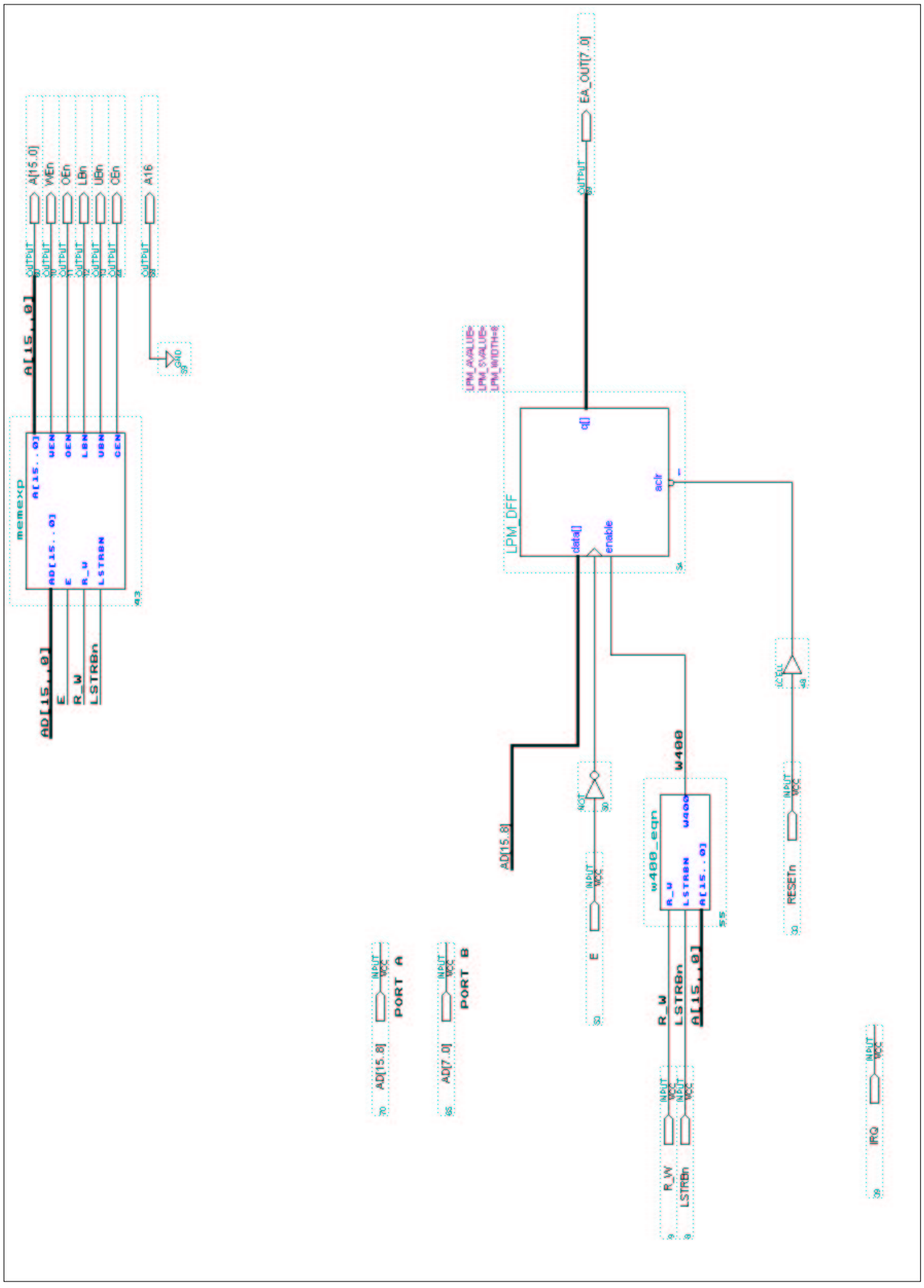
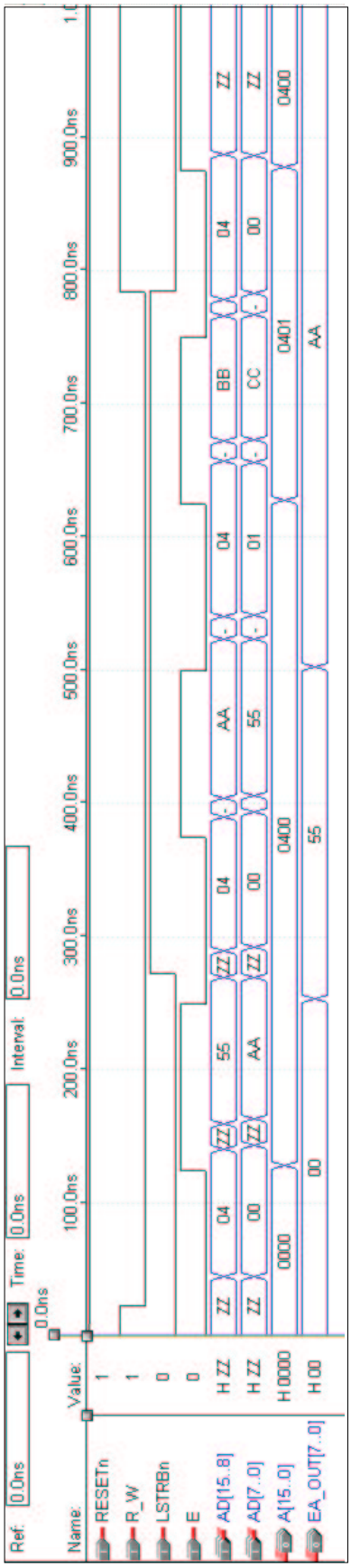
Building Expansion Ports in Altera

- We will build two bi-directions expansion ports in the Altera EPF7128 chip on the memory expansion board
- The following Altera GDF files show how this is done
- The first GDF file shows an output port.
  - The address is demultiplexed from AD[15..0] in the `memexp` block
  - The output value is latched into the set of 8 flip-flops on a write to address 0x0400
  - The D inputs of the flip-flops are connected to AD[15..8], the high (even) byte of the data bus
  - The data is latched on the rising edge of E
  - The flip-flop latches are enabled on a write to address 0x0400
  - Here is the Altera TDF file for determining when a write to address 0x0400 takes place:

```
SUBDESIGN w400_eqn
(
    R_W             :  INPUT;   % R/W Line %
    LSTRBn          :  INPUT;

    A[15..0]        :  INPUT;  % Demultiplexed address bits %

    w400            :  OUTPUT;
)


BEGIN

    if ((A[15..0] == H"0400") & (R_W == GND)) THEN
        w400 = VCC;
    ELSE
        w400 = GND;
    END IF;

END;
```

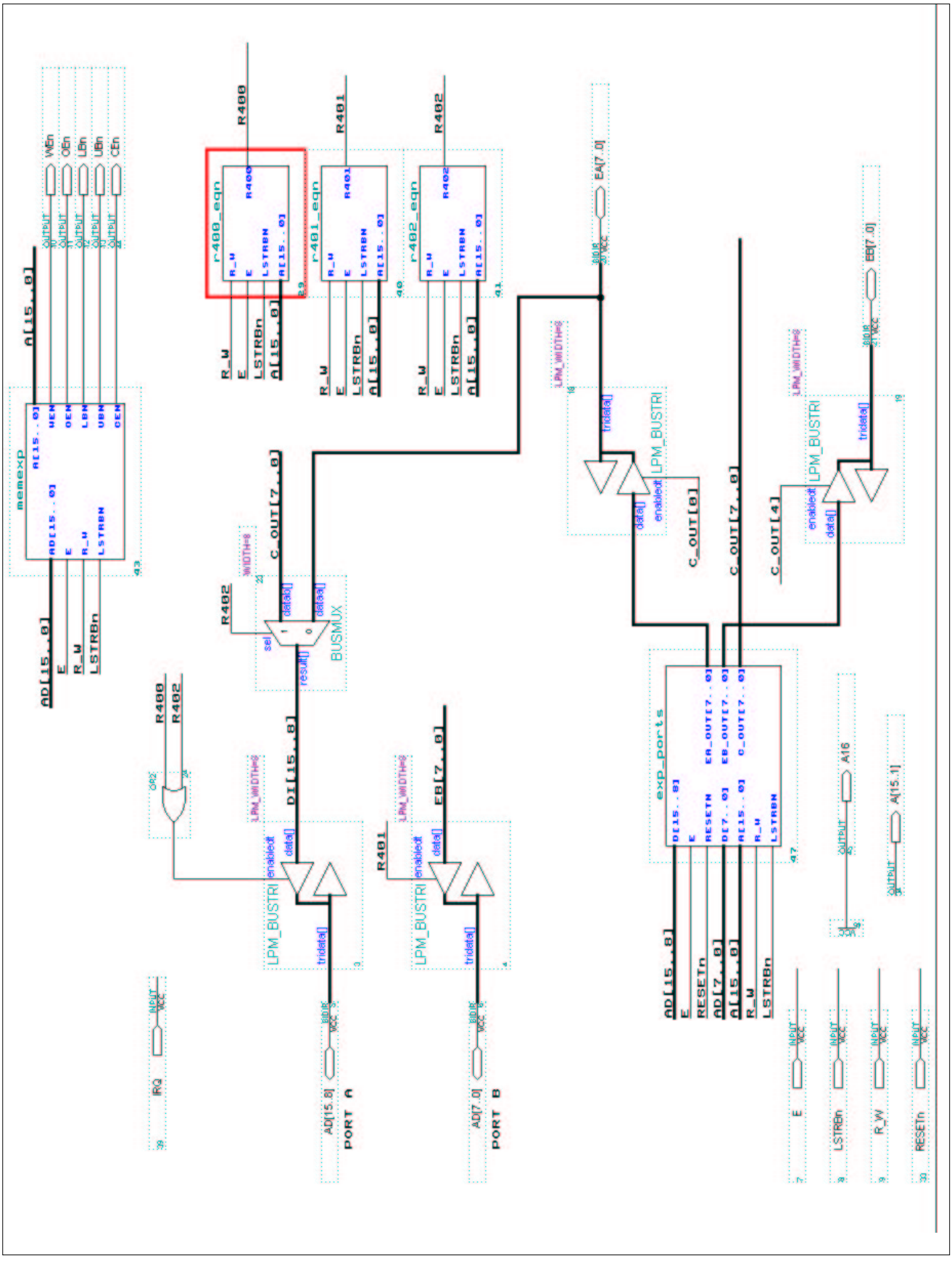

A[15..0]
WEn
OEn
LBn
UBn
OEn
A16
memexp
AD[15..0]
E
R_W
LSTRBn
EA_OUT[7..0]
LPM_AVALUE
LPM_SVALUE
LPM_WIDTH=8
LPM_DFF
q[]
aclr
data[]
enable
AD[15..8]
NOT
w400
w400_eqn
R_W
LSTRBn
A[15..0]
RESETn
PORT A
PORT B
AD[15..8]
AD[7..0]
R_W
LSTRBn
IRQ

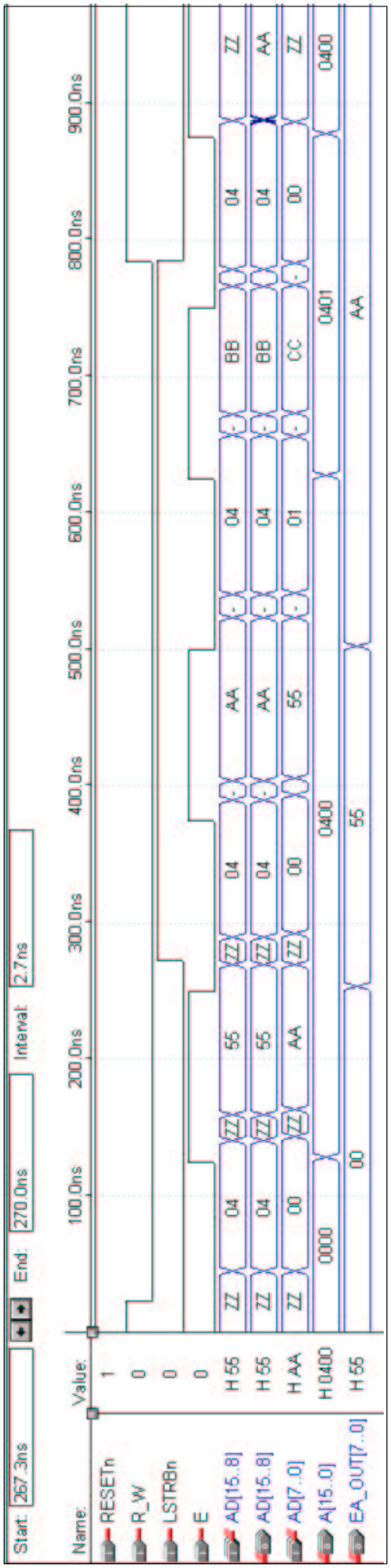

Ref: 0.0ns
Time: 0.0ns
Interval: 0.0ns
Name:
Value:
RESETn
R_W
LSTRBn
E
AD[15..8]
AD[7..0]
A[15..0]
EA_OUT[7..0]
100.0ns
200.0ns
300.0ns
400.0ns
500.0ns
600.0ns
700.0ns
800.0ns
900.0ns

Building Expansion Ports in Altera

- We now need to be able to read what was written to the expansion port
- We do this by putting a tri-state buffer between the ouput of the expansion port an the address/data bus
- We enable the tri-state buffer on a read from address 0x0400
- Here is the Altera TDF file which determines when we read from address 0x0400:

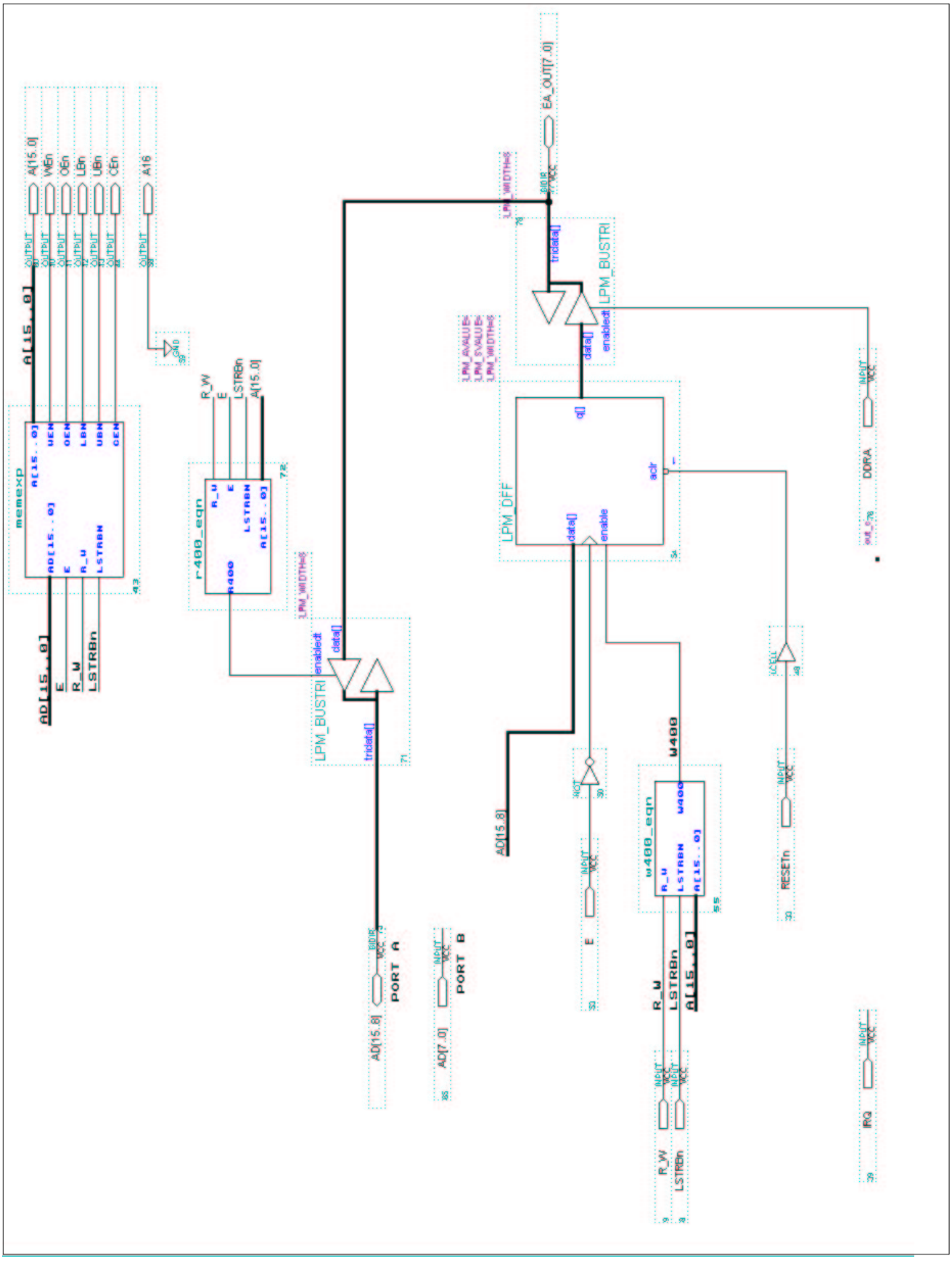
```
SUBDESIGN r400_eqn
(
    R_W             :  INPUT;   % R/W Line %
    E               :  INPUT;
    LSTRBn          :  INPUT;

    A[15..0]        :  INPUT;  % Demultiplexed address bits %

    r400            :  OUTPUT;
    )


BEGIN

    if ((A[15..0] == H"0400") & (R_W == VCC) & (E == VCC)) THEN
        r400 = VCC;
    ELSE
        r400 = GND;
    END IF;

END;
```

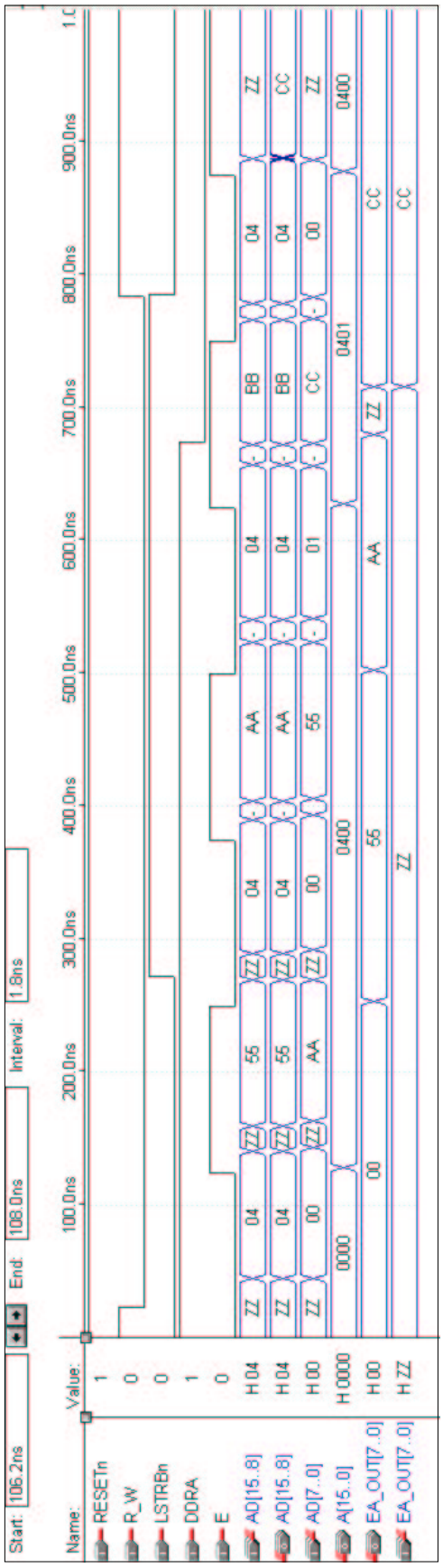

memexp
r400_eqn
r401_eqn
r402_eqn
R400
R401
R402
BUSMUX
LPM_BUSTRI
exp_ports
PORT A
PORT B
A[15..0]
AD[15..0]
DI[15..8]
EB[7..0]
C_OUT[7..0]
C_OUT[0]
C_OUT[4]
EA[7..0]
R_W
LSTRBn
RESETn
IRQ
E


Start: 267.3ns
End: 270.0ns
Interval: 2.7ns
Name:
Value:
100.0ns
200.0ns
300.0ns
400.0ns
500.0ns
600.0ns
700.0ns
800.0ns
900.0ns
RESETn
1
R_W
0
LSTRBn
0
E
0
AD[15..8]
H 55
AD[15..8]
H 55
AD[7..0]
H AA
A[15..0]
H 0400
EA_OUT[7..0]
H 55
ZZ 04 ZZ 55 ZZ 04 - AA - 04 - BB - 04 ZZ
ZZ 04 ZZ 55 ZZ 04 - AA - 04 - BB - 04 AA
ZZ 00 ZZ AA ZZ 00 - 55 - 01 - CC - 00 ZZ
0000 0400 0401 0400
00 55 AA

Building Expansion Ports in Altera

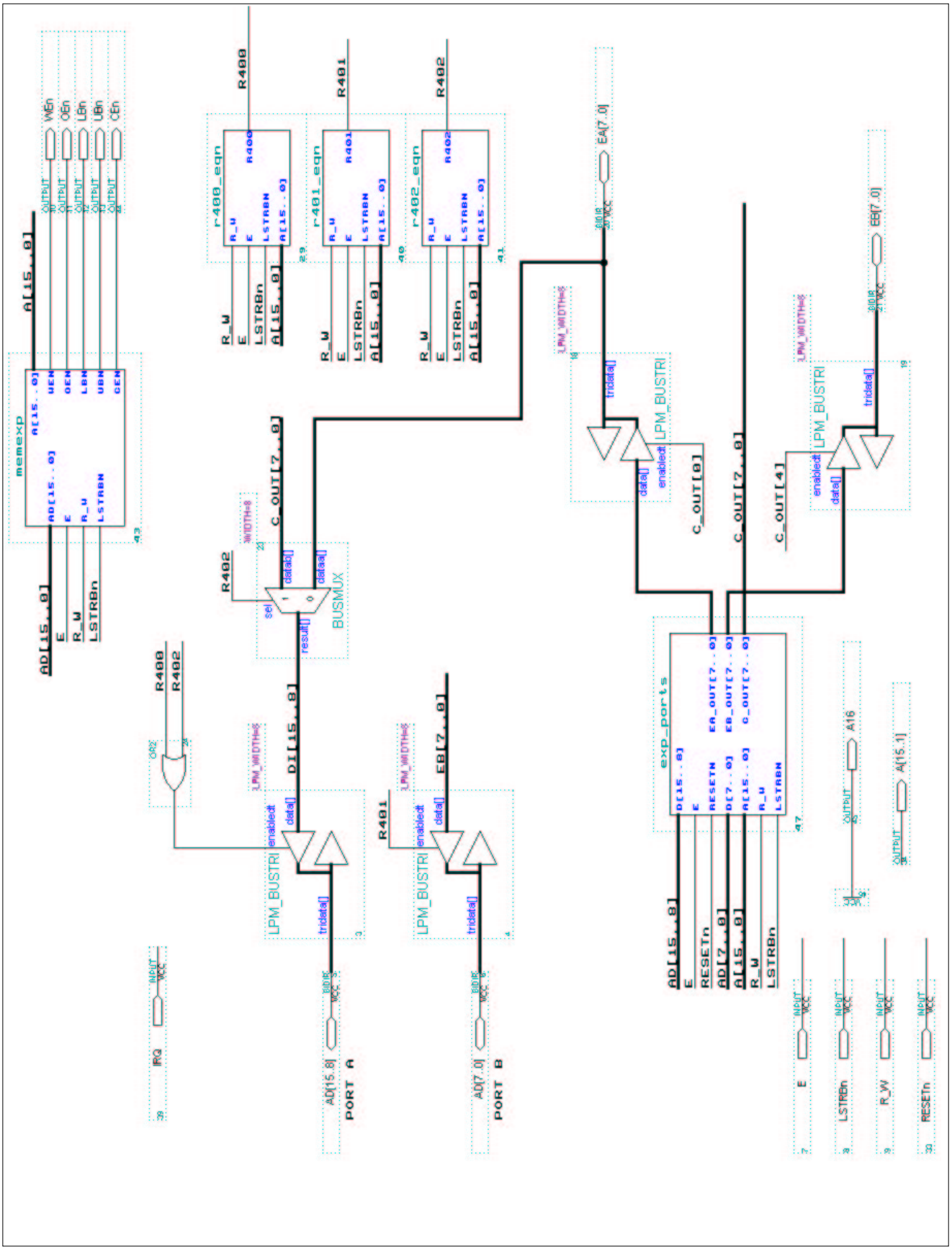
- We now need to make the expansion ports bi-directional
- We do this by putting a tri-state buffer between the output of the flip-flops and the expansion port pins
- We use a data direction input to allow us to choose whether to use the port as an input or output
- If the data direction bit is high, the outputs of the flip-flops are driven onto the expansion port pins, and the port is output
- If the data direction bit is low, the outputs of the flip-flops are not driven onto the expansion port pins, and the port is input
- The data direction bit is a bit of another register built into the Altera chip

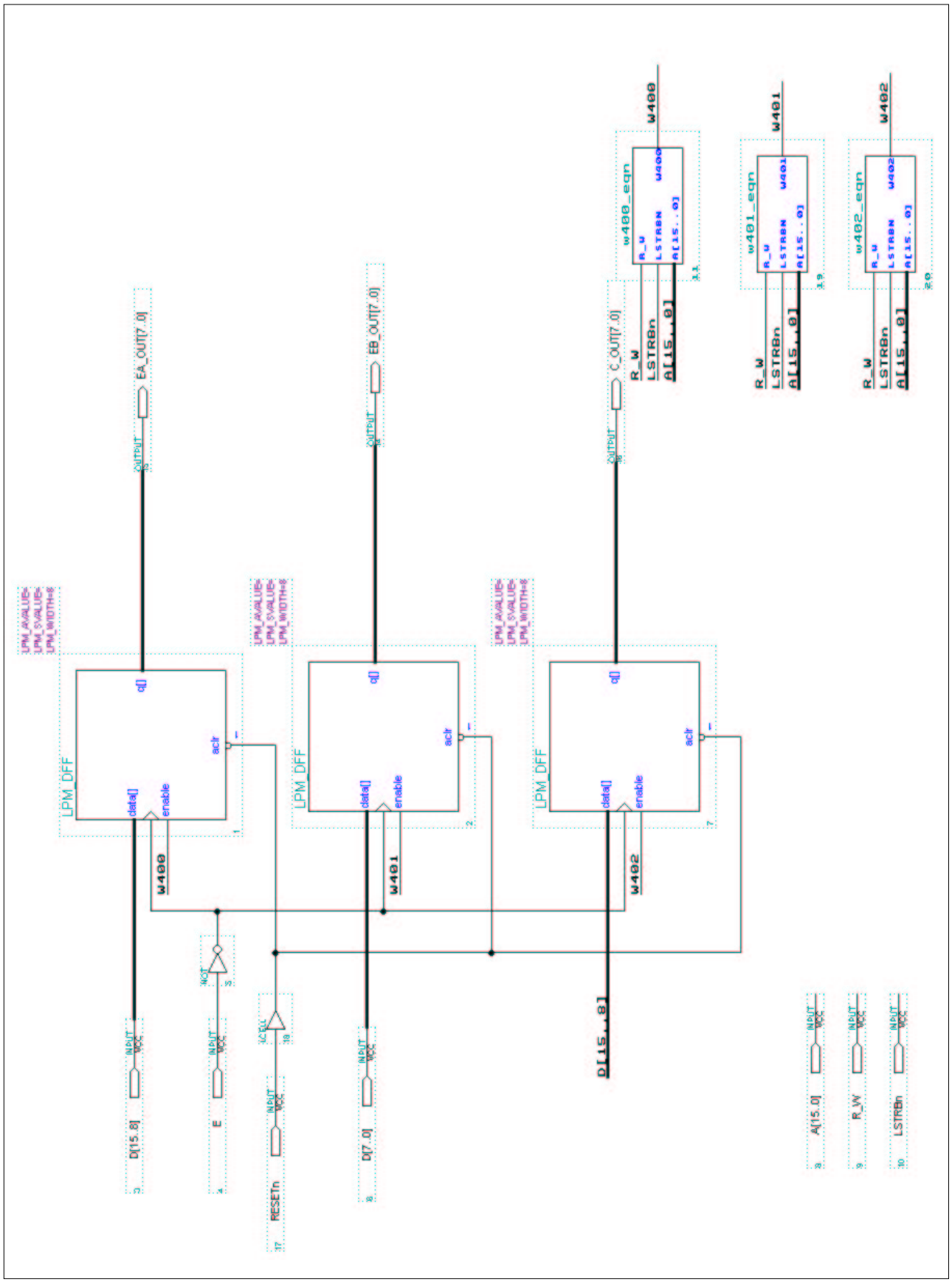

memexp
r488_eqn
w488_eqn
LPM_BUSTRI
LPM_DFF
A[15..0]
AD[15..0]
R_W
LSTRBn
EA_OUT[7..0]
A16
WEn
OEn
LBn
UBn
CEn
RESETn
IRQ
DDRA
PORT A
PORT B
AD[15..8]
AD[7..0]
INPUT
OUTPUT
NOT
VCC
GND

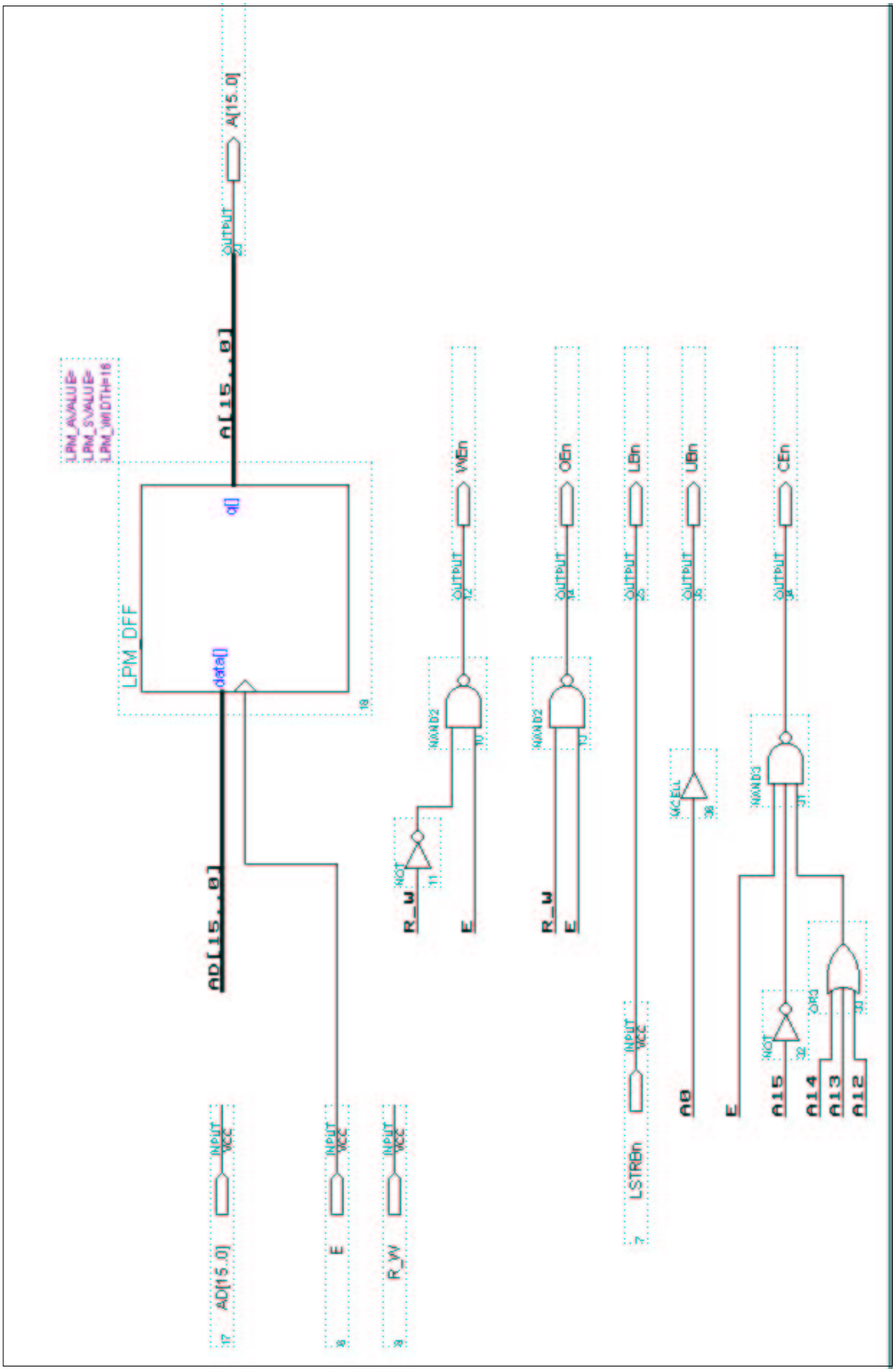

Start: 106.2ns
End: 108.0ns
Interval: 1.8ns
Name:
Value:
RESETn
R_W
LSTRBn
DDRA
E
AD[15..8]
AD[15..8]
AD[7..0]
A[15..0]
EA_OUT[7..0]
EA_OUT[7..0]
100.0ns
200.0ns
300.0ns
400.0ns
500.0ns
600.0ns
700.0ns
800.0ns
900.0ns

Building Expansion Ports in Altera

- We now add a second bi-directional expansion port at address 0x0401
- The data for this expansion port is connected to AD[7..0], the low data byte
- We also add a register at address 0x0402 to supply the data direction bits for the two new expansion ports
- Bit 0 of address 0x0402 controls the direction of Expansion Port A (EXPA) at address 0x0400
- Bit 4 of address 0x0402 controls the direction of Expansion Port B (EXPB) at address 0x0401


memexp
r-400_eqn
r-401_eqn
r-402_eqn
R400
R401
R402
R_W
E
LSTRBn
A[15..0]
AD[15..0]
WEn
OEn
LBn
UBn
CEn
C_OUT[7..0]
C_OUT[4]
C_OUT[0]
EA[7..0]
EB[7..0]
BUSMUX
LPM_BUSTRI
tridata[]
data[]
enabledt
exp_ports
RESETn
AD[7..0]
A16
A[15..1]
IRQ
PORT A
PORT B
AD[15..8]
AD[7..0]

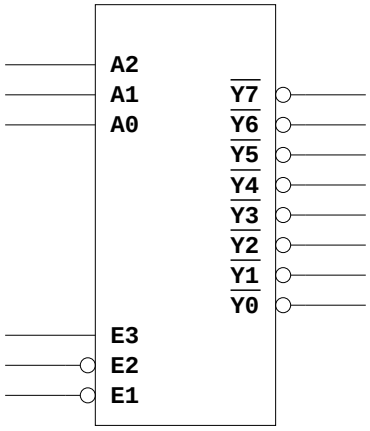

LPM_DFF
data[]
enable
aclr
q[]
EA_OUT[7..0]
EB_OUT[7..0]
C_OUT[7..0]
w400
w401
w402
D[15..8]
D[7..0]
E
RESETn
A[15..0]
R_W
LSTRBn
INPUT
OUTPUT
INVERT
ACLR
VCC


LPM_AVALUE=
LPM_SVALUE=
LPM_WIDTH=16
LPM_DFF
q[]
data[]
A[15..0]
AD[15..0]
OUTPUT
INPUT
VCC
NOT
NAND2
NAND3
WCELL
AD[15..0]
E
R_W
WEn
OEn
LBn
UBn
CEn
LSTRBn
A0
A15
A14
A13
A12

Using MSI Logic To Build An Output Port

- Many designs use standard MSI logic for microprocessor expansion
- This provides an inexpensive way to expand microprocessors
- One MSI device often used in such expansions is a decoder, such as the 74HC138 decoder chip

74HC138



When E3 high, E2 and E1 low, one of the outputs of the 74138 will go low

A2, A1, A0 determine which output will be low.

A2	A1	A0	Output
0	0	0	Y0
0	0	1	Y1
0	1	0	Y2
0	1	1	Y3
1	0	0	Y4
1	0	1	Y5
1	1	0	Y6
1	1	1	Y7