

# LCD Display (2)

Hsi-Pin Ma

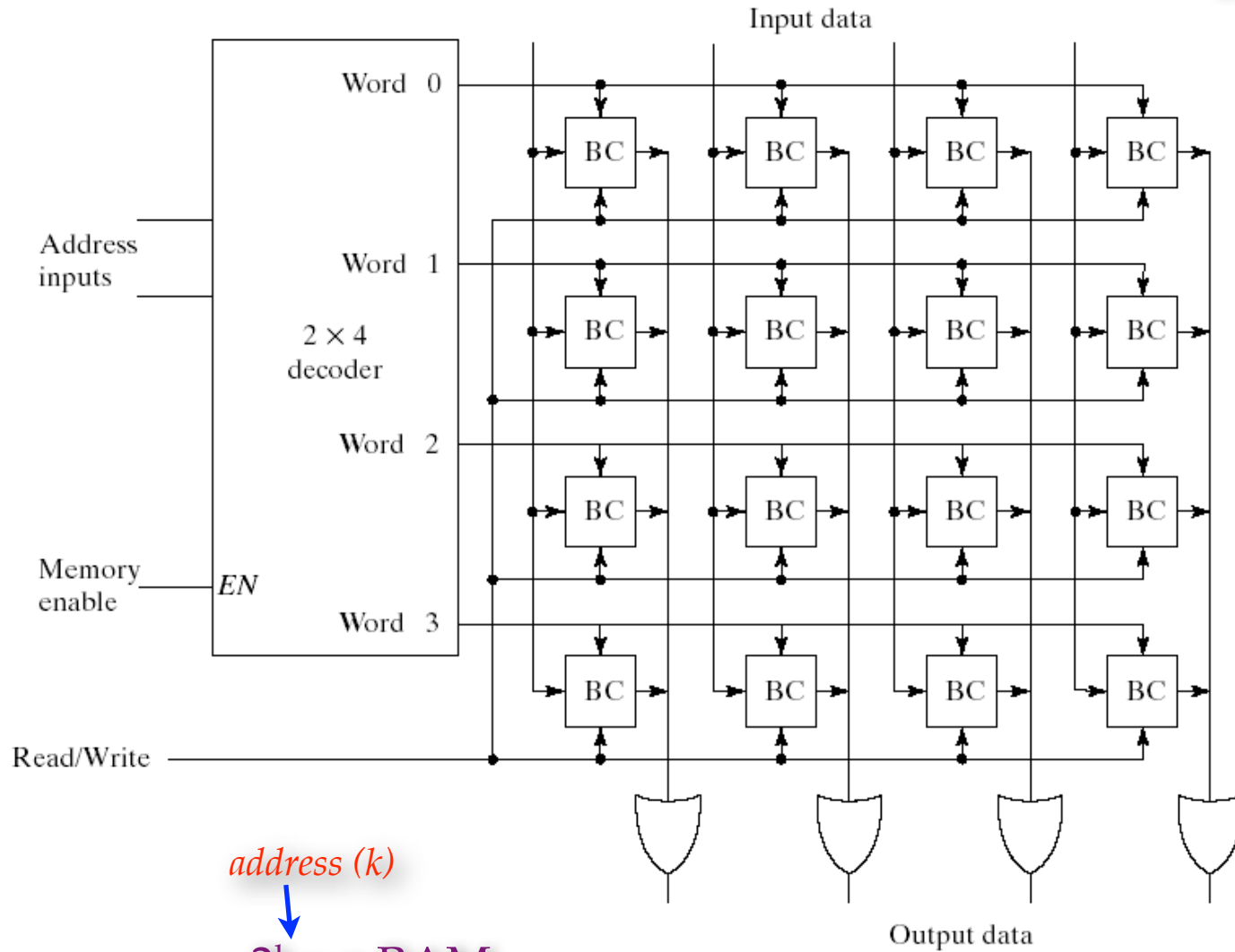
<http://lms.nthu.edu.tw/course/24953>

Department of Electrical Engineering

National Tsing Hua University

# RAM Revisit

$2^2 \times 4$  RAM



*address (k)*

$2^k \times n$  RAM

*# of output bits (n)*

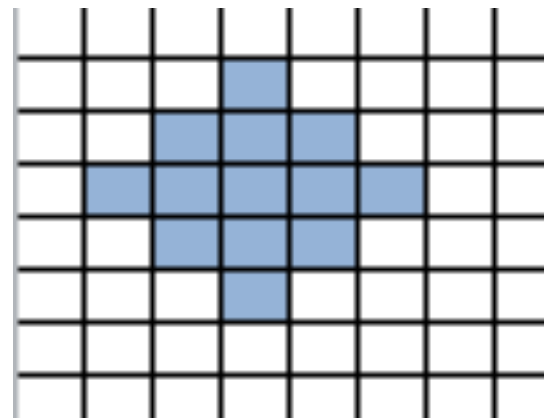
# COE Format

- COE: memory coefficient file
- Two parameter:
  - **memory\_initialization\_radix**
    - Radix of the values in the *memory\_initialization\_vector*
    - Ex: 2, 10, or 16
  - **memory\_initialization\_vector:**
    - Memory content
    - Memory words are separated by **whitespace**
    - You can use comma (,) to help identify the boundary
    - Vector (entire memory) ended by **semicolon**

# COE Example

```

; 8-bitwide by 8-deep RAM
memory_initialization_radix=2;
memory_initialization_vector=
00000000 , ← whitespace
00010000 ,
00111000 ,
01111100 ,
00111000 ,
00010000 ,
00000000 ,
00000000 ;
  
```



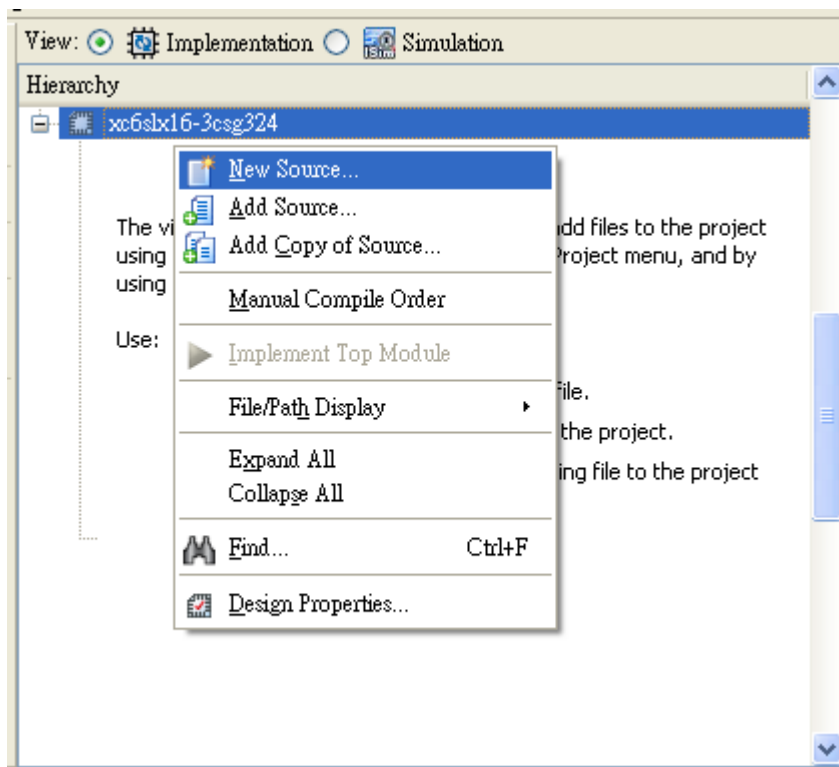
You can use ASCII art generator to generate the pictures or use drawing tool to export the figures for you.

# Use RAM

- You can use RAM for changeable LCD display of your project
- Similar as ROM
  - The same in IP generator, except choose 'Single Port RAM'
  - Now have write
- Timing
  - Write control, address, data should be at the same clock cycle
  - Data read out from RAM is one clock cycle late than the address control

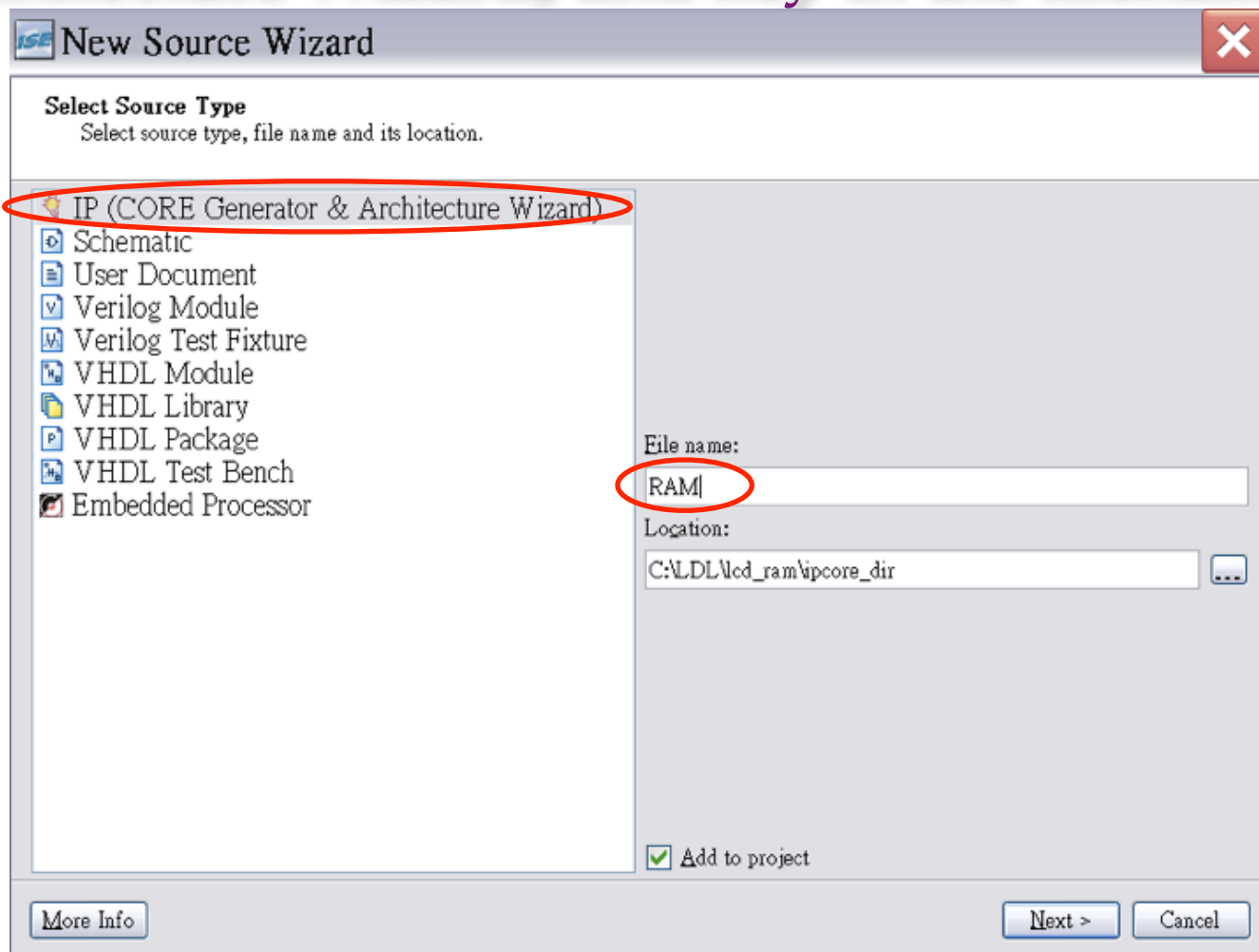
# Generate RAM (1/6)

- New Source



# Generate RAM (2/6)

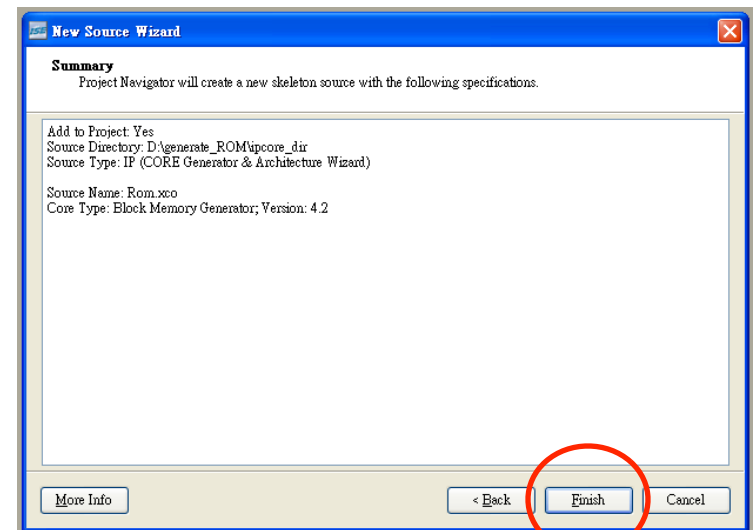
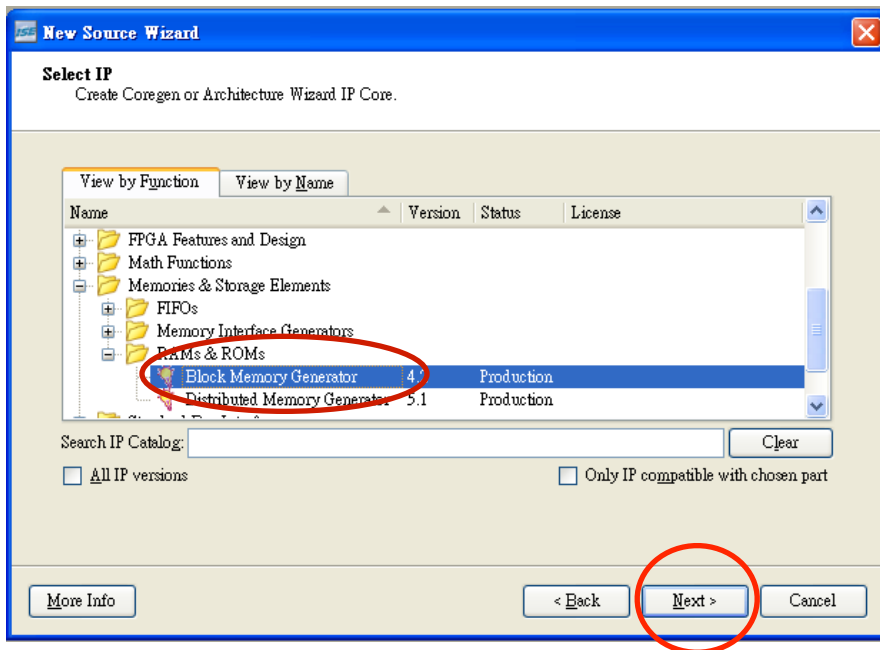
- Choose the source type: IP (CORE Generator & Architecture Wizard) and key in the filename



# Generate RAM (3/6)

- Select:

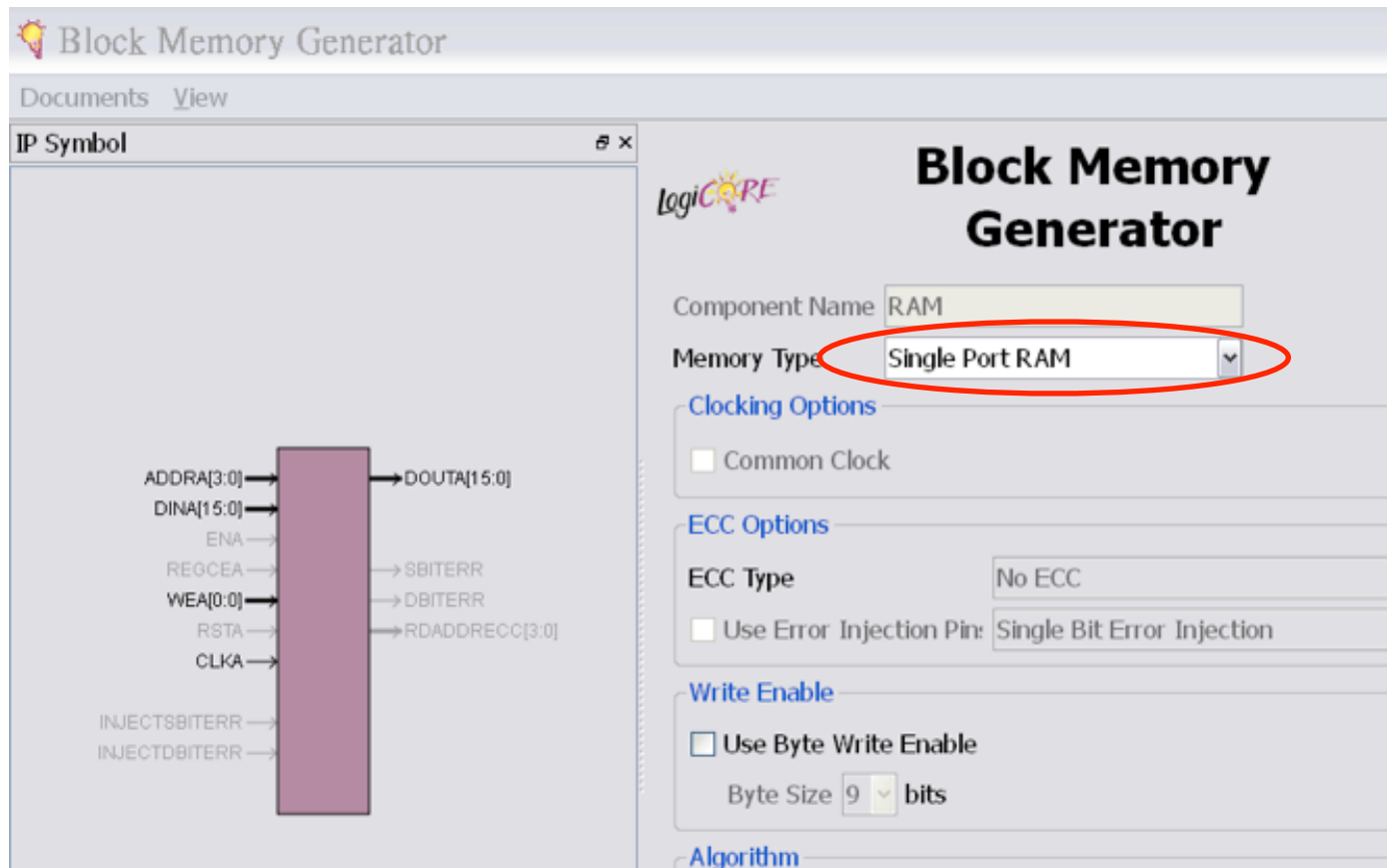
- Memories & Storage Elements -> RAMs/ROMs -> Block Memory Generator





# Generate RAM (4/6)

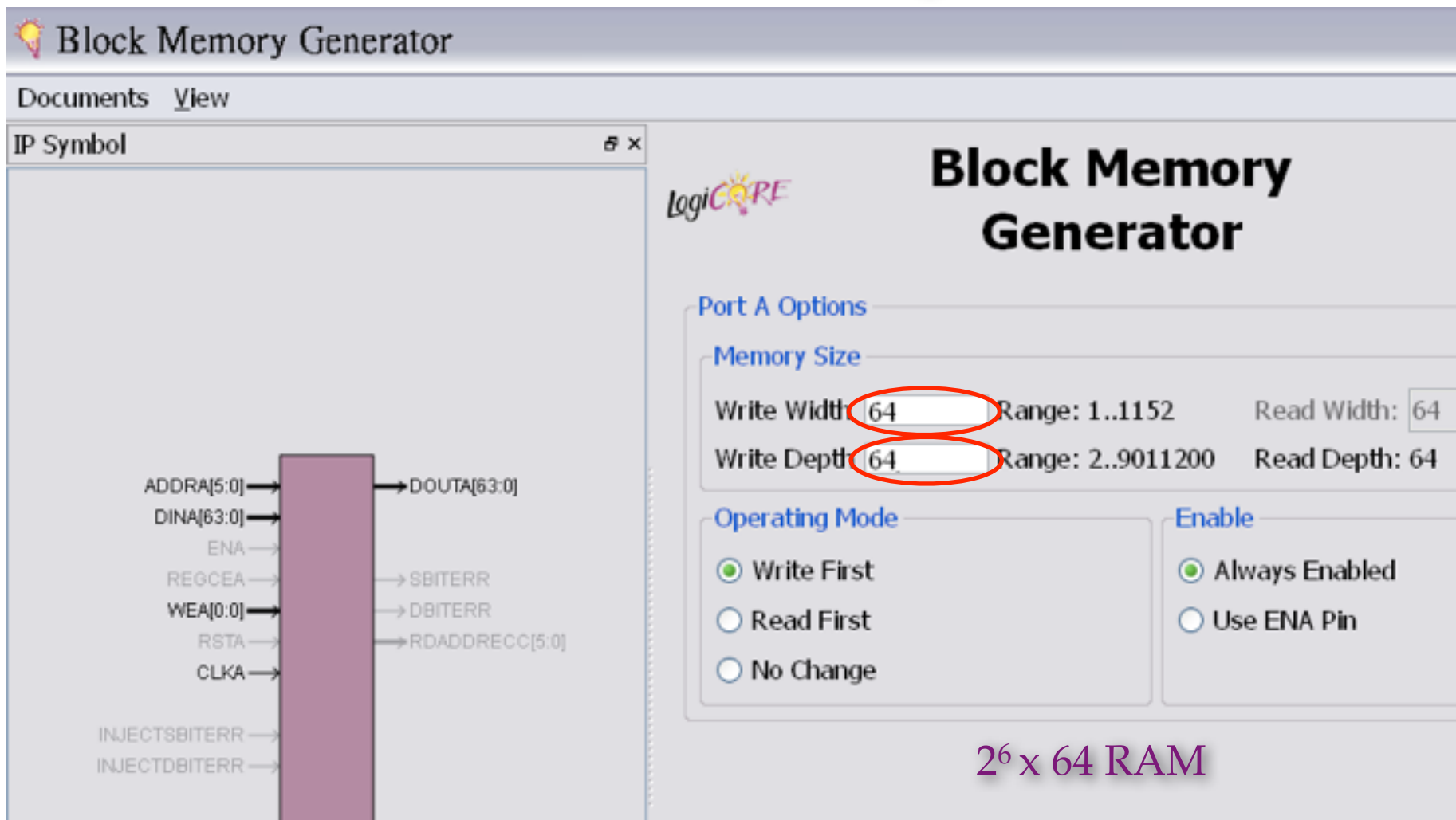
- Wait for a while
- Select Memory Type:  
Single Port RAM



The screenshot shows the 'Block Memory Generator' application window. The title bar reads 'Block Memory Generator'. Below the title bar, there is a 'Documents View' section and an 'IP Symbol' window. The main interface is divided into two panes. The left pane shows a block diagram of the RAM component with various input and output ports labeled: ADDR<sub>A</sub>[3:0], DIN<sub>A</sub>[15:0], EN<sub>A</sub>, REG<sub>CEA</sub>, WE<sub>A</sub>[0:0], R<sub>STA</sub>, CL<sub>KA</sub>, DOUT<sub>A</sub>[15:0], SBITERR, DBITERR, RDADDRECC[3:0], INJECTSBITERR, and INJECTDBITERR. The right pane contains the configuration options for the RAM. The 'Component Name' is set to 'RAM'. The 'Memory Type' dropdown menu is highlighted with a red circle and is set to 'Single Port RAM'. Below this, there are sections for 'Clocking Options' (with a 'Common Clock' checkbox), 'ECC Options' (with 'ECC Type' set to 'No ECC' and 'Use Error Injection Pin' set to 'Single Bit Error Injection'), 'Write Enable' (with 'Use Byte Write Enable' checkbox and 'Byte Size' set to '9 bits'), and 'Algorithm'.

# Generate RAM (5/6)

- Data width: 64 bits, address depth: 64 (1 frame)



Block Memory Generator

Documents View

IP Symbol 5 x

LogiCORE

## Block Memory Generator

Port A Options

Memory Size

Write Width: 64 Range: 1..1152 Read Width: 64

Write Depth: 64 Range: 2..9011200 Read Depth: 64

Operating Mode

Write First

Read First

No Change

Enable

Always Enabled

Use ENA Pin

$2^6 \times 64$  RAM

Block Diagram:

ADDRA[5:0] → DOUTA[63:0]

DINA[63:0] →

ENA →

REGCEA → SBITERR

WEA[0:0] → DBITERR

RSTA → RDADDRECC[5:0]

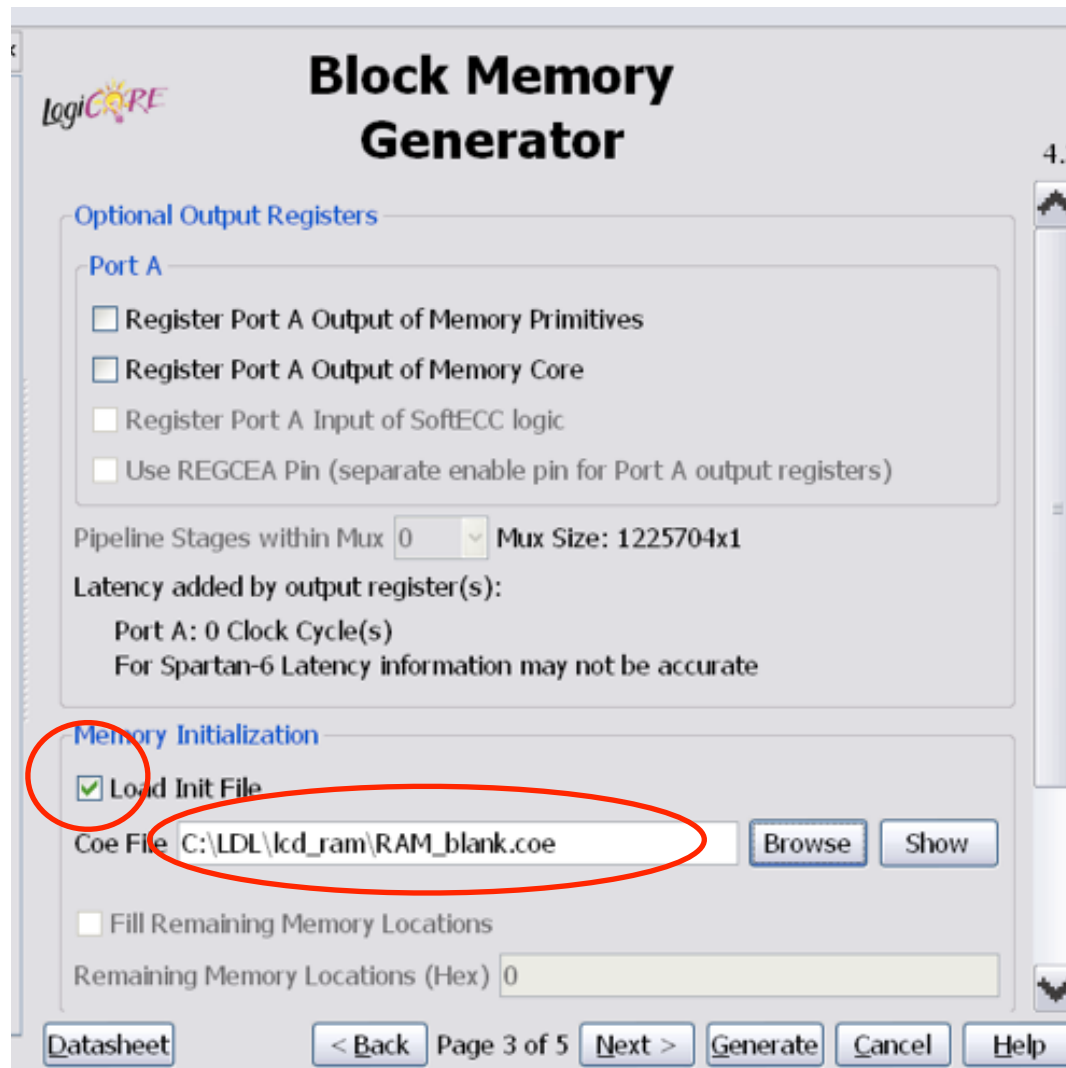
CLKA →

INJECTSBITERR →

INJECTDBITERR →

# Generate RAM (6/6)

- Check “Load Init File”
- Select “Browse” and load your COE file



# How to Use RAM Module

- You can find the port names through the functional model

The screenshot displays the ISE Project Navigator interface. The Hierarchy window on the left shows a project structure with 'RAM (RAM.xco)' highlighted. The Processes window below it shows 'View HDL Functional Model' selected. The main editor window shows a Verilog code snippet for RAM instantiation, with a red circle highlighting the instantiation block.

```
49 // core to be instantiated. Change the
50 // (in parentheses) to your own signal
51
52 //----- Begin Cut here for INSTAN
53 RAM YourInstanceName (
54     .clka(clka),
55     .wea(wea), // Bus [0 : 0]
56     .addra(addra), // Bus [5 : 0]
57     .dina(dina), // Bus [63 : 0]
58     .douta(douta)); // Bus [63 : 0]
59
60 // INST_TAG_END ----- End INSTANTIATIC
61
62 // You must compile the wrapper file RA
63 // the core, RAM. When compiling the wr
64 // reference the XilinxCoreLib Verilog
65 // instructions, please refer to the "C
66
67
```

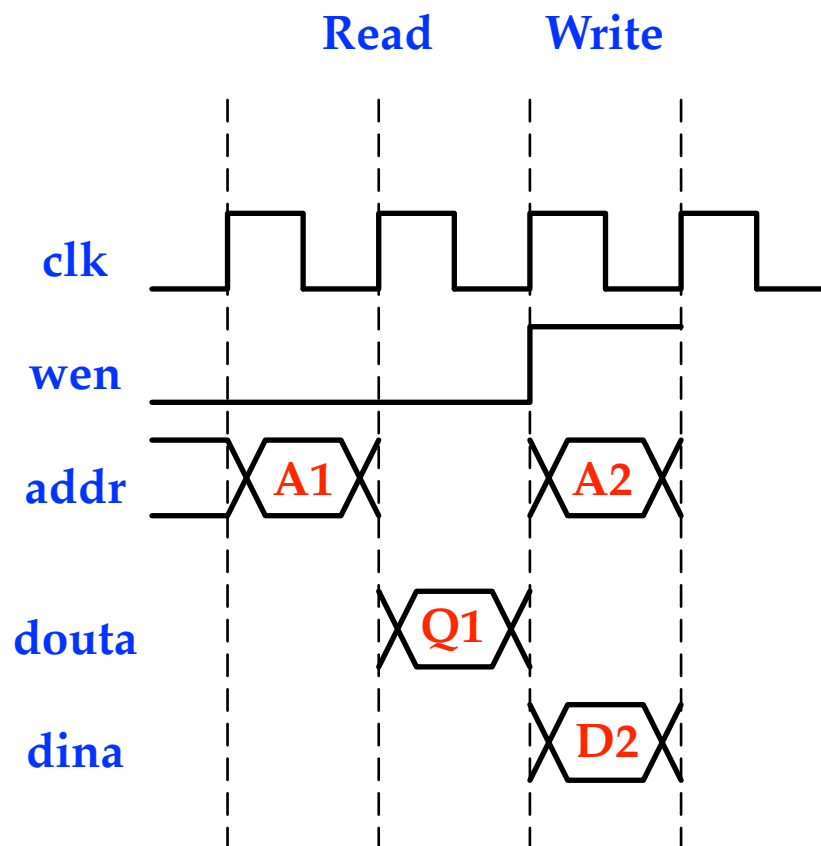
# How to Use RAM Module

```

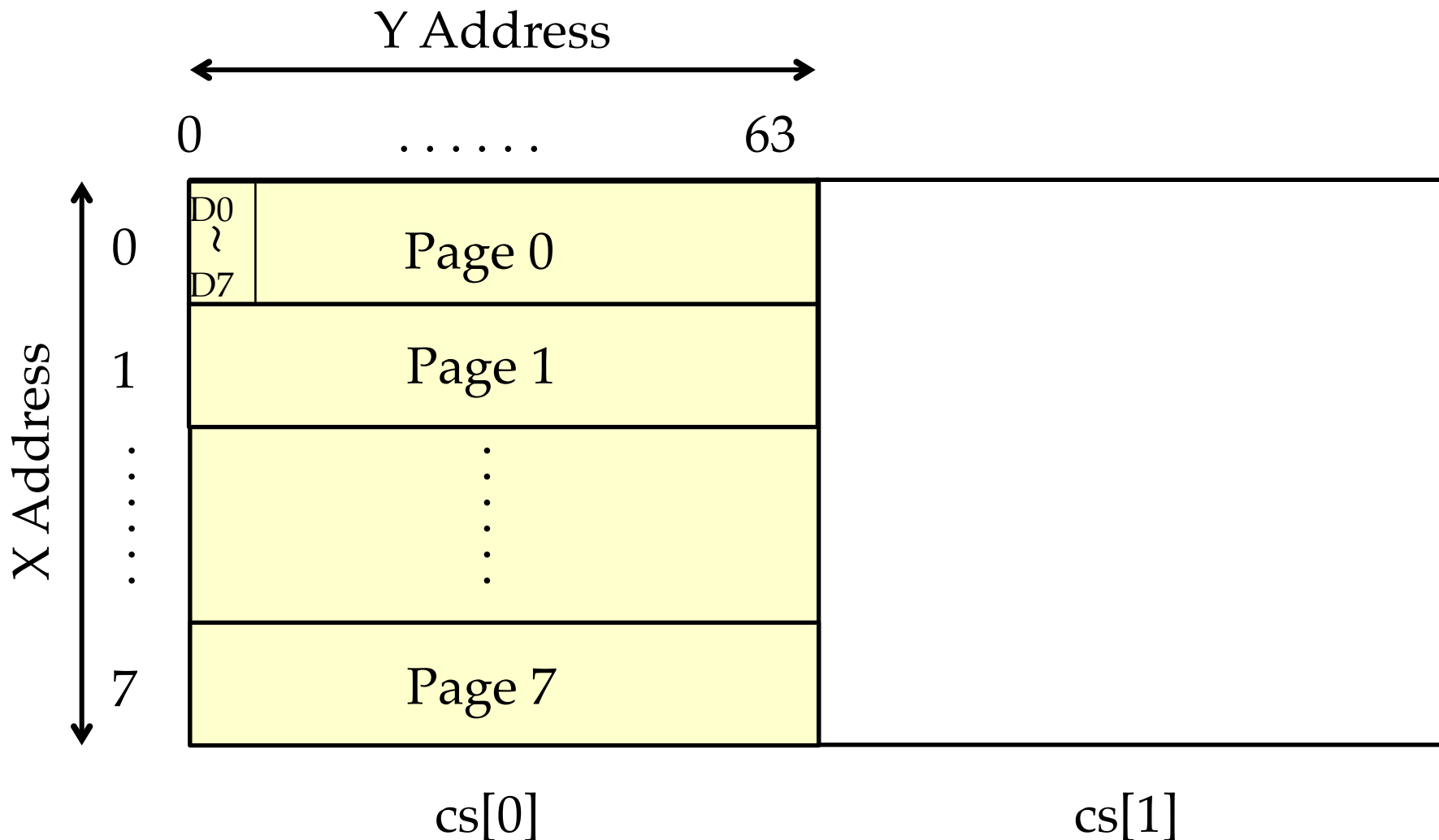
wire clk;
wire wen;
wire [63:0] data_in;
wire [63:0] out_64;
wire [5:0] addr;
  
```

```

RAM R1(
  .clka(clk),
  .wea(wen),
  .addra(addr),
  .dina(data_in),
  .douta(out_64)
);
  
```

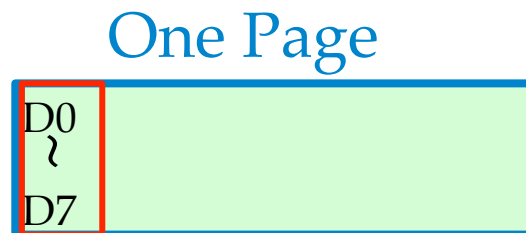
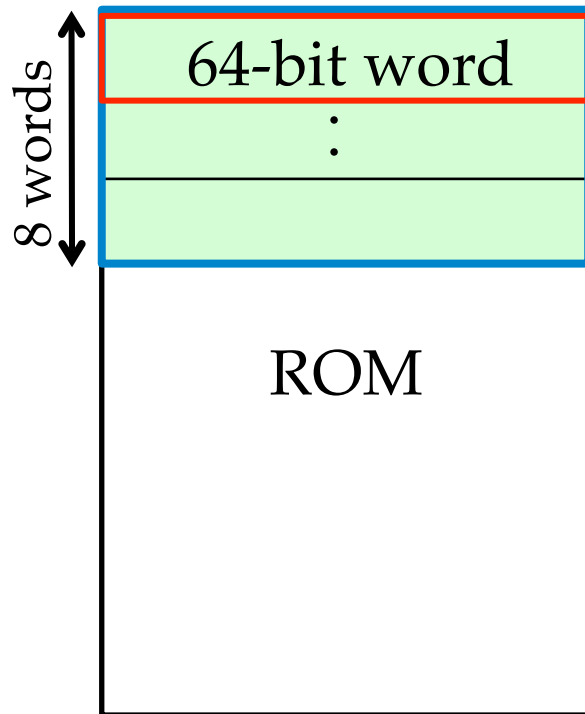


# LCD Display (128x64)

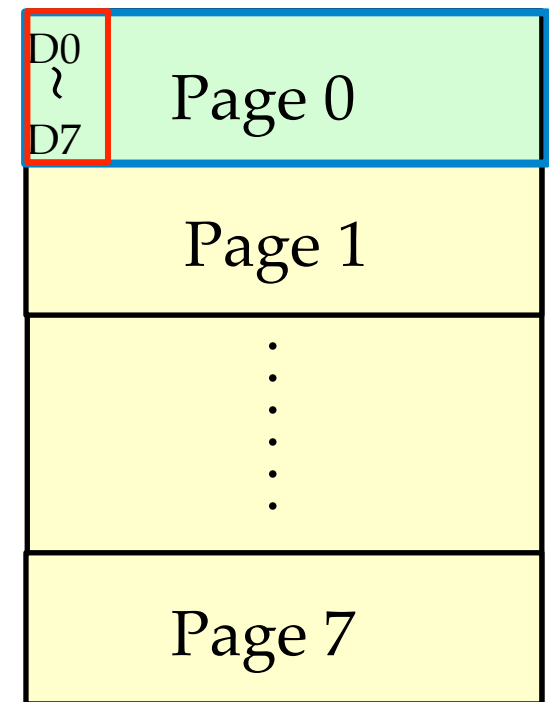


# Concept of a ROM Controller

- Fetch a page one time
- Data rearrangement (words to bytes)
  - 8x64-bit (8 *words*) to 64x8-bit (64 *bytes*)

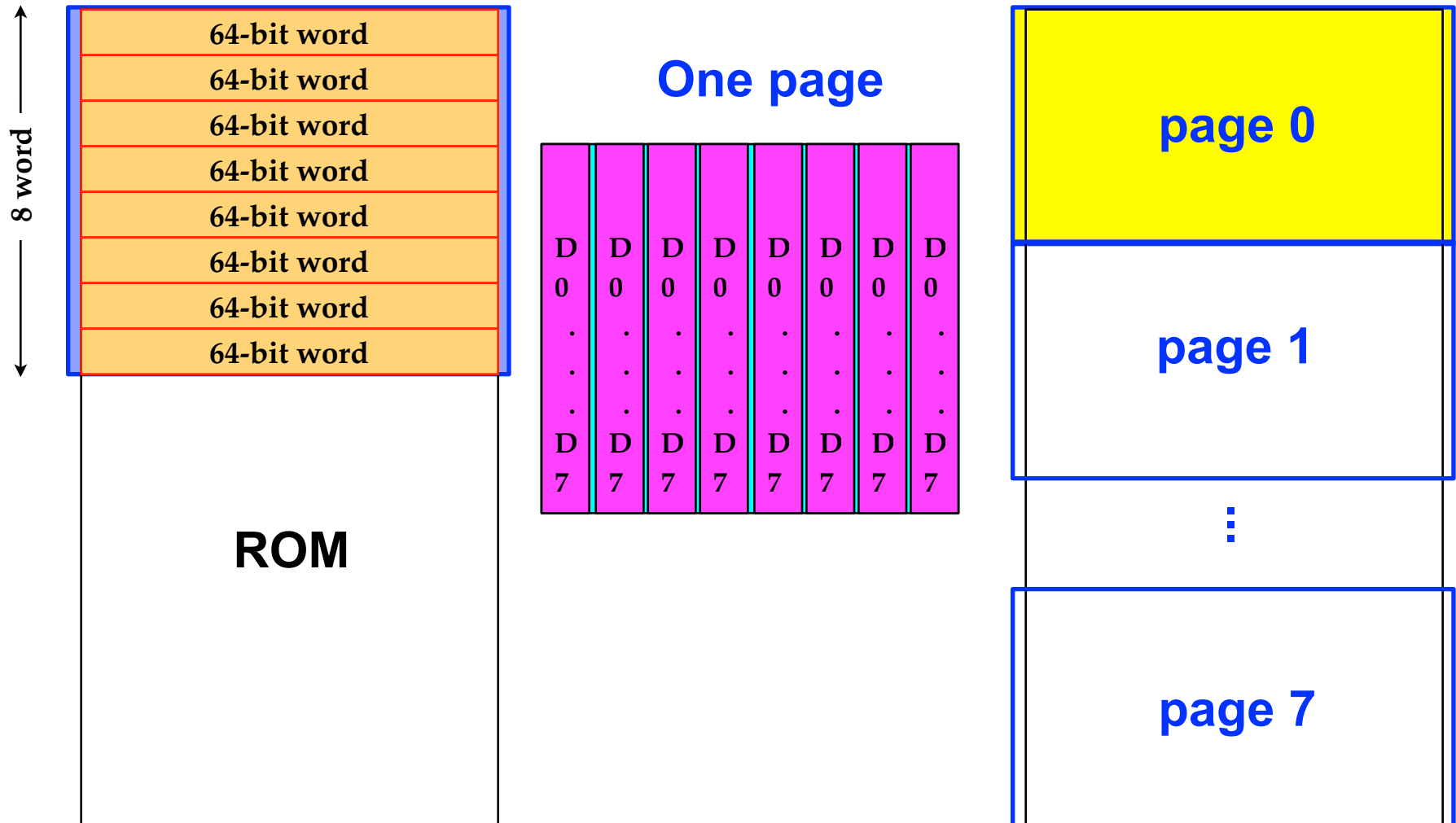


LCD Display



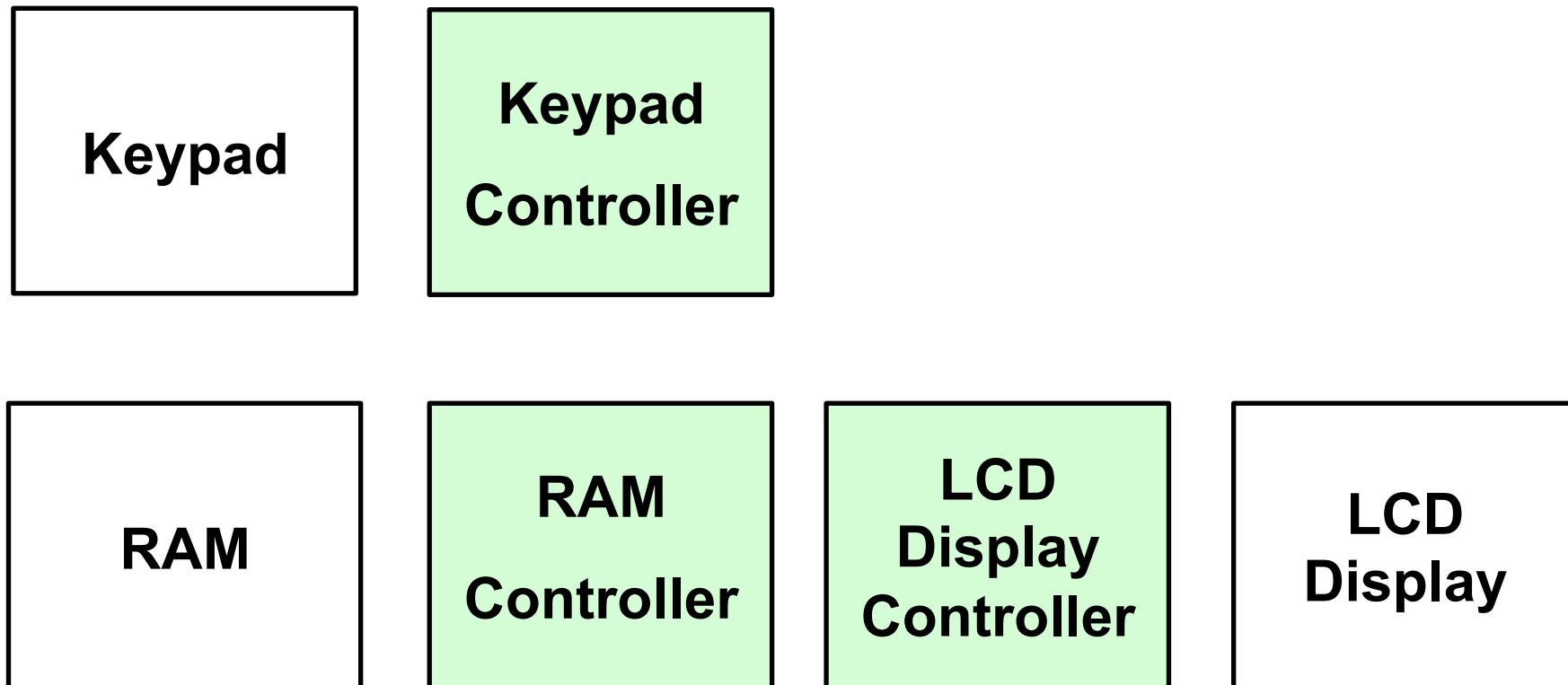
[cthuang]

# Concept of a ROM Controller



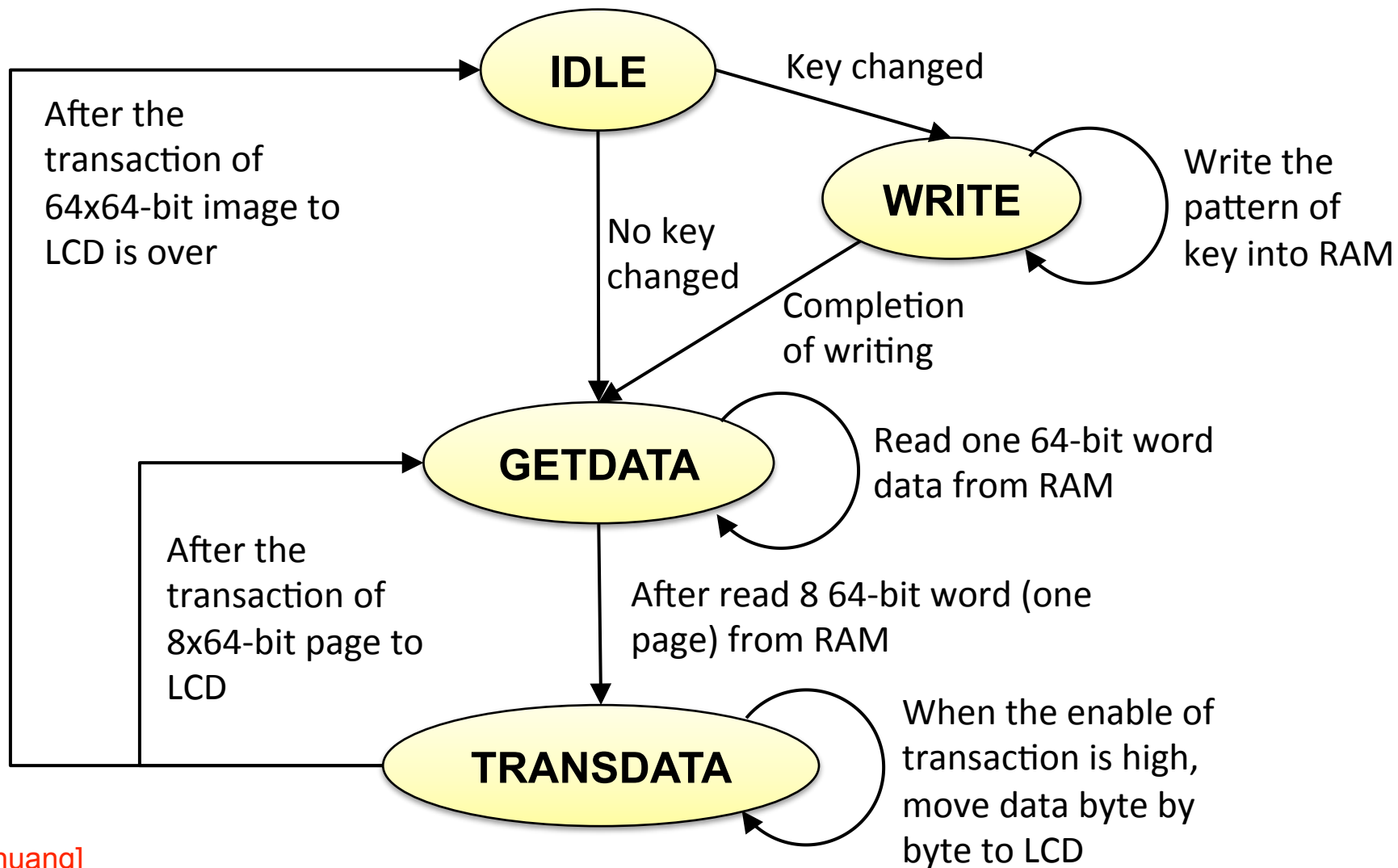


# LCD Display with Keypad



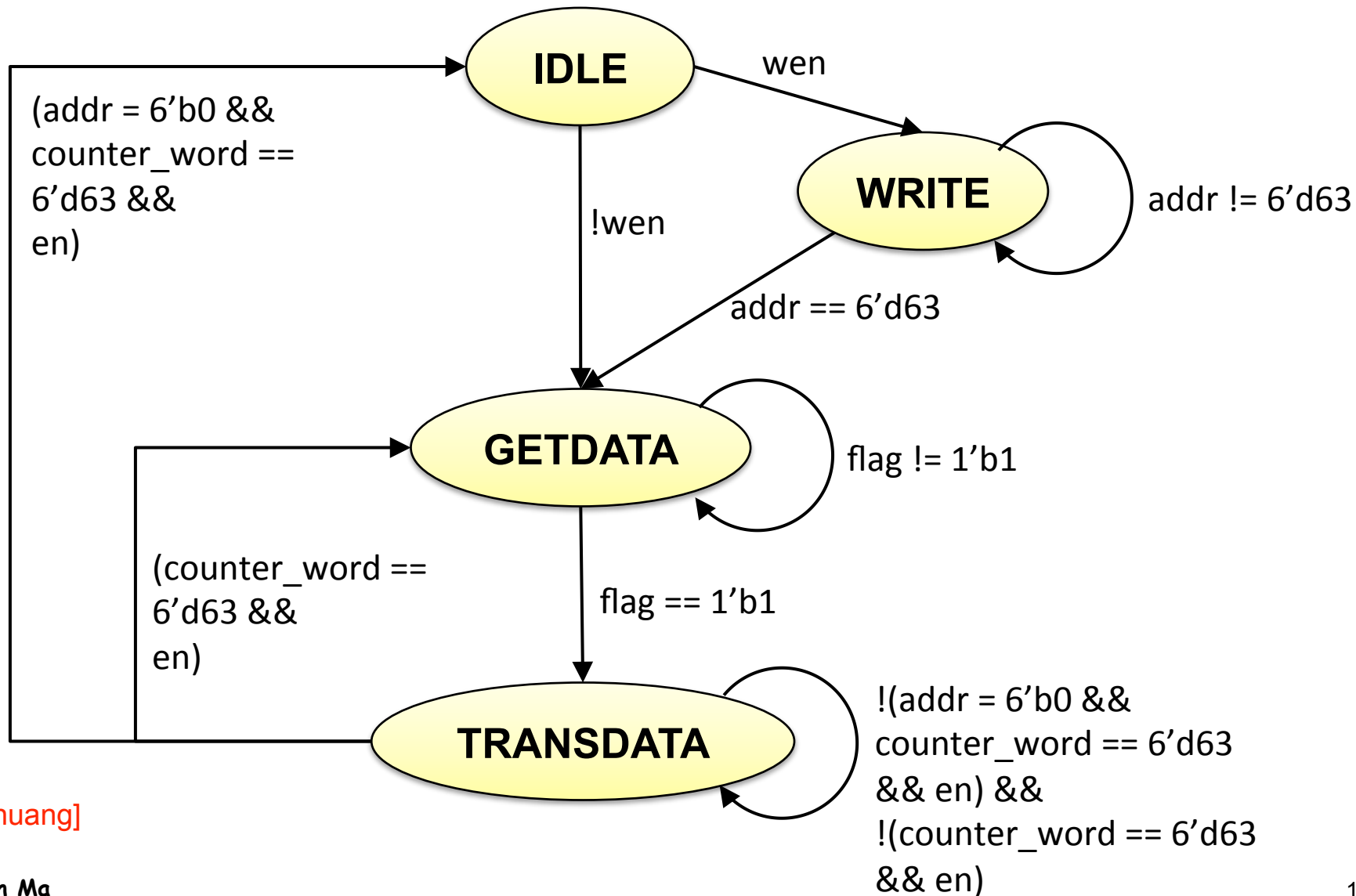
[cthuang]

# RAM Controller



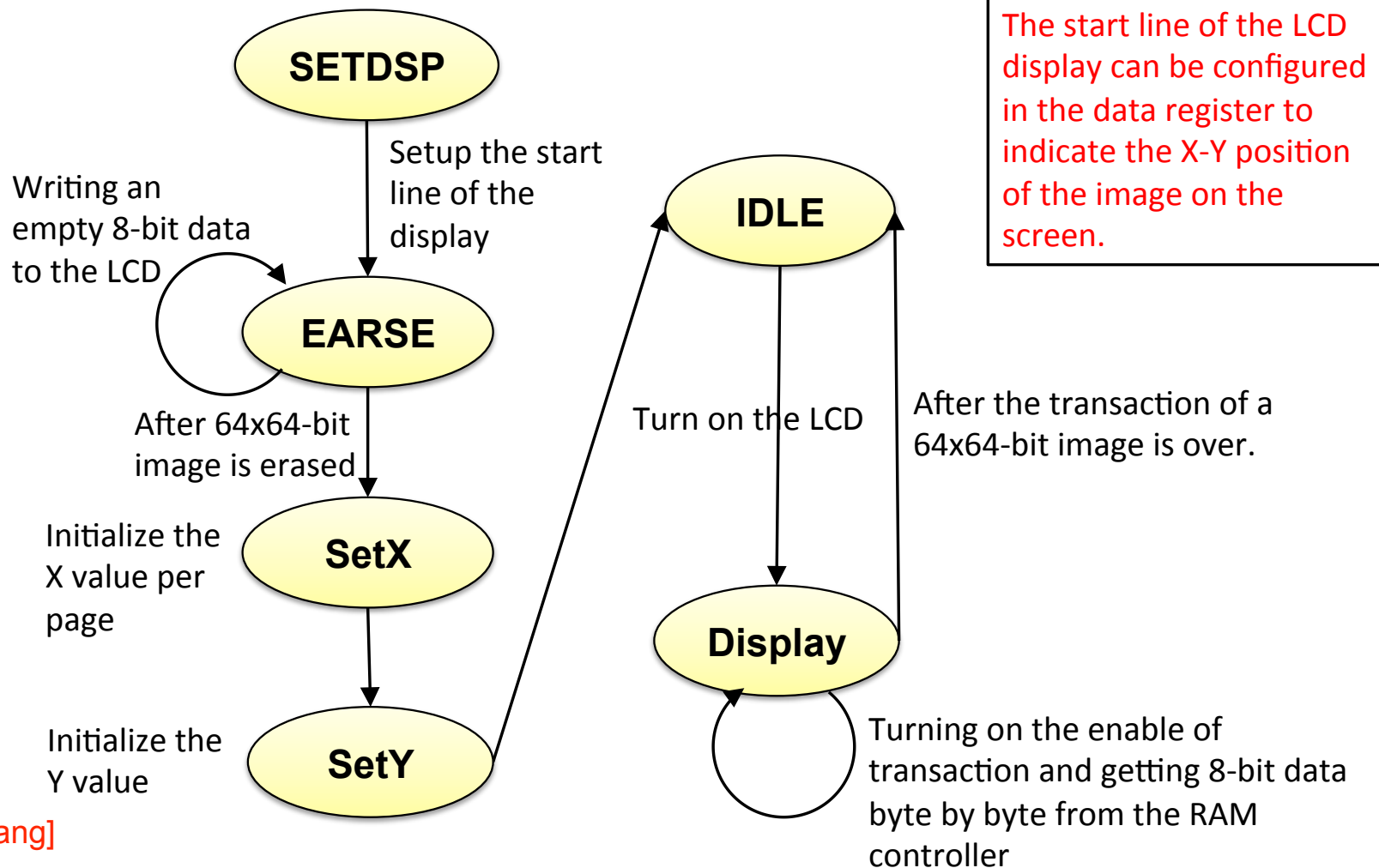
[cthuang]

# RAM Controller



[cthuang]

# LCD Display Controller



[cthuang]