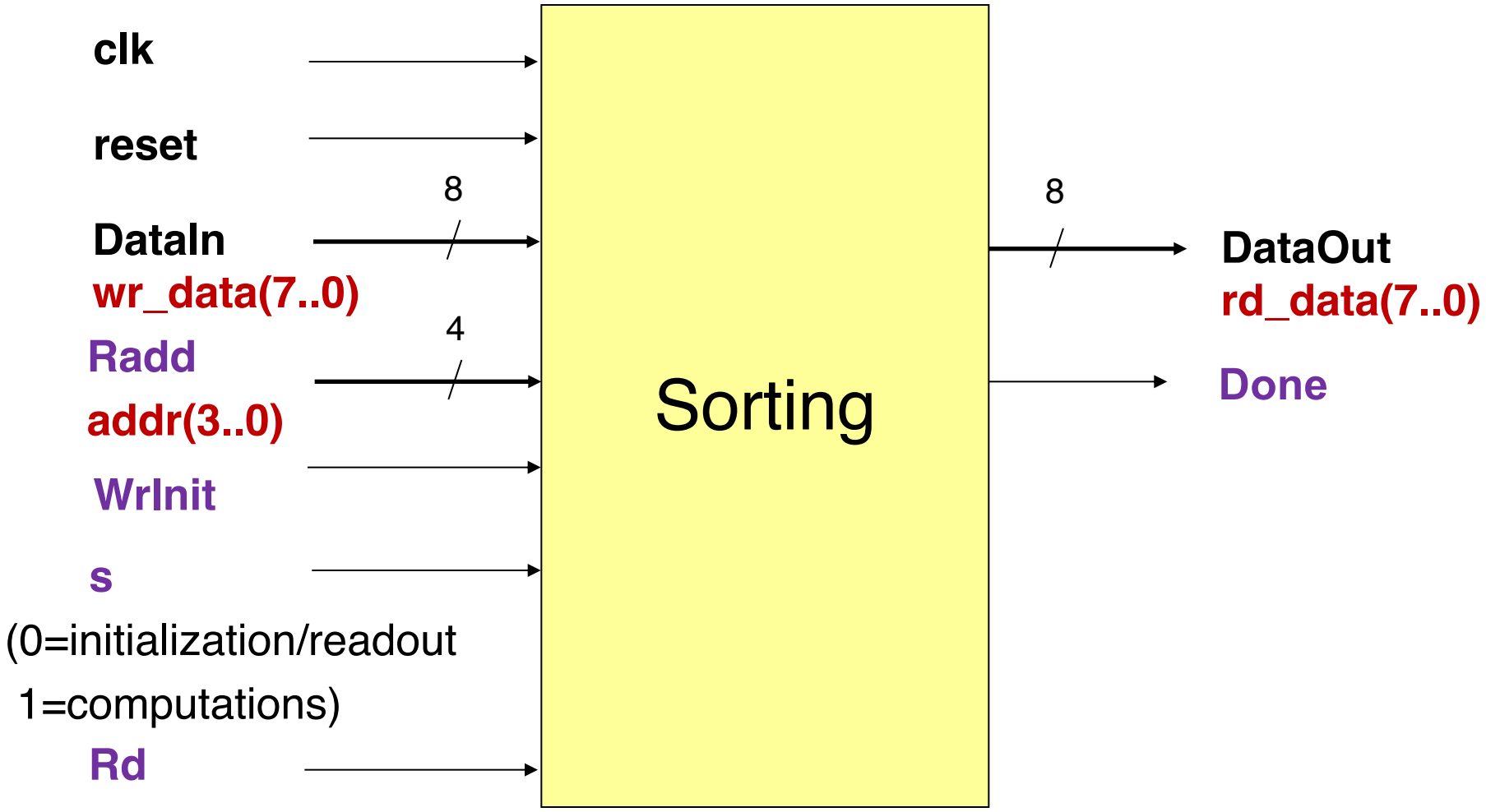
Interface of every MMIO Core

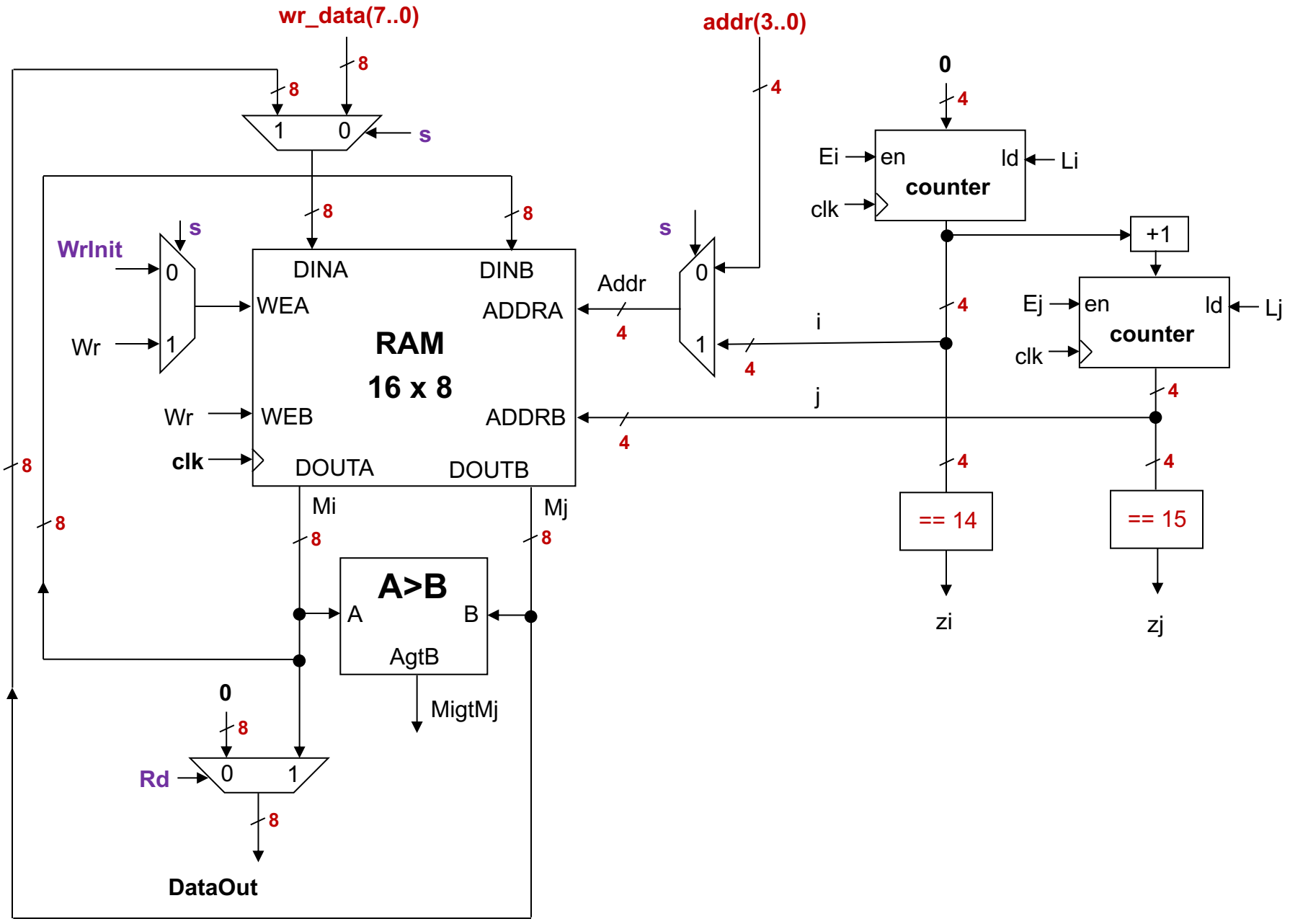


Sorting

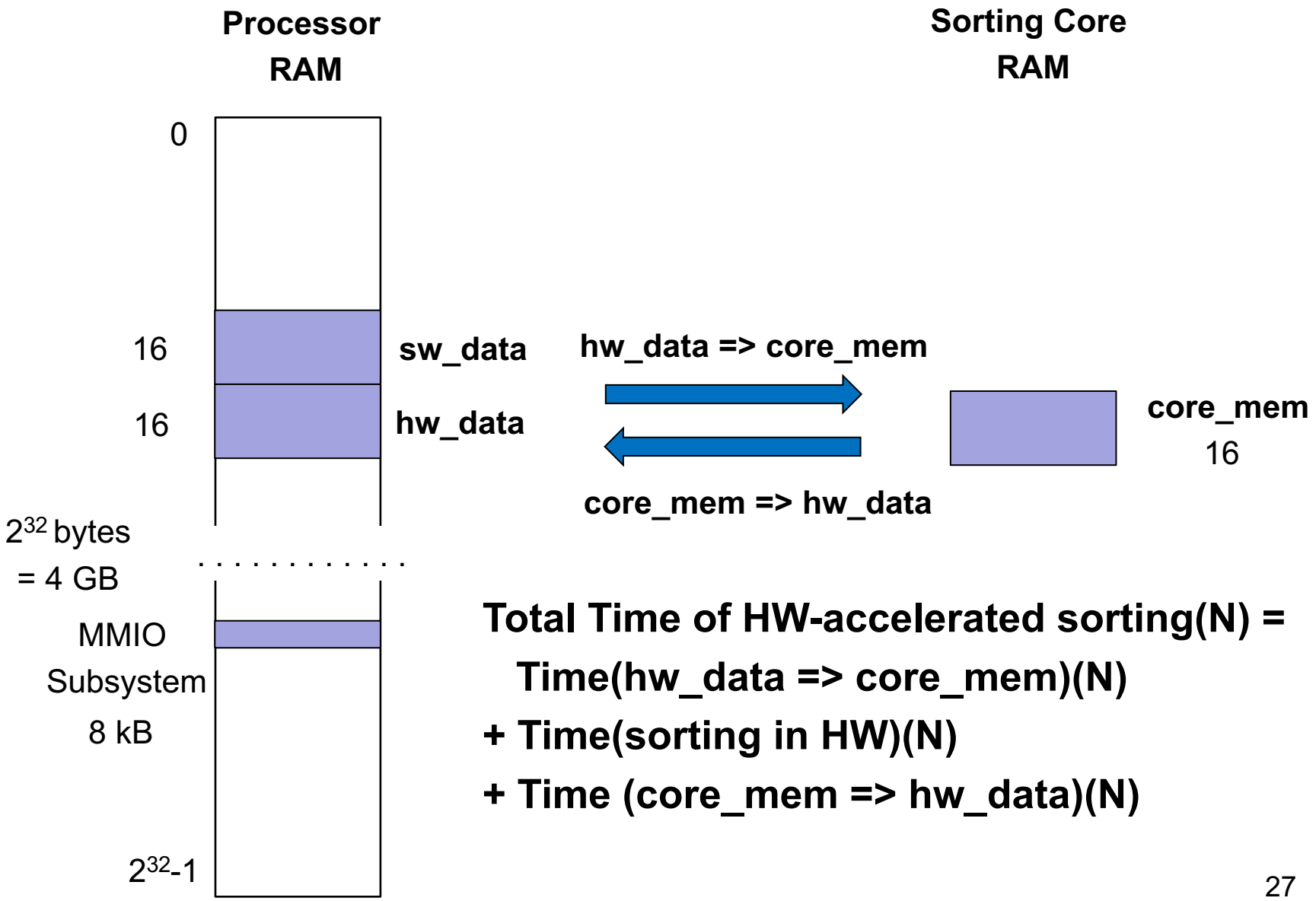

clk
reset
w
DataIn
k
RAdd
WrInit
s
(0=initialization/readout
1=computations)
Rd
Sorting
w
DataOut
Done

Sorting


clk
reset
8
DataIn
wr_data(7..0)
4
Radd
addr(3..0)
WrInit
s
(0=initialization/readout
1=computations)
Rd
Sorting
8
DataOut
rd_data(7..0)
Done


wr_data(7..0)
addr(3..0)
8
4
1
0
s
WrInit
Wr
clk
DINA
DINB
WEA
ADDRA
RAM
16 x 8
WEB
ADDRB
DOUTA
DOUTB
Addr
i
j
Mi
Mj
A>B
A
B
AgtB
MigtMj
Rd
DataOut
Ei
Li
Ej
Lj
en
ld
counter
+1
== 14
== 15
zi
zj

Data Transfer



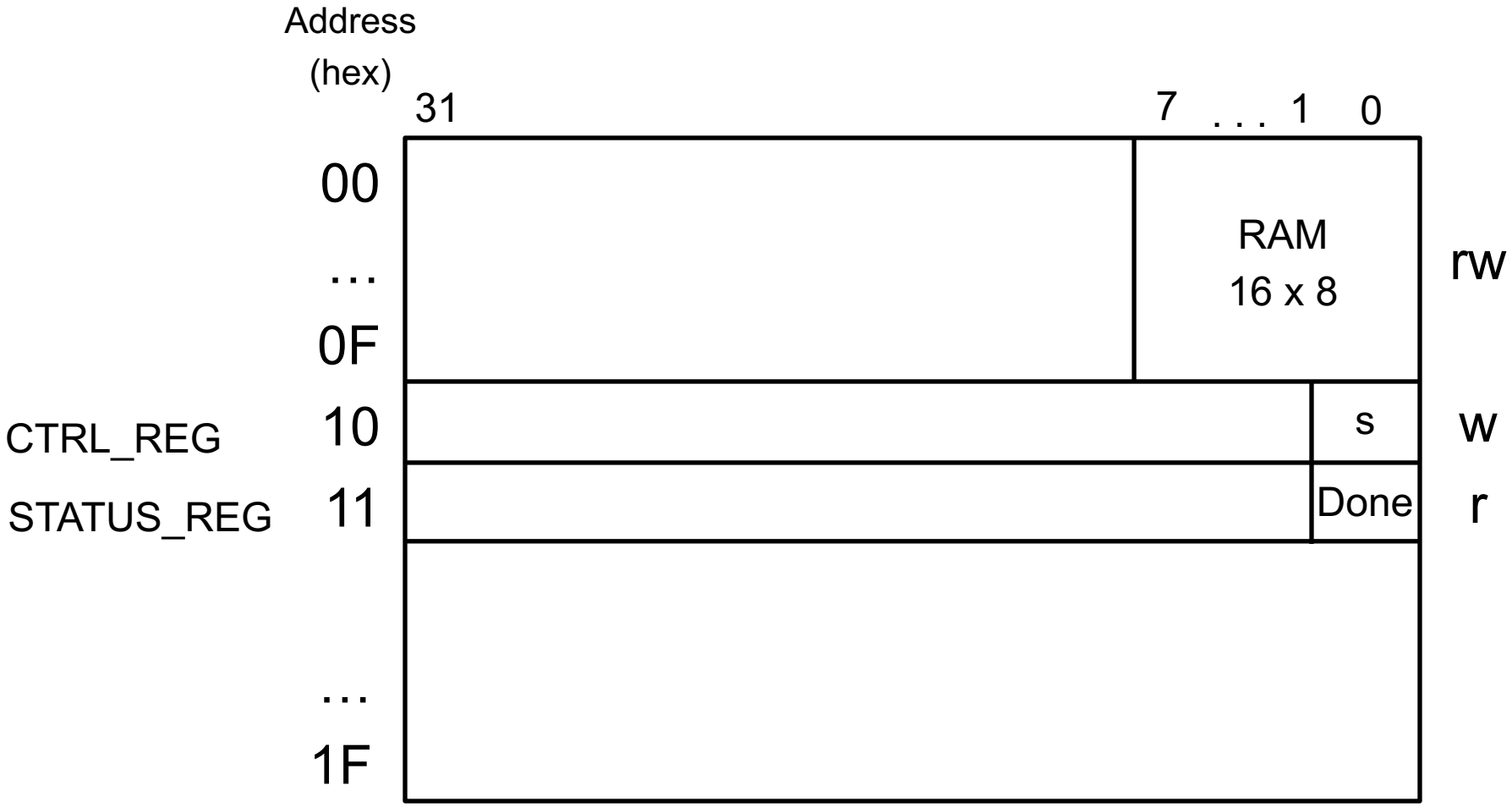
Total Time of HW-accelerated sorting(N) =

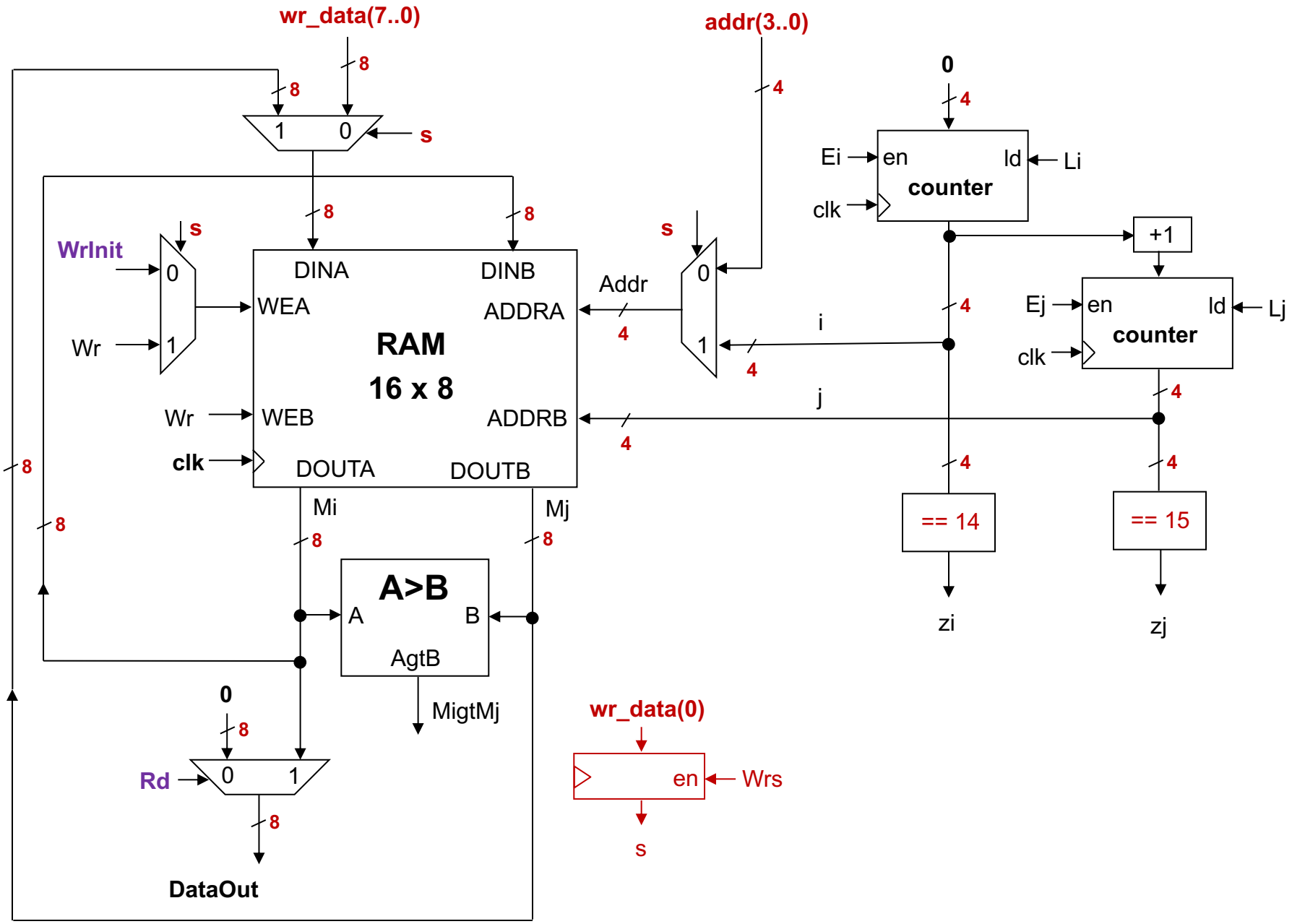
Time(hw_data => core_mem)(N)

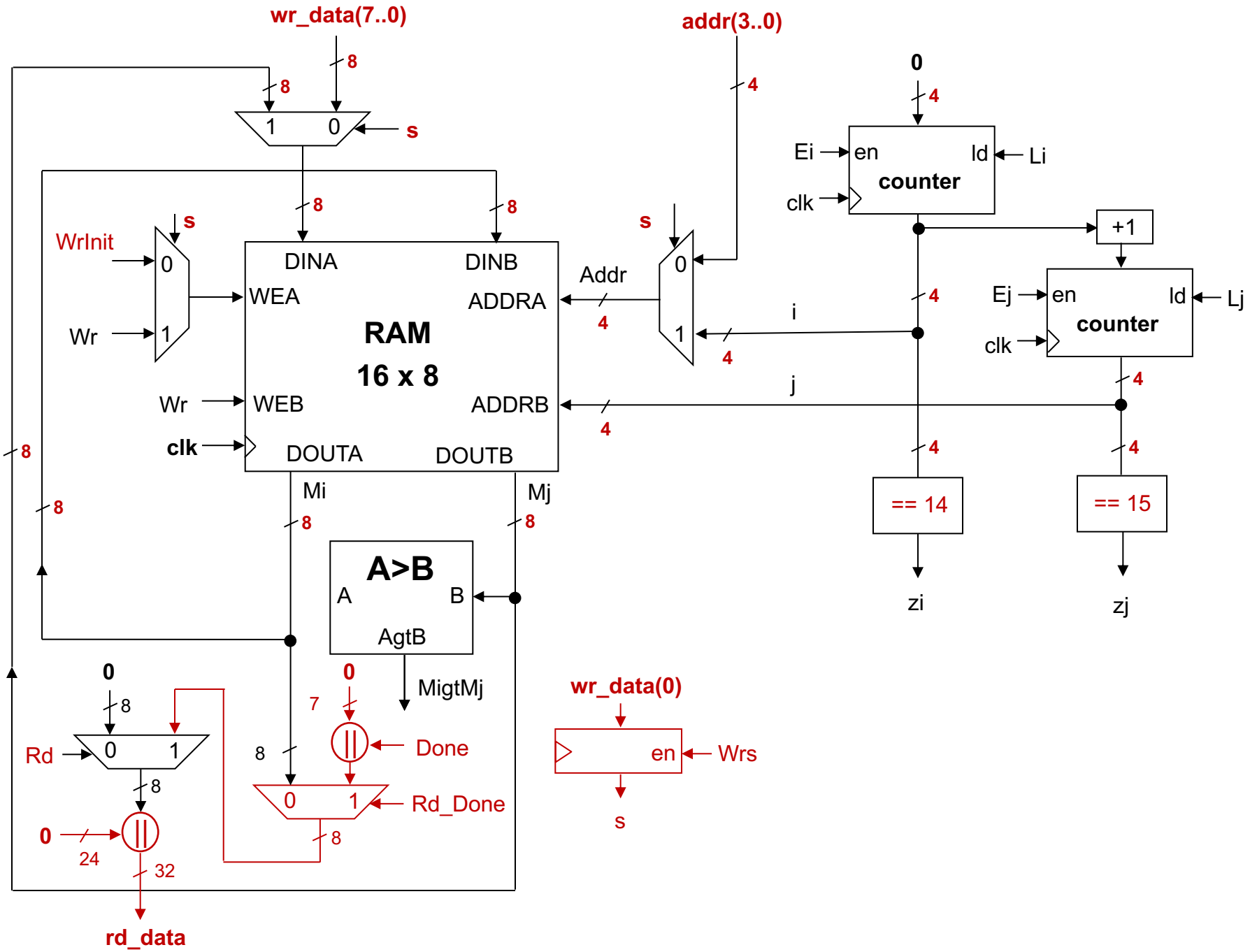
+ Time(sorting in HW)(N)

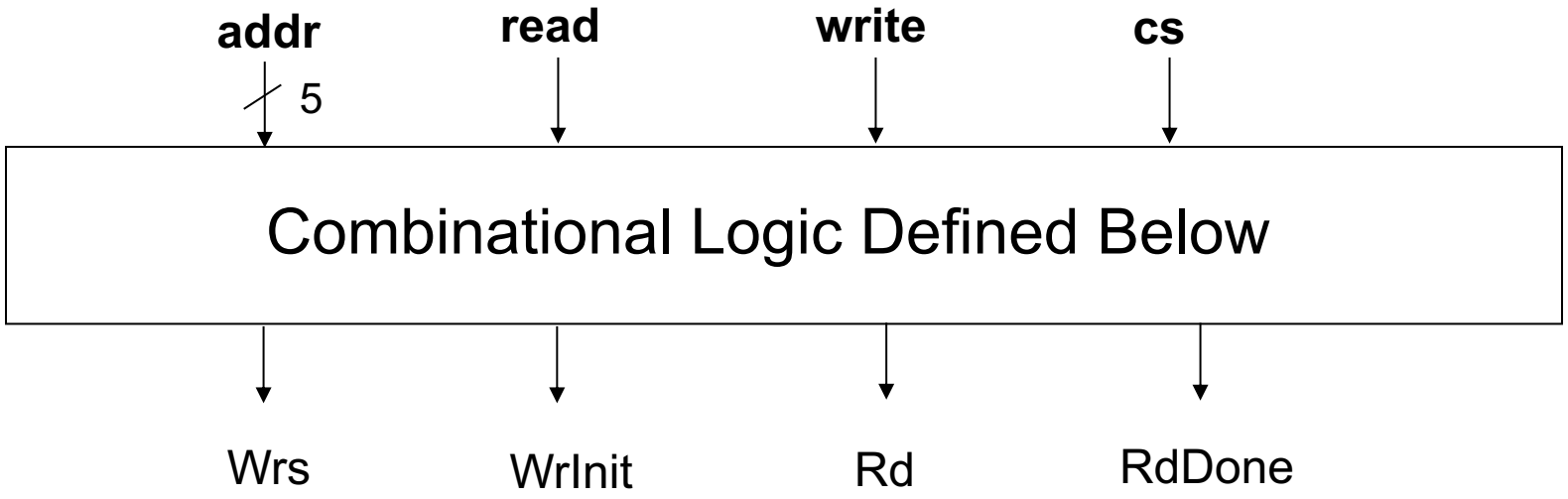
+ Time (core_mem => hw_data)(N)

I/O Register Map of the Sorting Core



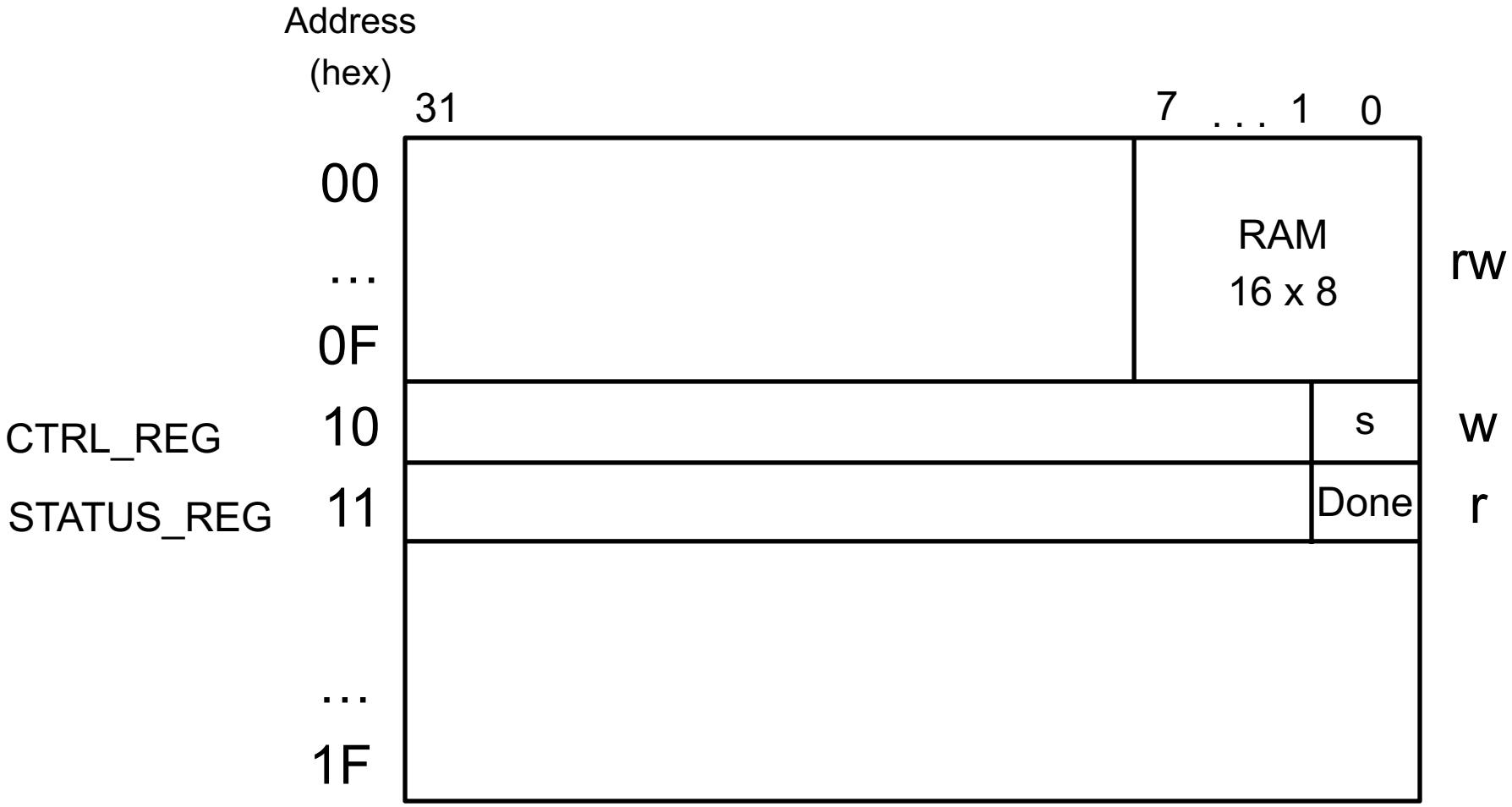

wr_data(7..0)
addr(3..0)
8
4
1
0
s
Ei
en
ld
Li
clk
counter
+1
Ej
Lj
WrInit
Wr
DINA
DINB
WEA
ADDRA
RAM
16 x 8
WEB
ADDRB
clk
DOUTA
DOUTB
Addr
i
j
Mi
Mj
A>B
A
B
AgtB
MigtMj
== 14
== 15
zi
zj
Rd
DataOut
wr_data(0)
Wrs


wr_data(7..0)
addr(3..0)
8
4
1
0
s
WrInit
Wr
clk
DINA
DINB
WEA
ADDRA
RAM
16 x 8
WEB
ADDRB
DOUTA
DOUTB
Addr
i
j
Mi
Mj
Ei
en
ld
Li
counter
Ej
Lj
+1
== 14
== 15
zi
zj
A>B
A
B
AgtB
MigtMj
Rd
Done
Rd_Done
24
32
rd_data
wr_data(0)
Wrs



Output	cs	write	read	addr
Wrs=1	1	1	0	0x10
WrInit=1	1	1	0	0x00..0x0F
Rd=1	1	0	1	--
RdDone=0	1	0	1	0x00..0x0F
RdDone=1	1	0	1	0x11

I/O Register Map 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

SortCore class definition in sort_core.h (1)

```
class SsegCore {
public:
   /* Register map */
   enum {
      MEM_LOW_ADDR = 0x00,
      MEM_HIGH_ADDR = 0x0F,
      CTRL_REG = 0x10,
      STATUS_REG =0x11
};

SortCore(uint32_t core_base_addr);

 ~SortCore();    // not used
```

SsegCore class definition in sort_core.h (2)

```
void init_readout();

void write(uint8_t addr, uint8_t data);

uint8_t read(uint8_t addr);

void sort();

uint8_t done();
```