Address Decoding Techniques

Simple Exercise 1

- Determine how many address lines the following byte wide memory devices will have.

  - 256 bytes
  - 4k bytes
  - 32k bytes
  - 1M byte
  - 16M bytes

Address Decoding Techniques

Simple Exercise 2

- For a CPU with 16 address lines, how many address lines would connect to a full address decoder for the following memory devices.
  - 1k byte
  - 8k byte
  - 32k byte
  - 64k byte

Address Decoding Techniques

Simple Exercise 3

- How many blocks of memory of the following sizes fit into the 68000 memory map.
  - 256 bytes
  - 4k bytes
  - 32k bytes
  - 1M byte
  - 16M bytes

Address Decoding Techniques

Simple Exercise 4

- Determine how many memory chips of the following type are required to realise the following sizes of memory on a 68000 system and how they will be organised on the data bus.

Memory	Using
2k bytes	1k x 8 devices
16k bytes	4k x 4 devices
64k bytes	16k x 1 devices

Address Decoding Techniques

Simple Exercise 5

- What is the range of addresses occupied by the following blocks of memory, assuming the base (or start) address shown below:-

Base Address	Block Size
0F 0000	16k bytes
2F 0000	64k bytes
C0 0000	4M bytes

Address Decoding Techniques

Process for creating address decoders

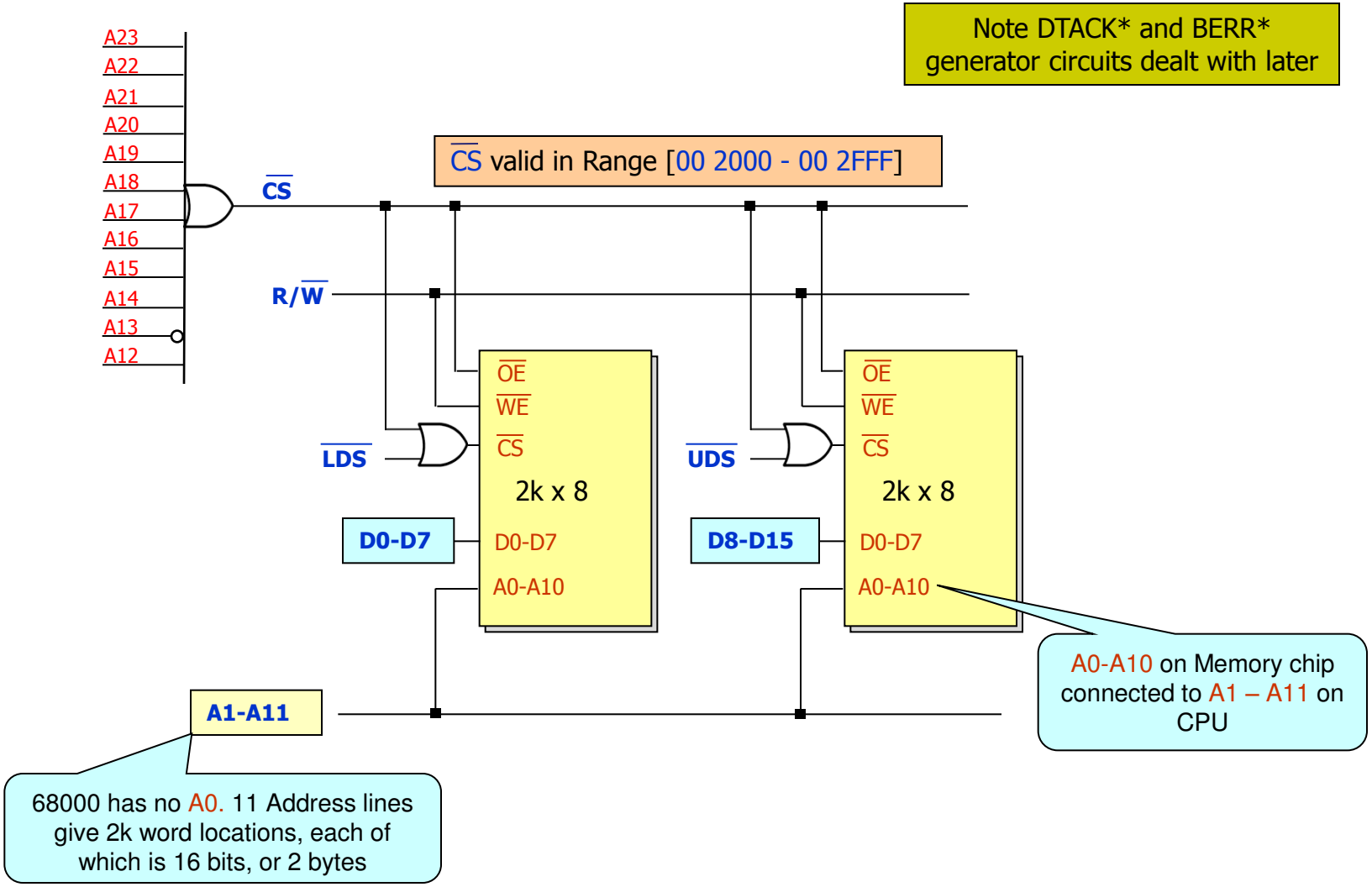
- Decide how much total memory you need (i.e. size in bytes)
- Decide how to organize it – 68000 wants 16 bit wide data, with individual byte or word access so use UDS* and LDS* to access each byte/word.
- Determine how many 16 bit banks of memory you need to meet total need.
- Determine how many memory chips you need in each bank to make a 'x16' memory.
- Determine how many address lines the memory chips in that bank will need e.g. a 16k location bank has 14 address lines (a0 – a13 on chip). These connect to CPU address lines starting at a1 - a14 on CPU.
- Unused CPU address lines go to the address decoder.
- Assign base/start addresses to each bank making sure they align to a natural boundary for the bank size. This will happen automatically if you feed unused address lines to the decoder.
- Figure out the range of addresses Start – End address.

Address Decoding Techniques

Exercise 6

- Design a FULL address decoder for a 68000 based system that has a total of 4k bytes of RAM beginning at a base address of hex 002000 using 2k x 8 ram chips.

Typical 2k Word Memory Interface to 68000 CPU



Two 2k byte devices working in parallel to give 2k words (or 4k bytes)

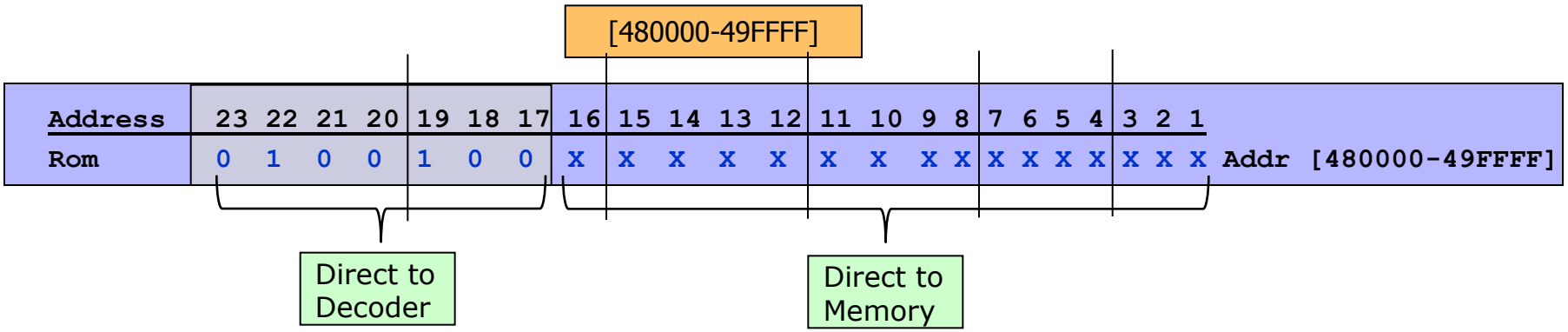
Address Decoding Techniques

Exercise 7

- Design a FULL address decoder for a 68000 system with 128k bytes of RAM constructed from 64k byte devices, to reside at a base address of $480000
  - How many banks do we need?
  - How many Address lines will our decoder need to resolve?
  - What is the entire Address Range of the RAM, i.e. from $480000 - ??????

Solution 7

- The memory must be organised into 16 bit words, thus our 128k bytes of Ram could be constructed using two 64k byte devices working in parallel, one to handle/store the data on d0-d7, the other for d8-d15. So we need just 1 bank of 128k bytes
- Each 64k byte device connects to 16 CPU address lines (2^{16} = 64k)
- Thus we connect A0-A15 on each memory chip to A1-A16 on the 68000 leaving A17-A23 to drive the decoder.
- UDS* and LDS* should be used to select each device individually or together as a word.
- The full address range of the RAM is from the specified base address up to base address plus 128k bytes i.e. $480000 + $01FFFF = $49FFFF
- The address table and decoder circuit for this design is given below.



$\overline{SEL}$ = ! (!A23 . A22 . !A21 . !A20 . A19 . !A18 . !A17)

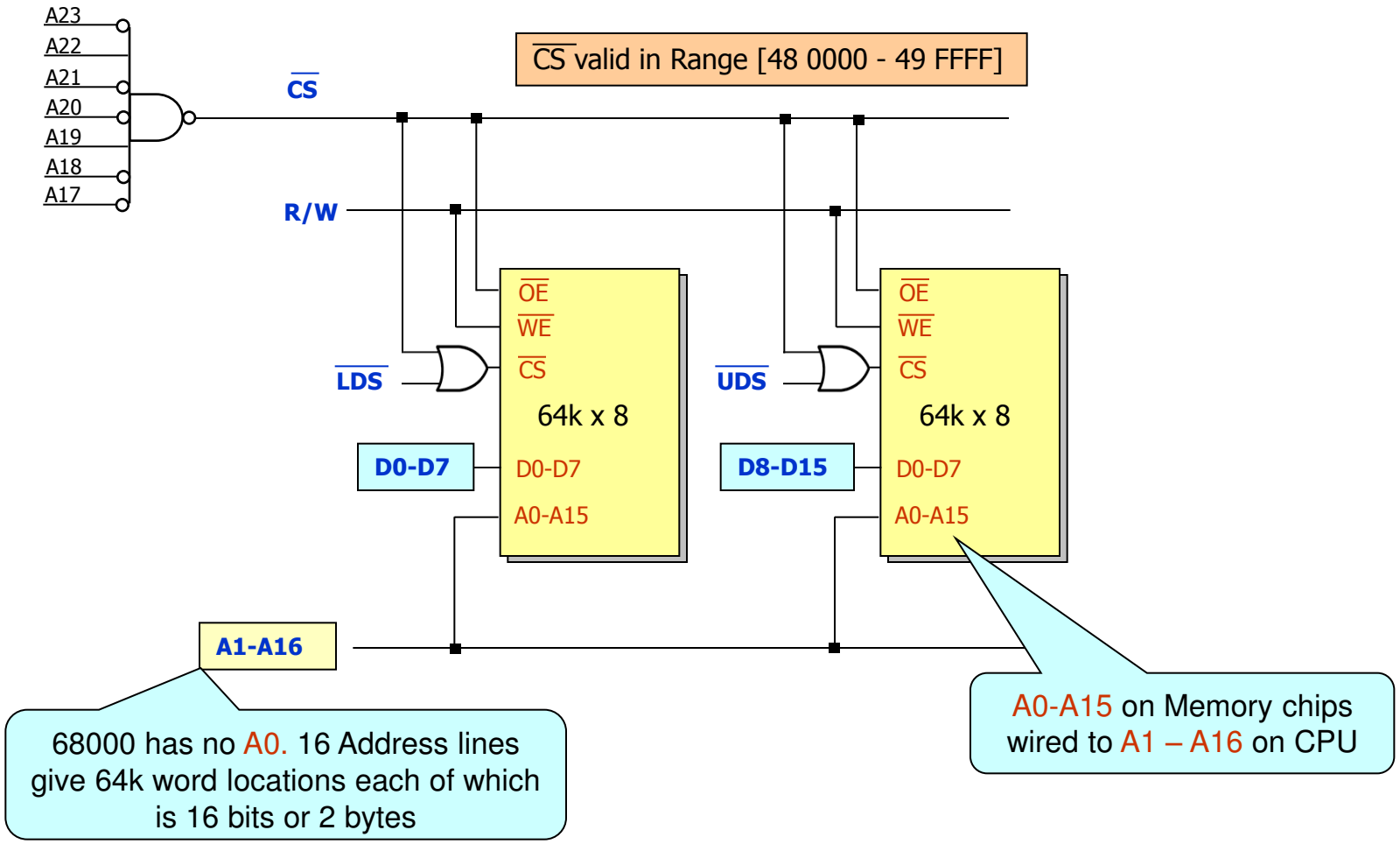
In VHDL Dataflow notation

```
Sel <= Not(Not A23 and A22 and Not A21 and Not A20 and A19 and Not A18 and Not A17) ;
```

In VHDL Process notation

```
if(Address(23 downto 17) = "0100100") then
        Sel <= '0' ;
else
        Sel <= '1' ;
end if ;
```

Typical 64k Word Memory Interface to 68000 CPU



Two 64k byte devices working in parallel to give 64k words (or 128K bytes)

Address Decoding Techniques

Exercise 8

- Design a full address decoder for the following 68000 based system:
  1. A block of 256k bytes of Rom, using 128k byte devices, located at base address 0 upwards.
  2. A block of 4M bytes of Ram using 512k * 4 bit devices (any start address).
  3. A block of 4 Mbytes of Ram using 1M x 1 bit chips (any start address).

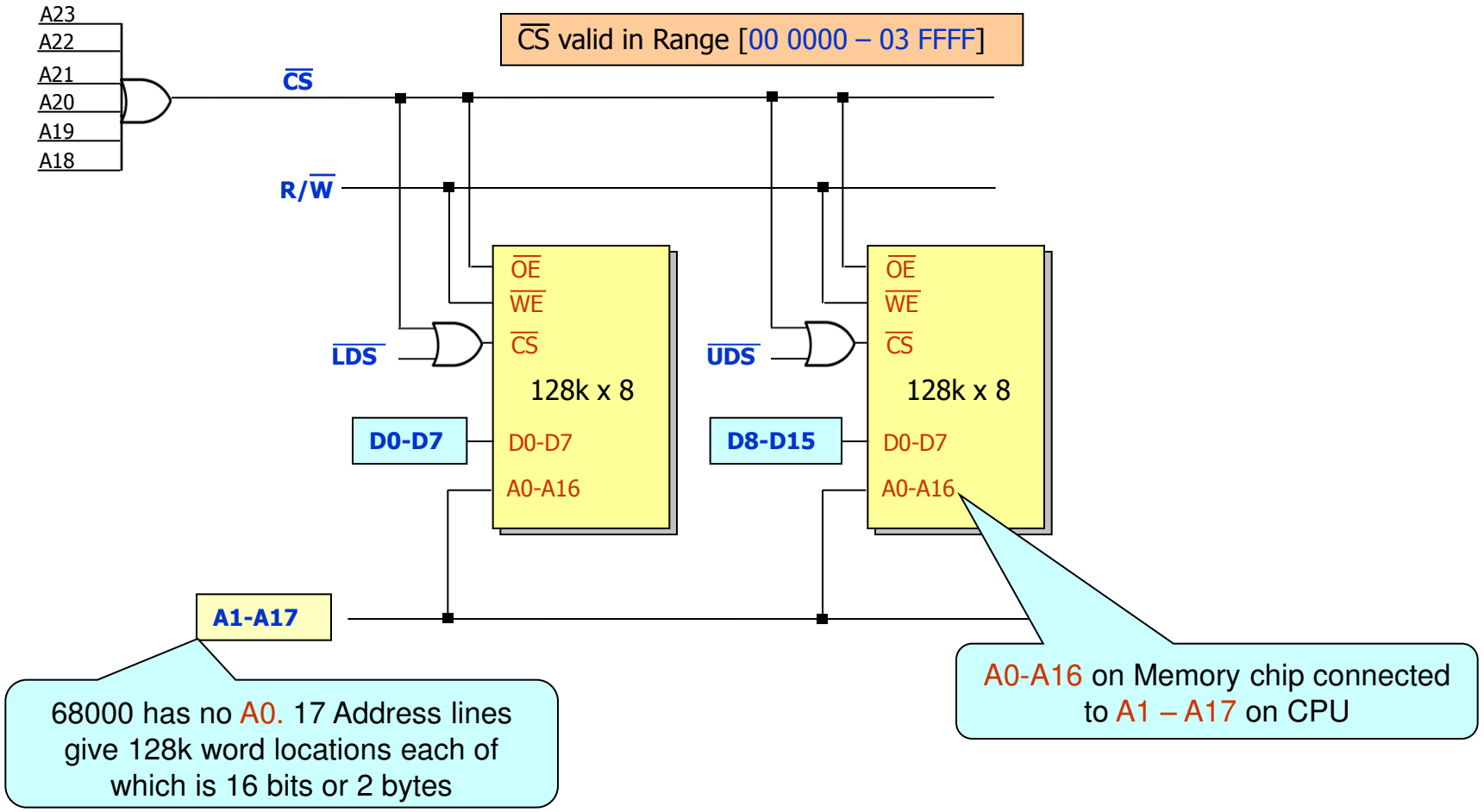
Solution 8: The 256k byte Block of Rom

- We need two 128k byte devices giving us 256k bytes. One device handles data on d0 - d7, the other on d8 - d15. That is 1 bank of rom meaning 1 decoder.
- Each 128k device has 17 address lines (a0-a16) which we connect to CPU a1-a17
- The address table looks like this.

Address	23 22 21 20	19 18	17 16	15 14 13 12	11 10 9 8	7 6 5 4	3 2 1	
Rom	0 0 0 0	0 0	x x	x x x x	x x x x	x x x x	x x x	Addr [000000-03FFFF]

- $\overline{\text{RomCS}}$ = (A23 + A22 + A21 + A20 + A19 + A18)
- UDS* and LDS* can then be gated with $\overline{\text{RomCS}}$ to produce a chip select for the upper or lower memory devices as we saw in Exercises 5 and 6.

Typical 256k Byte Rom Interface to 68000 CPU



Two 128k byte devices working in parallel to give 128k words (or 256K bytes)

Address Decoding Techniques

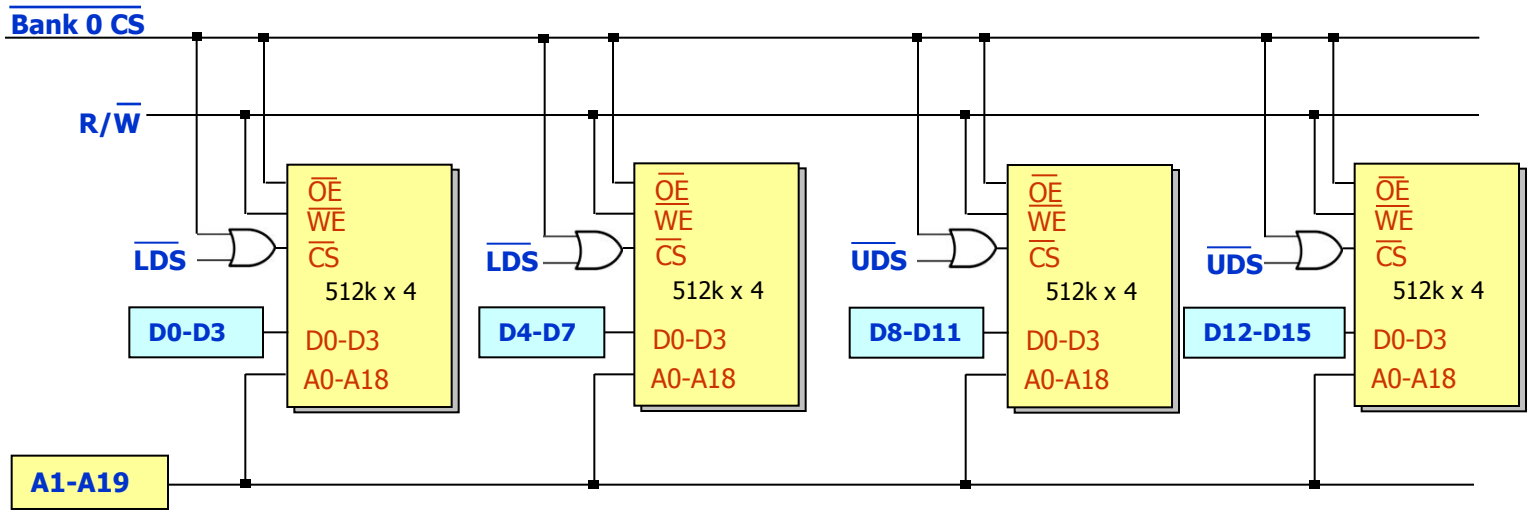
Solution 8 - 4 Mbytes Block of Ram using 512k * 4 bit devices

- To work with the 68000 we need to arrange this much Ram into banks of 16 bit wide data.
- Therefore we need 4 devices per bank giving us a bank size of 1MByte (or 512K Words).
- Two devices in each bank therefore respond to LDS* and connect to [D0-D3, D4-D7], while the other two devices respond to UDS* and use [D8-D11, D12-D15].
- Each bank of 512k words will thus require 19 address lines directly (2^{19} = 512K), leaving 4 CPU address lines (A20-A23) for use by our decoder.
- To get 4MBytes of ram for our system, we need to replicate this arrangement 4 times (i.e. 4 banks of 512k words), and 4 address decoders, one for each bank.
- We must be careful to ensure that our Ram addresses do not overlap with those of the earlier Rom at location 0, so the Ram is placed *arbitrarily* at address $800000 upwards, placing each bank at the start of a natural 1MByte address boundary. The address table thus looks like this.

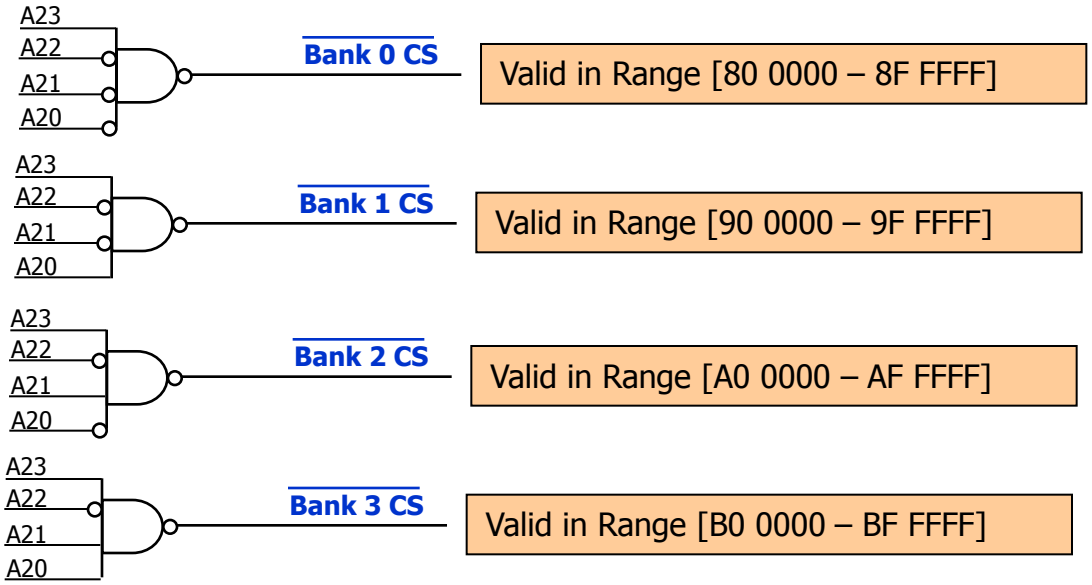
Address	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	
Bank0	1	0	0	0	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	Addr [800000-8FFFFF]
Bank1	1	0	0	1	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	Addr [900000-9FFFFF]
Bank2	1	0	1	0	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	Addr [A00000-AFFFFF]
Bank3	1	0	1	1	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	Addr [B00000-BFFFFF]

Logic Equations

$$\overline{\text{Bank0 CS}} = !(A23 \,.\, !A22 \,.\, !A21 \,.\, !A20) \quad \text{(Active low CS)}$$
$$\overline{\text{Bank1 CS}} = !(A23 \,.\, !A22 \,.\, !A21 \,.\, A20) \quad \text{(Active low CS)}$$
$$\overline{\text{Bank2 CS}} = !(A23 \,.\, !A22 \,.\, A21 \,.\, !A20) \quad \text{(Active low CS)}$$
$$\overline{\text{Bank3 CS}} = !(A23 \,.\, !A22 \,.\, A21 \,.\, A20) \quad \text{(Active low CS)}$$



1 Bank of 512K word (1Mbyte). Replicate 4 times to give 4Mbytes



4 address decoders.
1 per bank

Address Decoding Techniques

Solution 8 : 4M bytes of Ram using 1M x 1 bit chips.

To work with the 68000 we again need to arrange the Ram into banks of 16 bit data.

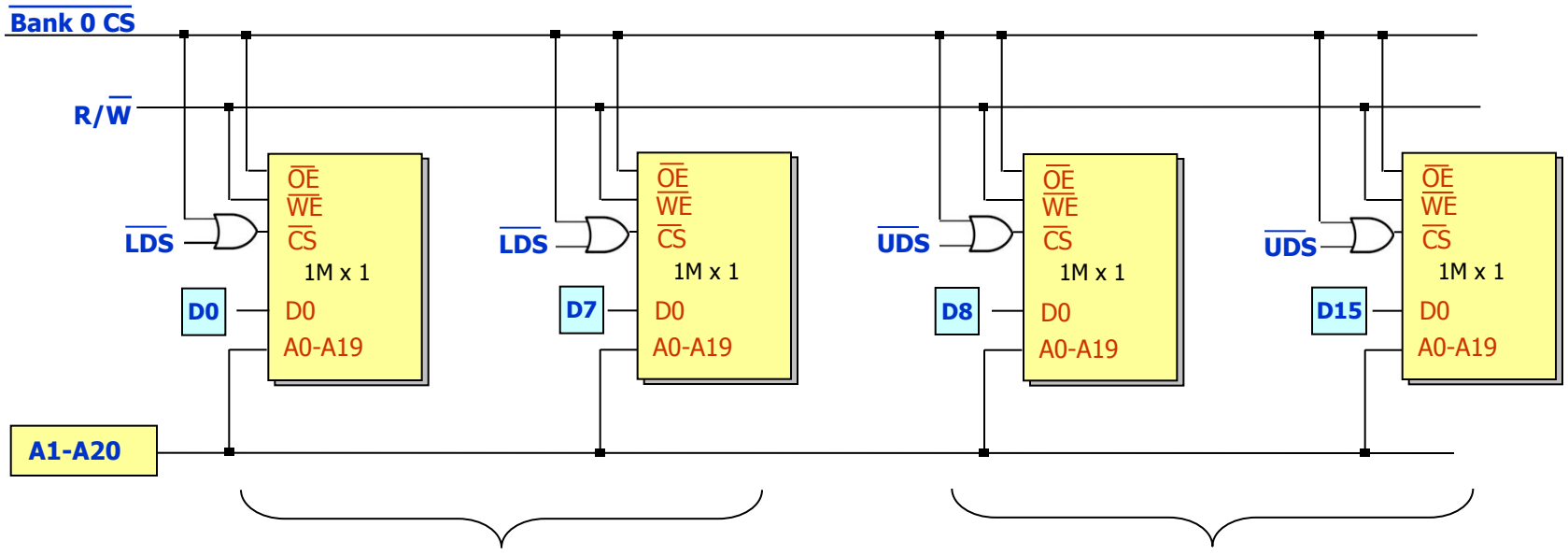
- Here we need 16 devices per bank giving us a bank size of 1M words (2Mbytes)
- 8 devices in each bank connect to LDS* and D0-D7, 8 devices connect to UDS* and D8-D15.
- Each 1M memory chip will have 20 address lines (2^{20} = 1M), a0-19 which connect to a1-a20 on CPU leaving 3 address lines to feed the decoder.
- 4M Bytes can thus be implemented as 2 Banks of 1M word, thus we need 2 address decoders.
- We must be careful to ensure that our addresses do not overlap with those of the Rom or Ram1 and it would be desirable to make Ram 2 continue where Ram 1 left off, i.e. base address $C00000
- Placing each bank on a natural 2MByte boundary, the address table looks like this.

Address	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	
Bank0	1	1	0	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	addr [C00000-DFFFFF]
Bank1	1	1	1	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	addr [E00000-FFFFFF]

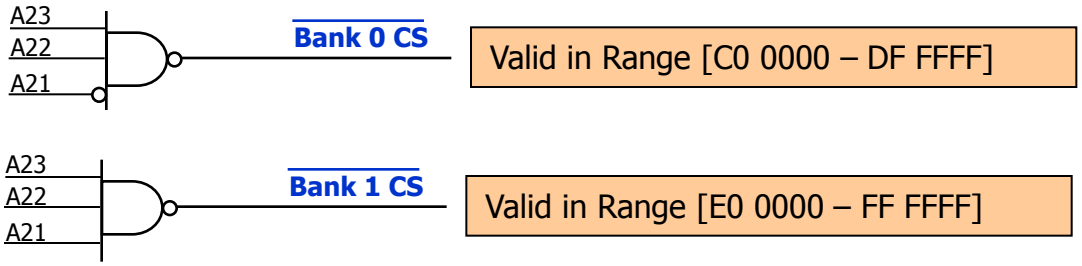
Logic Equations

$\overline{\text{Bank0 CS}}$ = !(A23.A22.!A21) (Active low CS)

$\overline{\text{Bank1 CS}}$ = !(A23.A22.A21) (Active low CS)



1 Bank of 1M word (2Mbyte). Replicate twice to give 4Mbytes



Address Decoding Techniques

Exercise 9

Design a full address decoder for the following 68000 based system.

- 2 Mbytes of Ram composed of 256 k * 8 bit devices located at base address $A00000
- 256 Kbytes of Rom composed of 128 K * 8 bit devices located at base address $F80000

Solution 9: The Ram

- The 2Mbytes (or 1M word) Ram can be composed of 4 banks of 256k Words (i.e. 512k bytes), where each bank uses a pair of byte wide devices in parallel with one device connecting to LDS*/D0-D7 and the other to UDS*/D8-D15.
- 4 Chip select lines will be required, one for each bank.
- Each chip/bank will have 18 address lines (2^{18} = 256k) leaving 5 for the decoder.
- The address table could look like this.

Address	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	
Bank0	1	0	1	0	0	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	Addr [A00000-A7FFFF]
Bank1	1	0	1	0	1	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	Addr [A80000-AFFFFF]
Bank2	1	0	1	1	0	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	Addr [B00000-B7FFFF]
Bank3	1	0	1	1	1	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	Addr [B80000-BFFFFF]

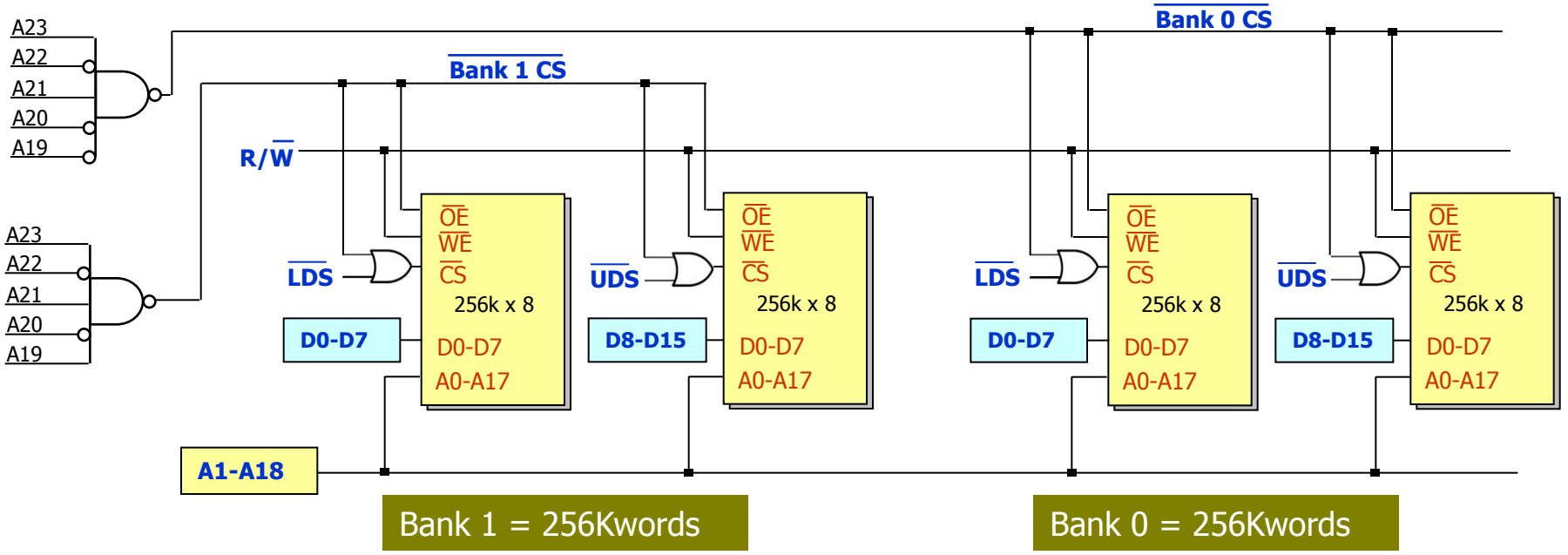
Logic Equations

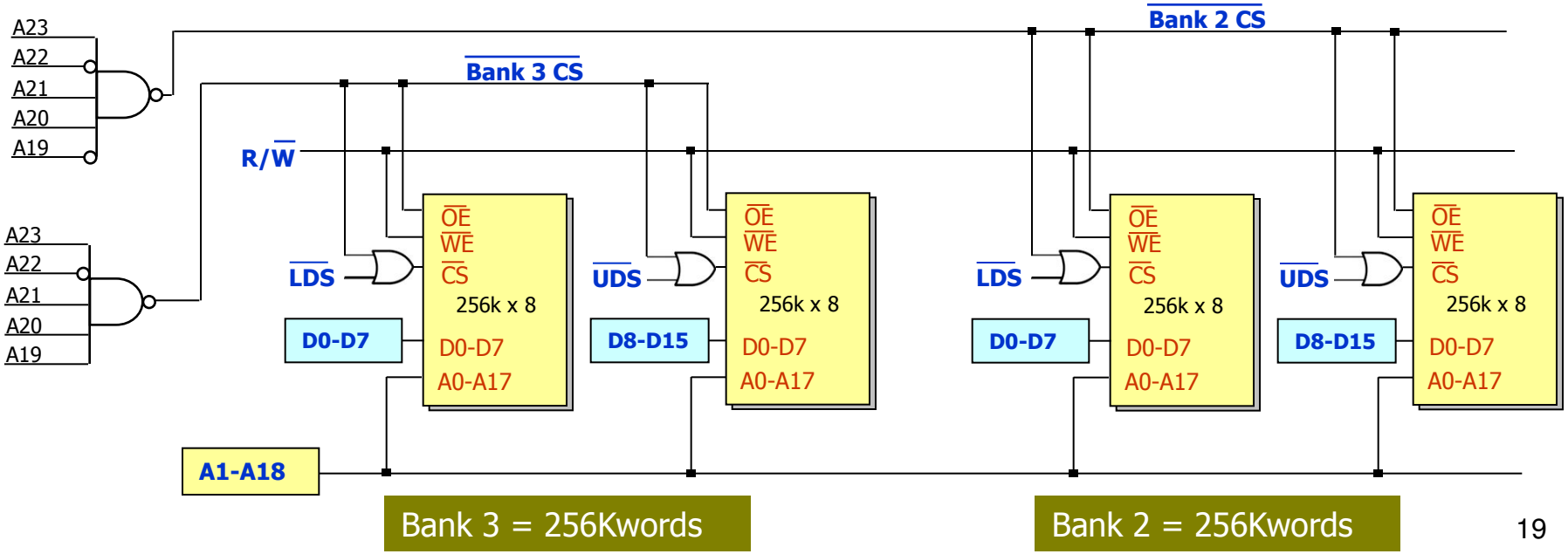
$\overline{\text{Bank0 CS}} = !(A23 \,.\, !A22 \,.\, A21 \,.\, !A20 \,.\, !A19)$ (Active low CS)

$\overline{\text{Bank1 CS}} = !(A23 \,.\, !A22 \,.\, A21 \,.\, !A20 \,.\, A19)$ (Active low CS)

$\overline{\text{Bank2 CS}} = !(A23 \,.\, !A22 \,.\, A21 \,.\, A20 \,.\, !A19)$ (Active low CS)

$\overline{\text{Bank3 CS}} = !(A23 \,.\, !A22 \,.\, A21 \,.\, A20 \,.\, A19)$ (Active low CS)


A23
A22
A21
A20
A19
Bank 1 CS
Bank 0 CS
R/W
LDS
UDS
OE
WE
CS
256k x 8
D0-D7
D8-D15
A0-A17
A1-A18
Bank 1 = 256Kwords
Bank 0 = 256Kwords


A23
A22
A21
A20
A19
Bank 3 CS
Bank 2 CS
R/W
LDS
UDS
OE
WE
CS
256k x 8
D0-D7
D8-D15
A0-A17
A1-A18
Bank 3 = 256Kwords
Bank 2 = 256Kwords

Address Decoding Techniques

Exercise 9 Rom Solution

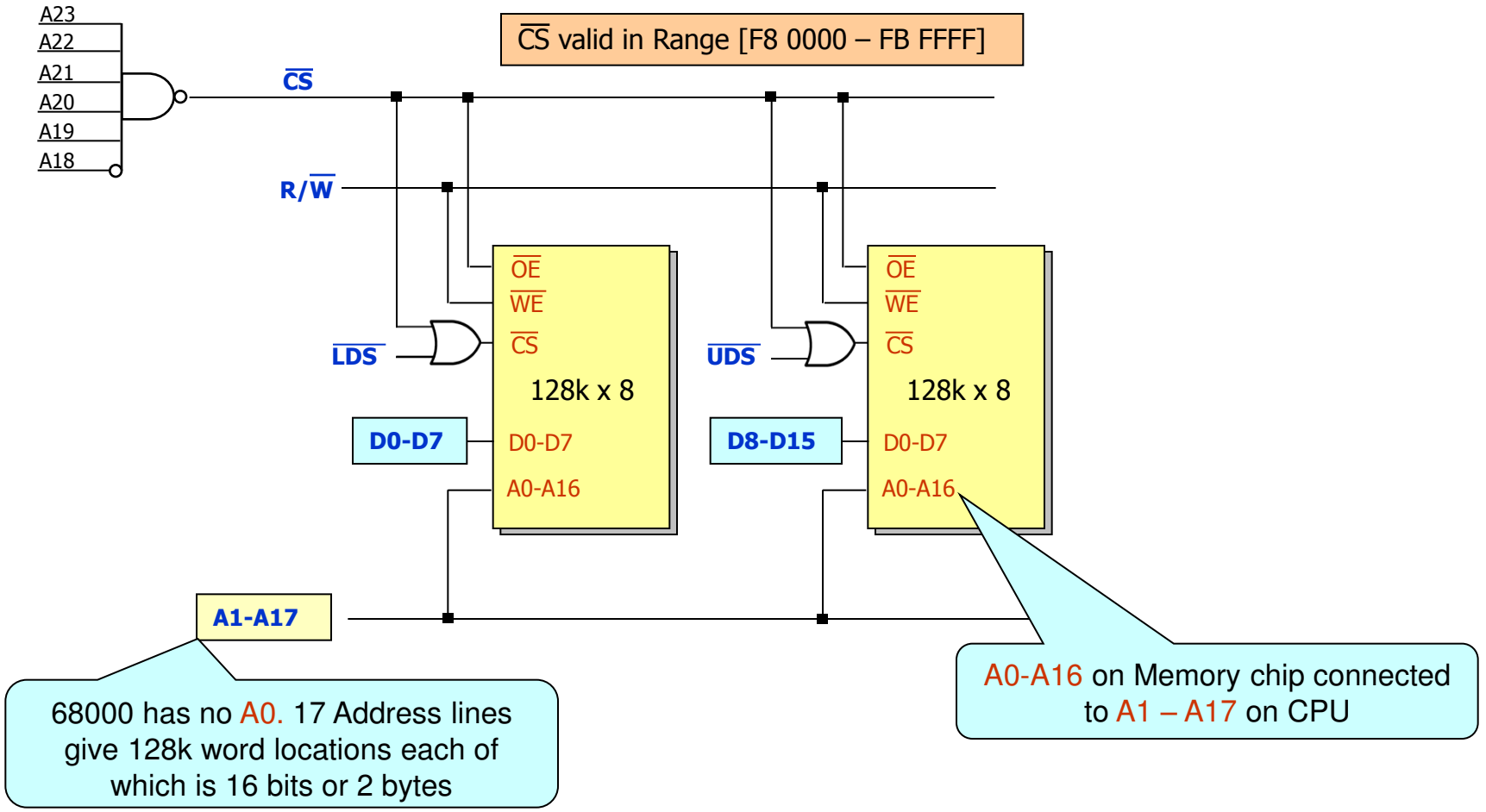
- The 256k bytes (or 128k words) of Rom at base address hex F80000 can be composed of a single bank of 2 x 128 kbyte devices working in parallel, with one device connecting to LDS*/D0-D7 and the other to UDS*/D8-D15.
- Only 1 chip select/address decoder is thus required.
- The bank will use 17 address lines (2^{17} = 128k) leaving 6 address lines for the decoder.
- The Address Table would look like this

Address	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	
Bank0	1	1	1	1	1	0	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	addr [F80000-FBFFFF]

Logic Equations

$\overline{\text{Bank0 CS}}$ = !(A23.A22.A21.A20.A19.!A18) (Active low CS)

Typical 128k word Rom Interface to 68000 CPU



Two 128k byte devices working in parallel to give 128k words (or 256K bytes)

Address Decoding Techniques

Exercise 10

- Design a full address decoder that locates three 16 bit wide blocks of memory in the range
  - Hex 00 0000 - 7F FFFF
  - Hex A0 8000 - A0 8FFF
  - Hex F0 0000 – FF FFFF

Solution 10:

- The Address Tables looks like this

Address	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	
addr1	0	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	addr [000000-7FFFFF]
addr2	1	0	1	0	0	0	0	0	1	0	0	0	x	x	x	x	x	x	x	x	x	x	x	addr [A08000-A08FFF]
addr3	1	1	1	1	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	addr [F00000-FFFFFF]

Logic Equations - (All Active low CS)

```
addr1 = A23
addr2 = (!A23 + A22 + !A21 + A20 + A19 + A18 + A17 + A16 + !A15 + A14 + A13 + A12)
addr3 = !(A23 . A22 .  A21 . A20)
```