# Logos2 Family FPGAs Configurable Logic Module (CLM) User Guide

(UG040001, V1.4)

(14.07.2023)

# Revisions History

## Document Revisions

| Version | Date of Release | Revisions |
|---|---|---|
| V1.4 | 14.07.2023 | Initial release. |
| | | |

# About this Manual

## Terms and Abbreviations

| Terms and Abbreviations | Meaning |
|---|---|
| CLM | Configurable Logic Module |
| LUT | Look Up Table |
| SRB | Signal Relay Block |
| GTP | Generic Technology Primitive |
| | |

# Table of Contents

# Tables

# Figures

# Chapter 1 General Introduction

CLM (Configurable Logic Module) is the basic logic unit of Logos2 family products. CLMs are distributed in columns within the Logos2 family products and support two forms: CLMA and CLMS. Both CLMA and CLMS can implement logic, arithmetic, shift registers, and ROM functions, but only CLMS supports distributed RAM functionality. The ratio of CLMA to CLMS is approximately 3:1. CLMs are interconnected with each other and with other on-chip resources through the Signal Relay Block (SRB), as shown in
Figure 1-1.

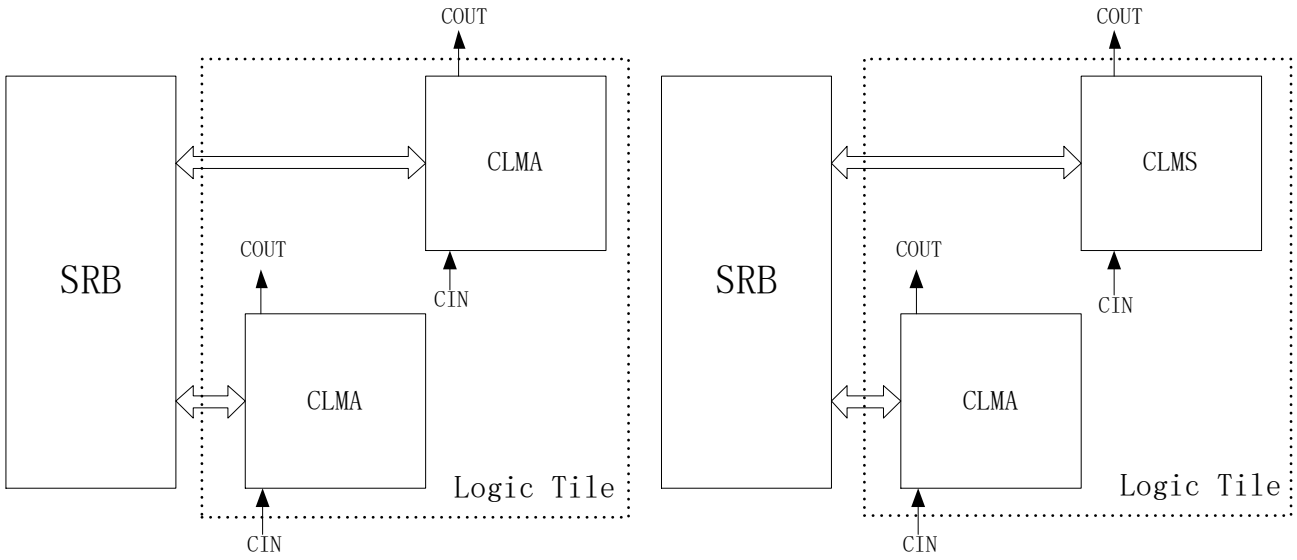


Figure 1-1 CLM Distribution Diagram

Main CLM features of Logos2 series FPGA products include:

- Using an innovative LUT6 logic architecture
- Each CLM contains 4 multifunctional LUT6s
- Each CLM contains 8 registers
- Arithmetic functional mode supported
- Fast arithmetic carry logic supported
- Efficiently implementing multiplexer functions
- Capable of implementing ROM functions
- Cascade chain for the shift register supported.
- CLMS supports distributed RAM mode

The CLMs in the Logos2 family products adopt the LUT6 structure and have an improved carry

chain structure. The number of flip-flops included in each CLM has been increased, and the internal logic has been optimized. While ensuring timing, it consumes less power and is generally more supportive for large-capacity designs.

Device capacity is typically measured by logic units composed of an LUT4 and a flip-flop. As mentioned above, the CLMs in the Logos2 family FPGA products use the innovative LUT6 logic architecture, which includes a wealth of flip-flops, latches, carry logic, as well as the ability to create distributed RAM/ROM on-chip, thereby increasing their effective capacity. The ratio of the number of logic units to LUT6 is 1.6:1. For the CLM resource scale of Logos2 family devices, please refer to ***"DS04001_Logos2 Family FPGAs Device Data Sheet"***.

# Chapter 2 Function Description

## 2.1 Introduction to CLM Structure

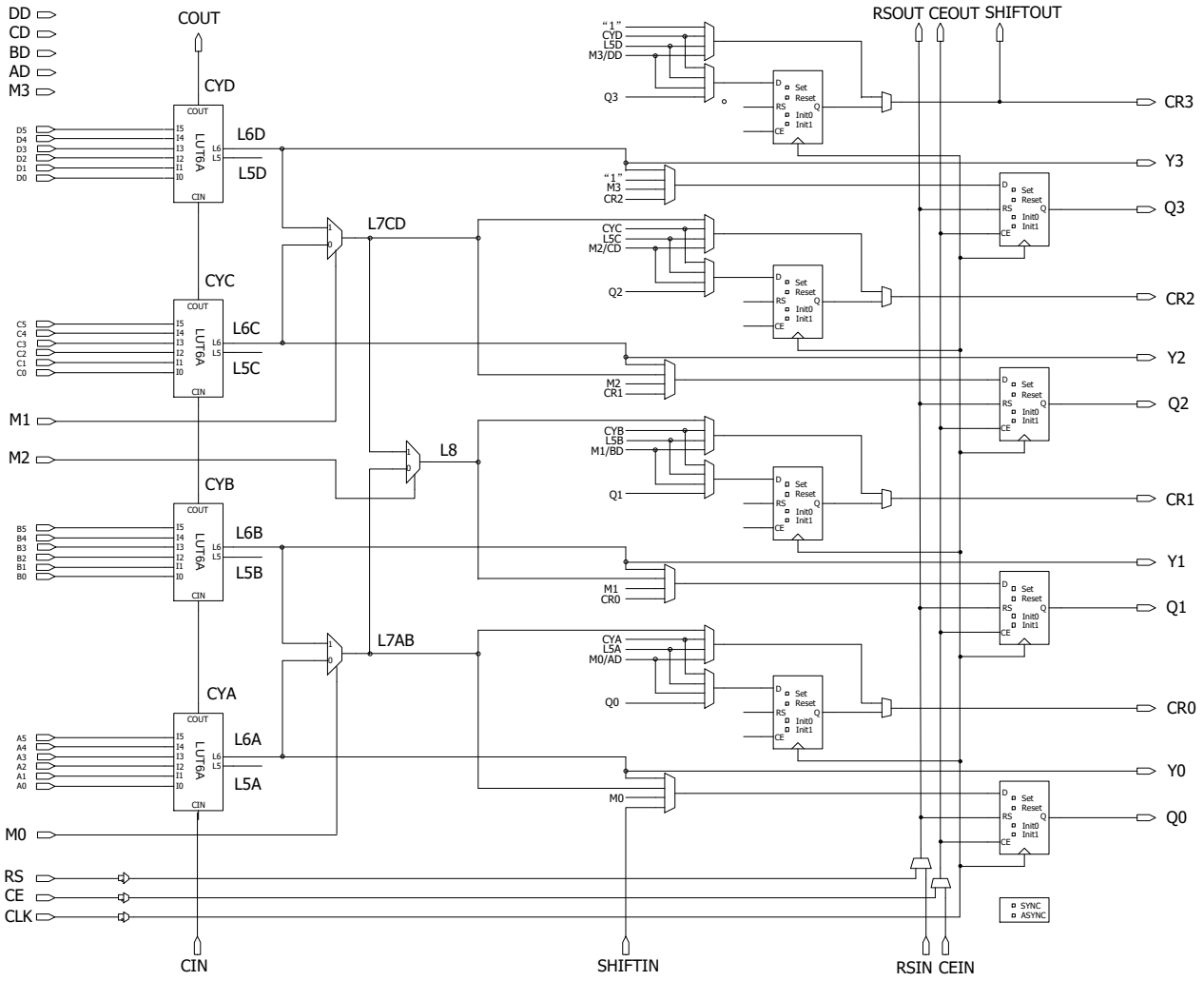


Figure 2-1 Logic Block Diagram of CLMA

The logic block diagram of CLMA is shown in

Figure 2-1. Each CLMA contains 4 LUT6s, 8 registers, multiple extension function selectors, and four independent cascade chains. (For clarity, hereinafter LUT6A will be used as a collective term for the four LUT6s in CLMA, with the LUT6A units having inputs A0~A5, B0~B5, C0~C5, and D0~D5 referred to as LUT6A_A, LUT6A_B, LUT6A_C, and LUT6A_D respectively.) LUT6A features an innovative architectural design that integrates dedicated circuits on top of a 6-input lookup table to achieve a 4:1 multiplexer function and fast arithmetic carry logic; the extension function selector is mainly used to implement wide-bit lookup tables and output selection functions; the cascade chains include arithmetic logic carry chains (from CIN to COUT), dedicated shift register chains (from

SHIFTIN to SHIFTOUT), register reset/set cascade chains (from RSIN to RSOUT), and register CE cascade chains (from CEIN to CEOUT). A0~A5/B0~B5/C0~C5/D0~D5 are the six input ports of the four LUT6s respectively, M0~M2 are the control terminals of the extended LUT7/LUT8 multiplexer, RS is the register reset/set signal, CE is the clock enable, CLK is the clock input, Y0~Y3 are the data outputs of the four LUT6s, CR0~CR3 are the output selection ports for LUT7/8 and CR registers, and Q0~Q3 are the output selection ports for LUT6 and registers (different from CR registers).

CLMS is an extension of CLMA, which not only supports all functionalities of CLMA but also supports distributed RAM. The multifunctional LUT6 in CLMS is referred to as LUT6S. (For clarity, hereinafter LUT6S will be used as a collective term for the four LUT6s in CLMS, with inputs A0~A5, B0~B5, C0~C5, D0~D5 referred to as LUT6S_A, LUT6S_B, LUT6S_C, and LUT6S_D respectively). CLMS can be configured as either a 64*4 SP (Single Port) RAM or a 64*3 SDP (Simple Dual Port) RAM. The logic block diagram of CLMS is shown in

Figure 2-2, where AD~DD and M0~M3 are the write data inputs for the distributed RAM mode of the four LUT6s, and WE is the write enable for the distributed RAM mode.

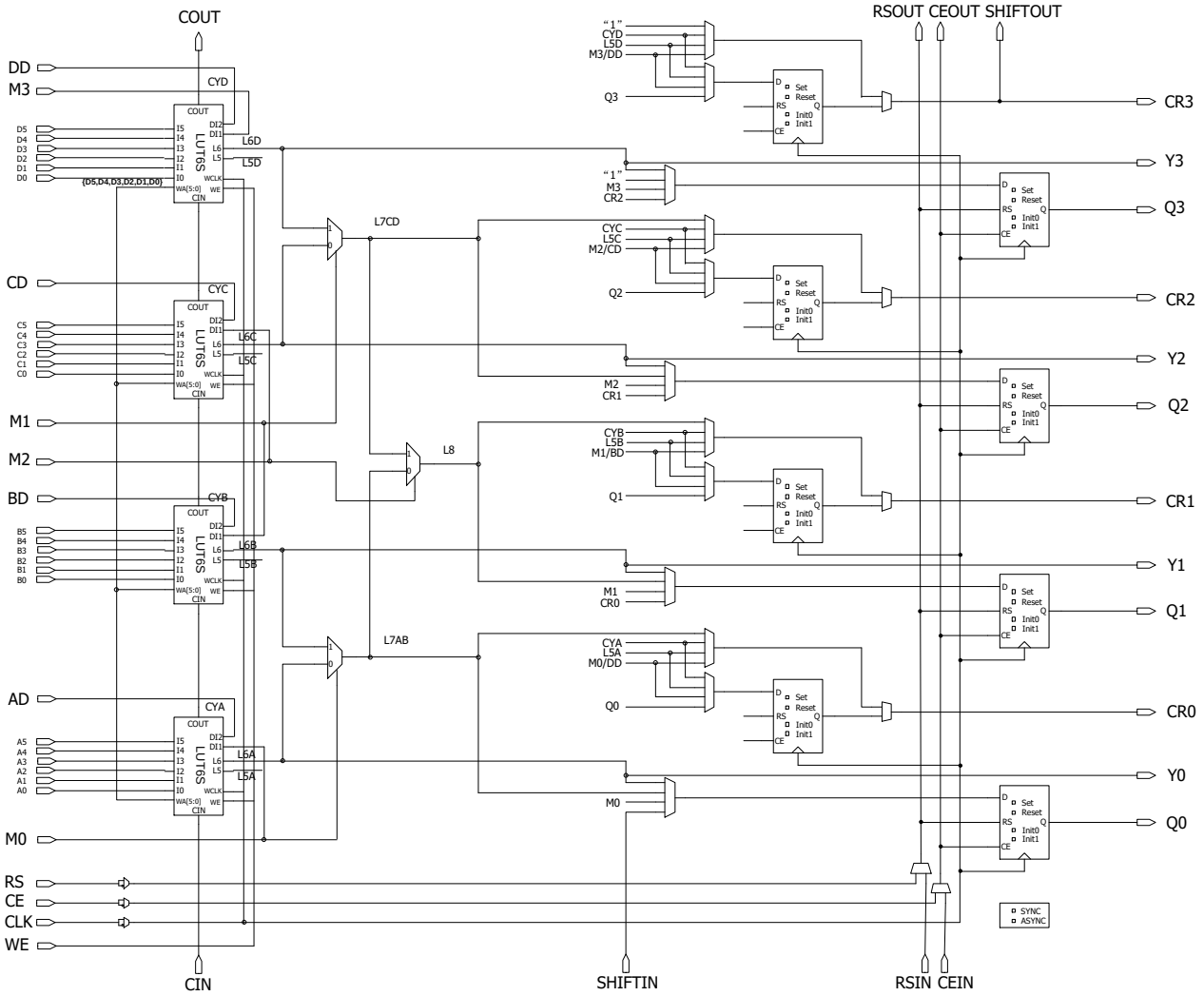


Figure 2-2 Logic Block Diagram of CLMS

Some internal resources can be shared by CLMA and CLMS. While using shared resources, if some shared resources such as RS, CE, and CLK are occupied, then the users can not use the remaining resources independently. For example, if a LUT6 and a CE clock enable signal in the CLMA are used, the remaining LUT6s cannot be controlled with an independent CE clock enable signal.

In

Figure 2-1 and

Figure 2-2, routing resources are multiplexed to the input of additional registers (left column registers) via M0/M1/M2/M3 or AD/BD/CD/DD ports. Among them, devices such as PG2L100H multiplex M0/M1/M2/M3 ports, while devices such as PG2L50H multiplex AD/BD/CD/DD, as shown in the table below. Additionally, the CE signal structure of PG2L200H differs from the aforementioned diagram, please refer to Figure 2-17.

Table 2-1 Input of Routing Resources to Additional Register

| **Device** | **Multiplexed Ports** |
|---|---|
| PG2L100H/PG2L200H | M0/M1/M2/M3 |
| PG2L25H/PG2L50H | AD/BD/CD/DD |

Note: The AD port multiplexed by CLMA is left floating in the hardware due to the limited output of routing resources.

## 2.2 Operating Modes of LUT6A and LUT6S

LUT6A and LUT6S can be flexibly configured to support various functions such as basic logic, multiplexing, arithmetic logic, ROM functions, and distributed RAM functions (limited to LUT6S).

### 2.2.1 Logic Functional Mode

In logic function mode, each LUT6A (or LUT6S) can implement one LUT6, and combined with the extension function selector, each CLM can support four LUT6s, two LUT7s, or one LUT8.

### 2.2.2 Multiplexer Mode

In multiplexer mode, each LUT6A (or LUT6S) enables one 4:1 multiplexer, and each CLM can support four 4:1 multiplexers. Combined with the extension function selector, every two LUT6As (or LUT6Ss) can support one 8:1 multiplexer, and each CLM can support two 8:1 multiplexers; every four LUT6As (or LUT6Ss) enable one 16:1 multiplexer, and each CLM supports one 16:1 multiplexer. Wider multiplexed data selection can be generated through the combination of CLMs.

### 2.2.3 Arithmetic Functional Mode

In arithmetic function mode, LUT6A (or LUT6S) can implement addition and subtraction operations, counters, comparators, fast XOR logic operations, AND logical operation with large bit width, etc. Taking an adder as an example, to implement a full adder, its truth table is shown in Table 2-2.

Table 2-2 Truth Table for 1-bit Full Adder

| A | B | CIN | COUT | SUM |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 | 1 |
| 0 | 1 | 0 | 0 | 1 |
| 0 | 1 | 1 | 1 | 0 |
| 1 | 0 | 0 | 0 | 1 |
| 1 | 0 | 1 | 1 | 0 |
| 1 | 1 | 0 | 1 | 0 |
| 1 | 1 | 1 | 1 | 1 |

The logical expression is:

COUT=A⊕B*CIN+!(A⊕B)*B

SUM=A⊕B⊕CIN =A⊕B*!CIN+!(A⊕B)*CIN

The structure of CLM implements the full adder through the hardware circuit shown in Figure 2-3.

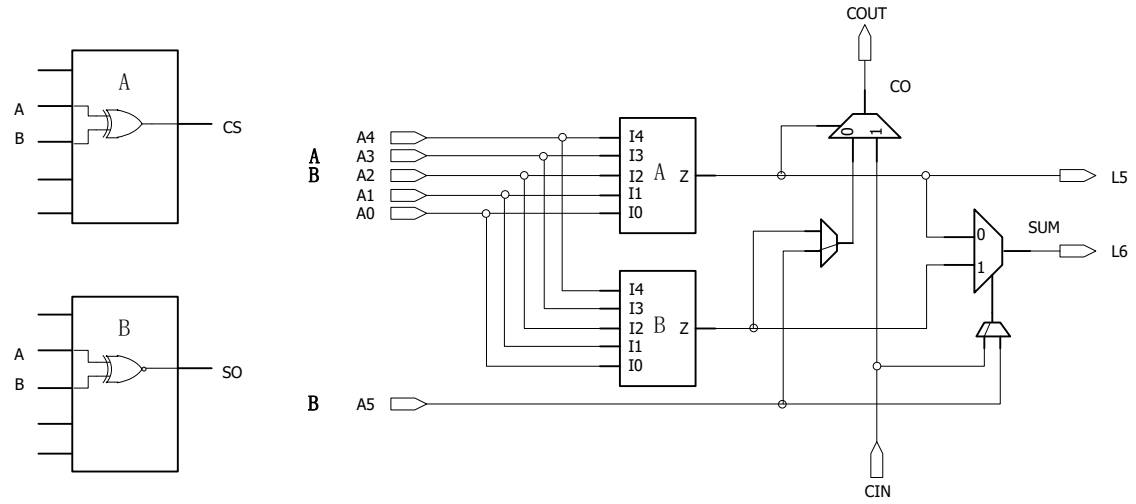


Figure 2-3 Diagram of Adder Implementation

LUT5_A is configured as the A⊕B function, serving as the selection port of the carry chain mux. When it is at high level, CIN is selected, and when it is at low level, the B input is selected. Its output serves as the carry output. The SUM output of the full adder uses CIN as the selection port for LUT6 MUX. CIN selects LUT5_B when it is at high level and LUT5_A when it is at low level. LUT5_B is configured as the A⊙B function.

Through the above method, the adder function is realized.

### 2.2.4 ROM Mode

In ROM mode, LUT6A (or LUT6S) can be used as a 64*1 ROM, and can be depth-cascaded using the built-in extension function selector. The initialization of ROM data is completed during the programming configuration process.

All supported ROM modes are shown in Table 2-3.

Table 2-3 ROM Modes Supported by CLM

| **ROM Mode** | **GTP** | **Required Number of LUTs** |
|---|---|---|
| 32×1 | GTP_ROM32X1 | 1 |
| 32×2 | GTP_ROM32X2 | 1 |
| 64×1 | GTP_ROM64X1 | 1 |
| 128×1 | GTP_ROM128X1 | 2 |
| 256×1 | GTP_ROM256X1 | 4 |

### 2.2.5 Distributed RAM Mode

Distributed RAM provides a compromise scheme of using storage elements in very small arrays and using DRM in larger arrays. Users can flexibly use distributed RAM and DRM based on the size of the memory. Distributed RAM can be generated using the IP Compiler tool embedded in the Pango Design Suite or through instantiation of GTP primitives.

In general, distributed RAM is recommended for all memories with a depth of 64 bits or less, unless there is a shortage of CLMS or logic resources for the target device. Distributed RAM is more efficient in terms of resources, performance, and power. For depths greater than 64 bits but less than or equal to 128 bits, the decision to use distributed RAM or DRM depends on these factors:

- Availability: If DRM resources are insufficient, distributed RAM can be used.
- Latency requirements: If asynchronous read capability is needed, distributed RAM must be used.
- Data width: For widths greater than or equal to 16 bits, DRM should be used if available.
- Necessary performance requirements: Distributed RAMs generally have shorter clock-to-out timing and fewer placement restrictions than DRMs.

This mode only involves CLMS. A CLMS can be configured into RAM blocks of various sizes as needed, with the RAM's address and bit width adjusted according to the configuration. The modes listed in the Table 2-4 are the RAM modes supported by Logos2.

Table 2-4 RAM Modes Supported by CLMS

| RAM Mode | GTP | Description |
|---|---|---|
| RAM64X1SP | GTP_RAM64X1SP | A LUT6 can be configured as a 64-bit single-port RAM |
| RAM32X1SP | GTP_RAM32X1SP | A LUT6 can be configured as two single-port RAMs with 32-bit shared addresses |
| RAM64X1DP | GTP_RAM64X1DP | A LUT6 can be configured as a 64-bit simple dual-port RAM |
| RAM32X1DP | GTP_RAM32X1DP | A LUT6 can be configured as two simple dual-port RAMs with shared 32-bit addresses |
| RAM128X1SP | GTP_RAM128X1SP | Two LUT6s are configured as a128-bit single-port RAM, with combinations of LUT6S_A and LUT6S_B, LUT6S_C or LUT6S_D |
| RAM128X1DP | GTP_RAM128X1DP | Two LUT6s are configured as a 128-bit simple dual-port RAM, with combination of LUT6S_A and LUT6S_B |
| RAM256X1SP | GTP_RAM256X1SP | Four LUT6s can be configured as a 256-bit single-port RAM, with a combination of LUT6S_A, LUT6S_B, LUT6S_C and LUT6S_D |
| RAM32X2SP | GTP_RAM32X2SP | A LUT6 can be configured as a 2x32-bit single-port RAM |
| RAM32X2DP | GTP_RAM32X2DP | A LUT6 can be configured as a 2x32-bit simple dual-port RAM |
| RAM32X2X4 | GTP_RAM32X2X4 | A CLMS can be configured as four configurable RAMs with shared 2x32-bit write addresses |
| RAM64X1X4 | GTP_RAM64X1X4 | A CLMS can be configured as four configurable RAMs with shared 64-bit write addresses |

Single Port (SP) means that the write and read addresses are shared by a single port. Simple Dual Port (SDP) means that one port is used for synchronous write operations, while the other port is used for asynchronous read operations. In distributed RAM mode, the input of LUT6S_D is multiplexed as the write address, while the read address utilizes the input of LUT6S itself. When using simple dual-port mode, users need to prevent read-write conflicts, ensuring that the same address is not accessed for both reading and writing at the same time.

#### 2.2.5.1 RAM32X1SP

Taking LUT6S_A as an example to implement a RAM32X1SP, the input D[4:0] of LUT6S_D is multiplexed as the write address, and the read address is the input A[4:0] of LUT6S_A itself. Since it is a single-port type SRAM, A[4:0] needs to be connected to D[4:0]; the write data port is either the AD or M0 input, the read data output port is Y0, and the write enable is either WE or CE. One LUT6S can instantiate two RAM32X1SPs with shared addresses.

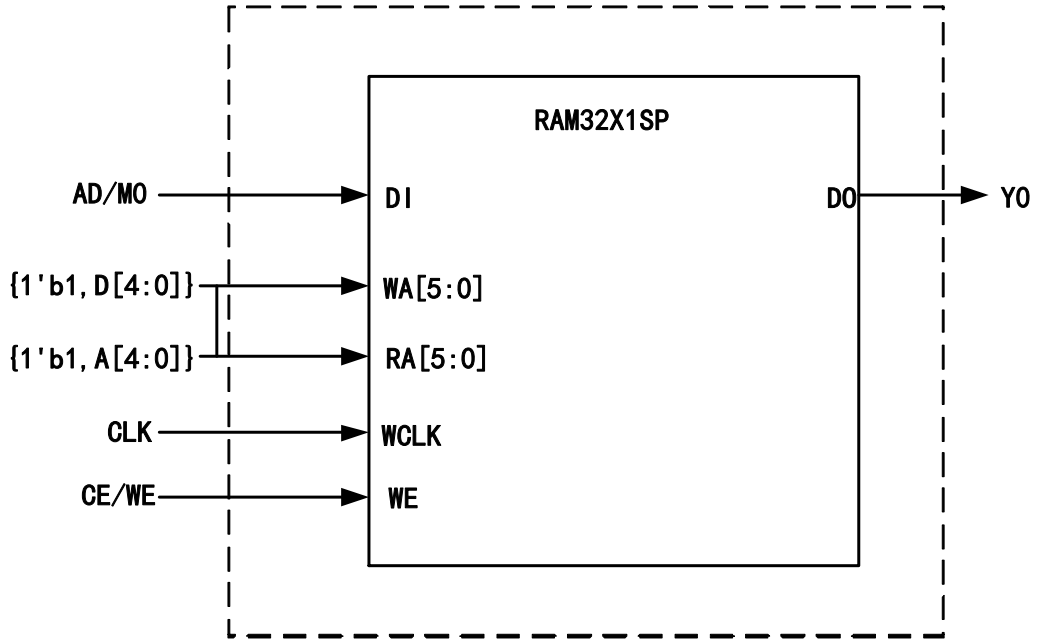


Figure 2-4 Logical Diagram of the RAM32X1SP Implementation

2.2.5.2 RAM32X1DP

Taking LUT6S_A as an example to implement a RAM32X1DP, with independent read address and write address, the D[4:0] input of LUT6S_D is multiplexed as the RAM's write address, while the read address is the A[4:0] input of LUT6S_A. The write data port is AD or M0 input, the read data output port is Y0, and the write enable is WE or CE. One LUT6S can instantiate two RAM32X1DPs with shared addresses.

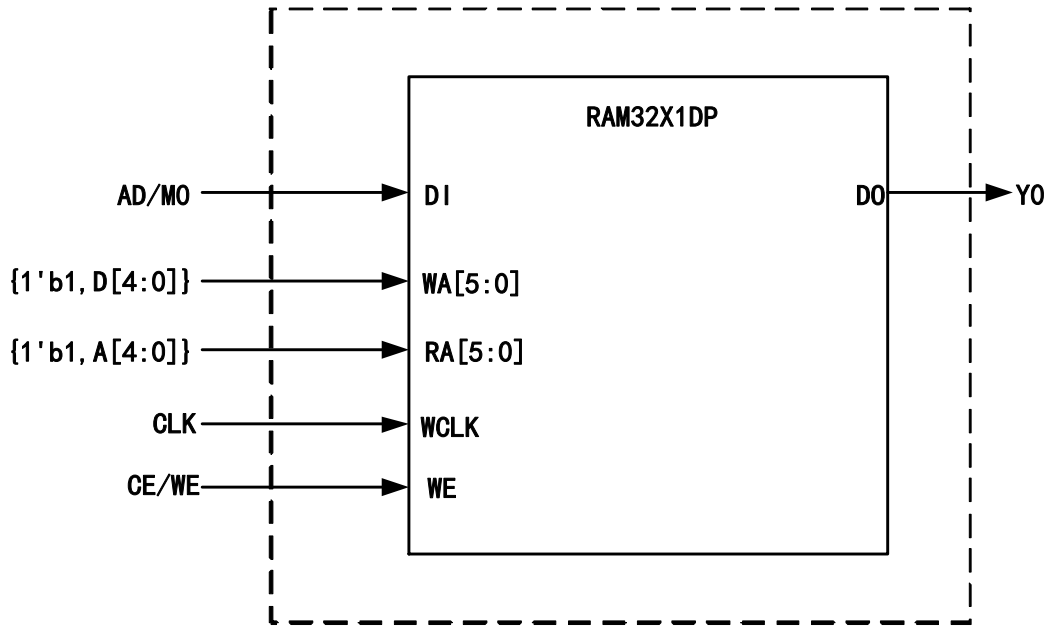


Figure 2-5 Logical Diagram of the RAM32X1DP Implementation

### 2.2.5.3 RAM64X1SP

Taking LUT6S_A as an example to implement a RAM64X1SP, the write address multiplexes the input D[5:0] of LUT6S_D, and the read address is the input A[5:0] of LUT6S_A itself. Since it is a single-port type SRAM, A[5:0] needs to be connected to D[5:0]; the write data port is either the AD input, the read data output port is Y0, and the write enable is either WE or CE. One LUT6S can instantiate one RAM64X1SP, and one CLMS can instantiate four RAM64X1SPs.

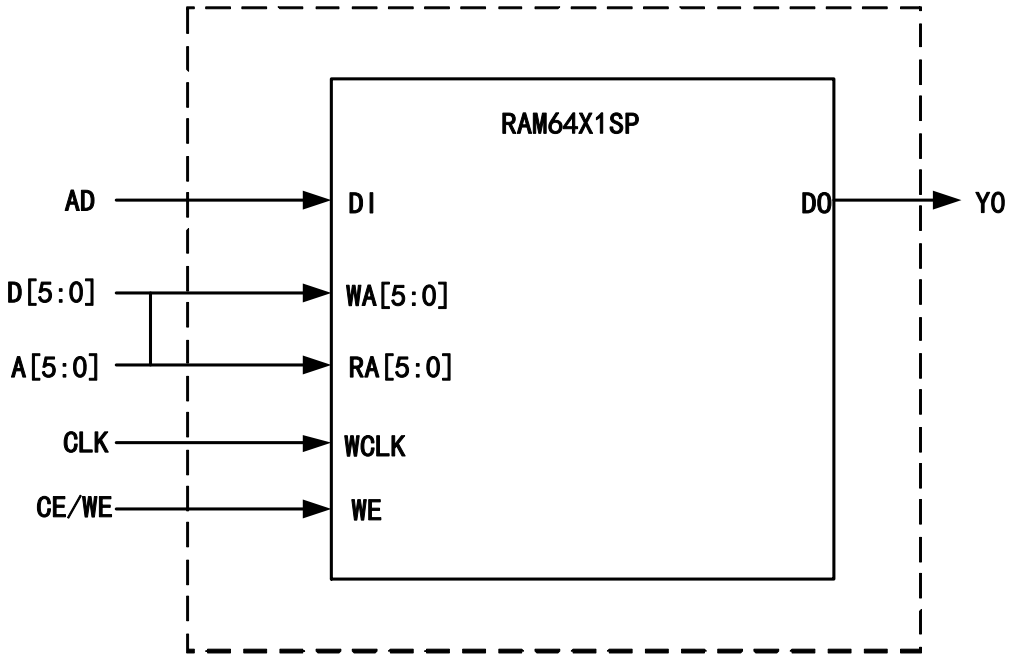


Figure 2-6 Logical Diagram of the RAM64X1SP Implementation

### 2.2.5.4 RAM64X1DP

Taking LUT6S_A as an example to implement a RAM64X1DP, with independent read and write addresses, the input D[5:0] of LUT6S_D is multiplexed as the write address of the RAM, the read address is the input A[5:0] of LUT6S_A, the write data port is the AD input, the read data output port is Y0, and the write enable is WE or CE. One LUT6S can instantiate one RAM64X1DP, and one CLMS can instantiate three RAM64X1DPs (the read and write addresses of LUT6S_D are connected together, making it unusable as simple dual-port distributed RAM).

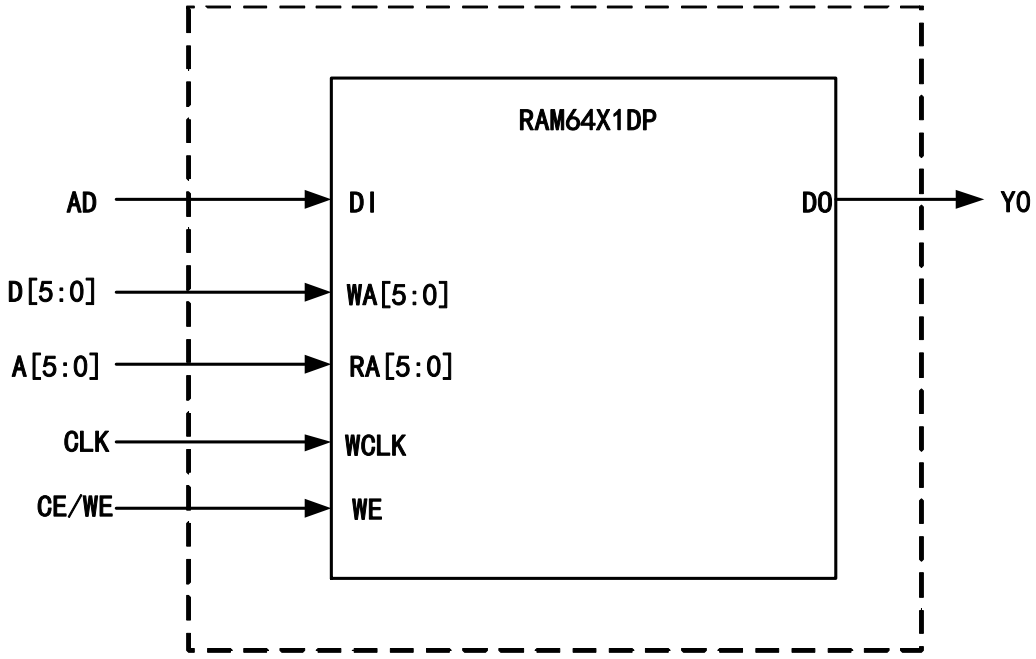


Figure 2-7 Logical Diagram of the RAM64X1DP Implementation

2.2.5.5 RAM128X1SP

FIGURE 2-8 is the diagram of a RAM128X1SP, which requires two LUT6Ss to implement. DI represents the write data of the RAM. The read and write addresses need to be connected together, with the read address using 7 bits, so the highest bit address is M0. One CLMS can instantiate two RAM128X1SPs.

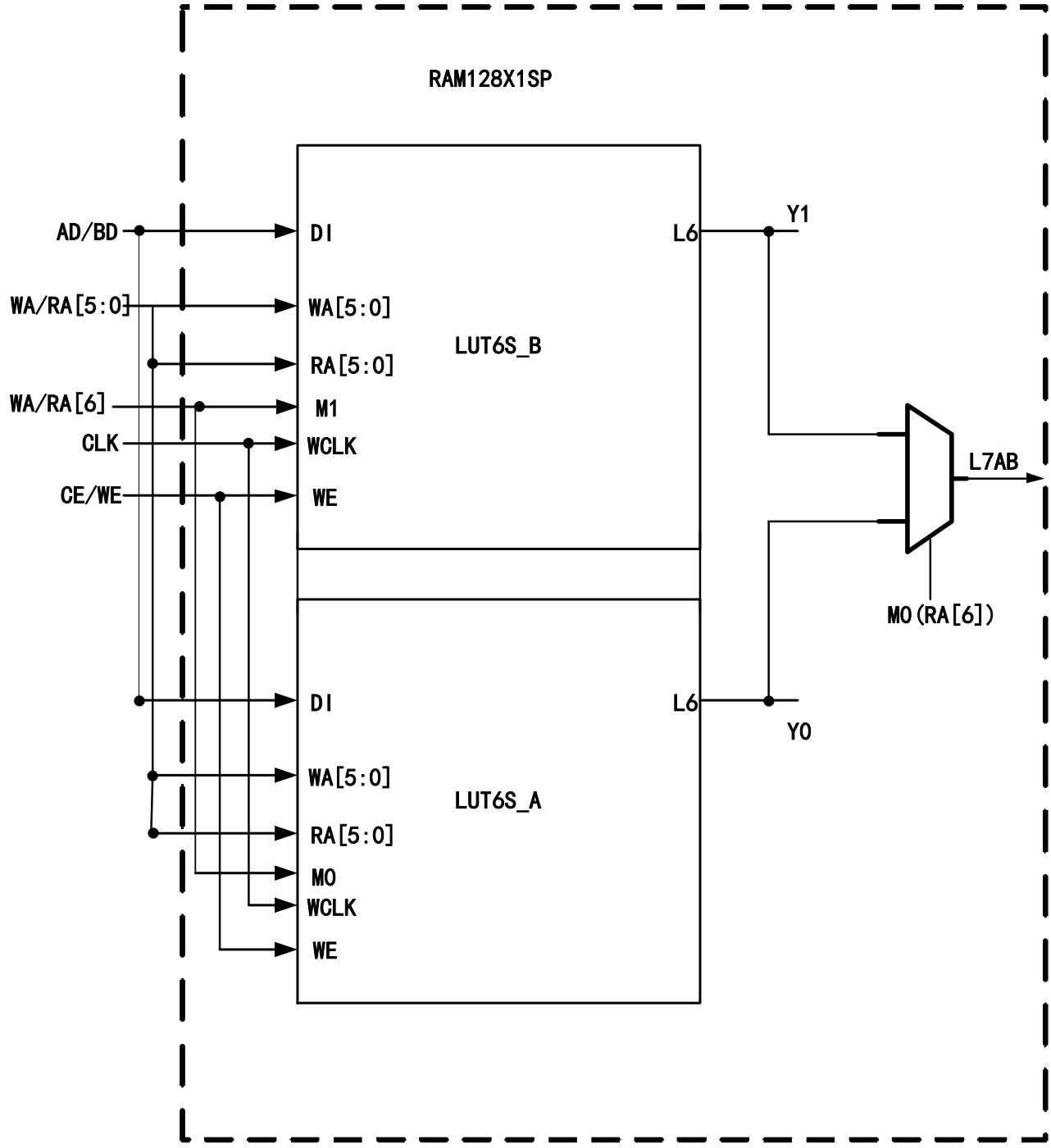


Figure 2-8 Logical Diagram of the RAM128X1SP Implementation

### 2.2.5.6 RAM128X1DP

Figure 2-9 is the diagram of a RAM128X1DP, which requires two LUT6Ss to implement. DI represents the write data of the RAM. The read address multiplexes the input D[5:0] of LUT6S_D, with M1 as the highest bit WA[6]; the write address uses the 6-bit input of LUT6S_A and LUT6S_B as the lower six bits, with M0 as the highest bit of write address. One CLMS can instantiate one RAM128X1DP (the read and write addresses of LUT6S_D are connected together, making it unusable as simple dual-port distributed RAM).

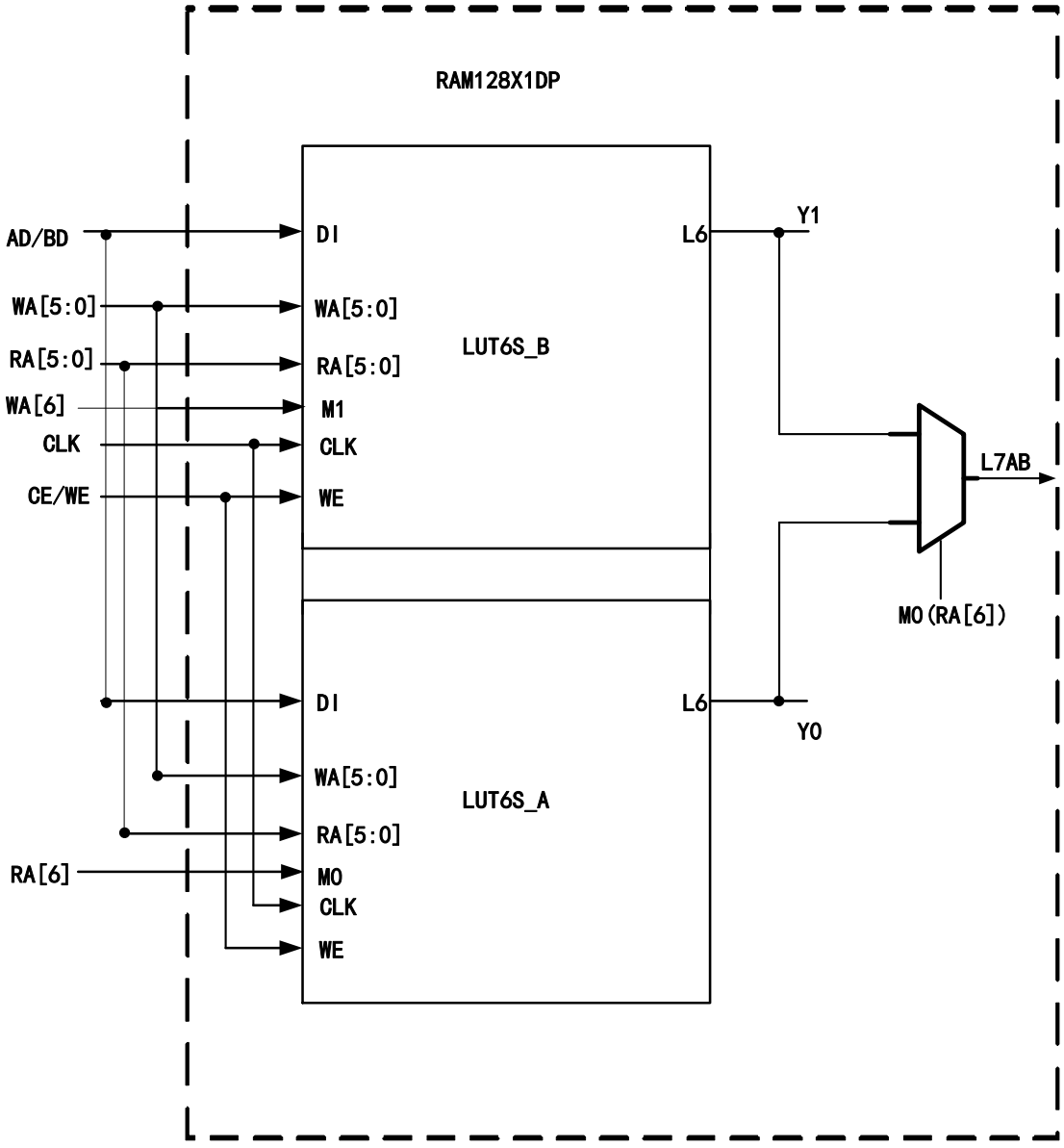


Figure 2-9 Logical Diagram of the RAM128X1DP Implementation

### 2.2.5.7 RAM256X1SP

Figure 2-10 is the diagram of a RAM256X1SP, which requires four LUT6Ss to implement. D is the write data for the RAM, input to the four LUT6S input terminals AD/BD/CD/DD. The read and write addresses need to be connected together. Here, 8 bits are used for the read address, with the highest bit address being M2. One CLMS can instantiate one RAM256X1SP.

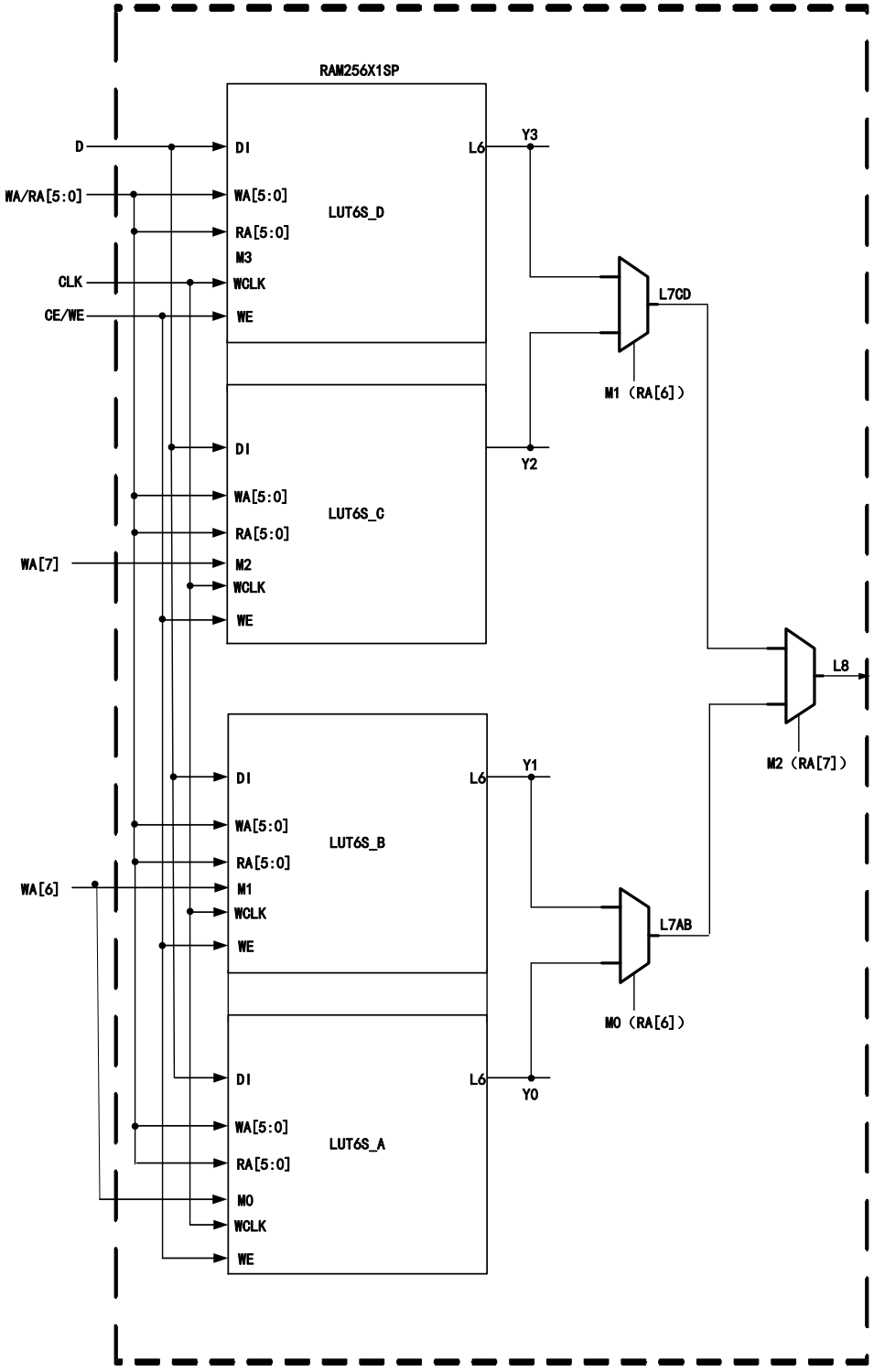


Figure 2-10 Logical Diagram of the RAM256X1SP Implementation

2.2.5.8 RAM32X2SP

Taking LUT6S_A as an example to implement a RAM32X2SP, the input D[4:0] of LUT6S_D is multiplexed as the write address, and the read address is the input A[4:0] of LUT6S_A itself. Since it is a single-port type SRAM, A[4:0] needs to be connected to D[4:0]; the write data input port is {AD,M0}, the read data output port is {CR0,Y0}, and the write enable is either WE or CE. One LUT6S can instantiate one RAM32X2SP.

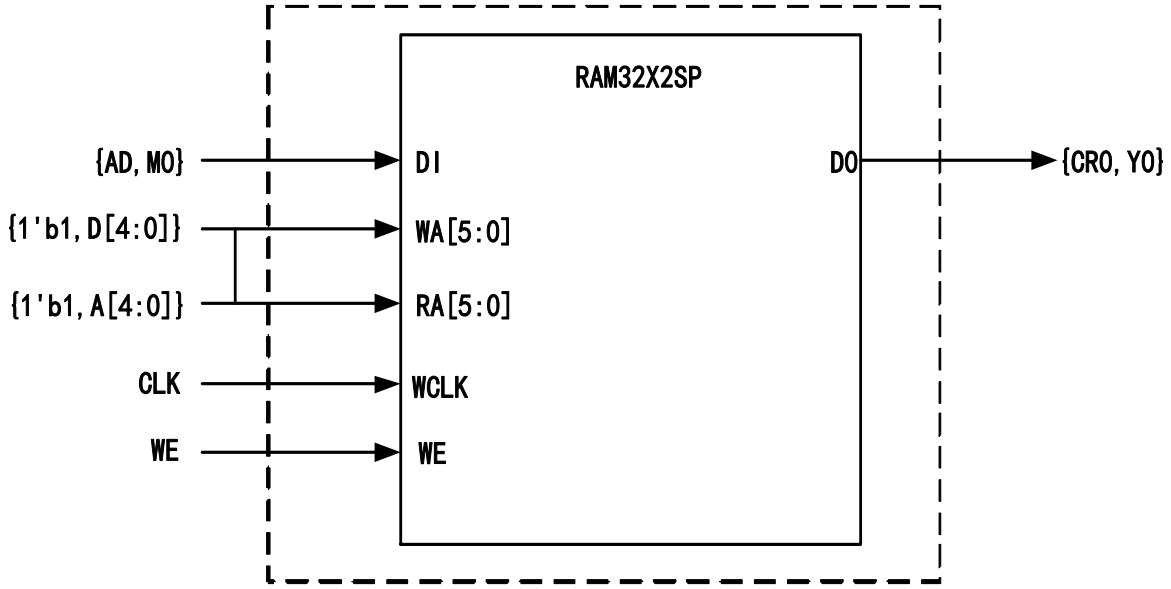


Figure 2-11 Logical Diagram of the RAM32X2SP Implementation

2.2.5.9 RAM32X2DP

Taking LUT6S_A as an example to implement a RAM32X2DP, with independent read address and write address, the D[4:0] input of LUT6S_D is multiplexed as the RAM's write address, while the read address is the A[4:0] input of LUT6S_A. The write data input port is {AD,M0}, the read data output port is {CR0,Y0}, and the write enable is WE or CE. One LUT6S can instantiate one RAM32X2DP.

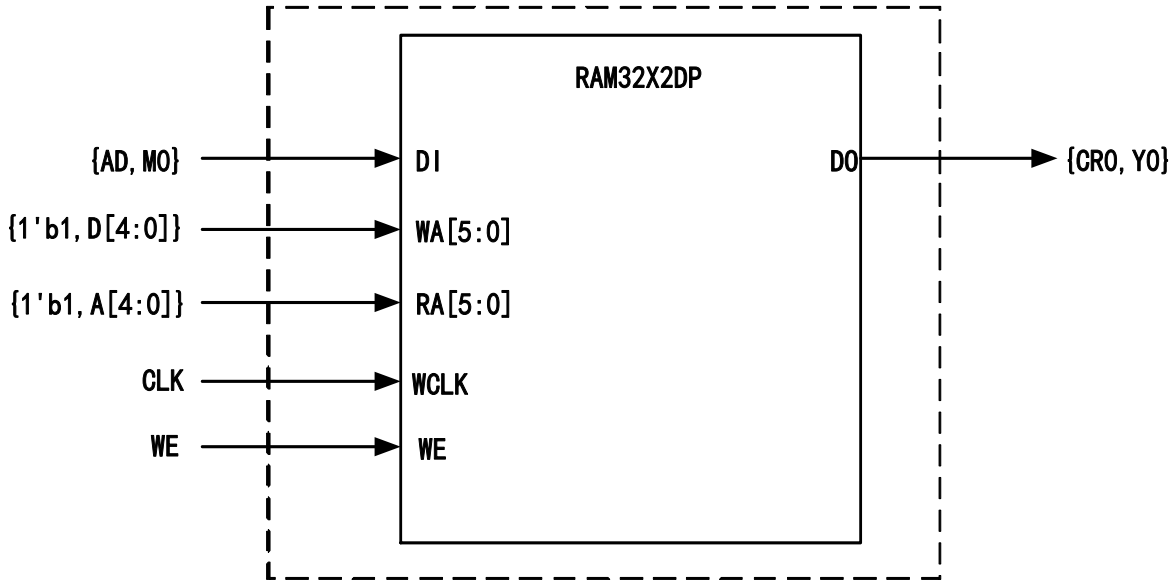


Figure 2-12 Logical Diagram of the RAM32X2DP Implementation

2.2.5.10 RAM32X2X4

Figure 2-13 is the diagram of a RAM32X2X4, which requires four LUT6Ss to implement. There are four read/write channels, each writing to the memory space of the four LUT6Ss. The read and write addresses of LUT6S_D need to be connected together. The write addresses of LUT6S_A, LUT6S_B, and LUT6S_C multiplex the read/write addresses of LUT6S_D. One CLMS can instantiate one RAM32X2X4.

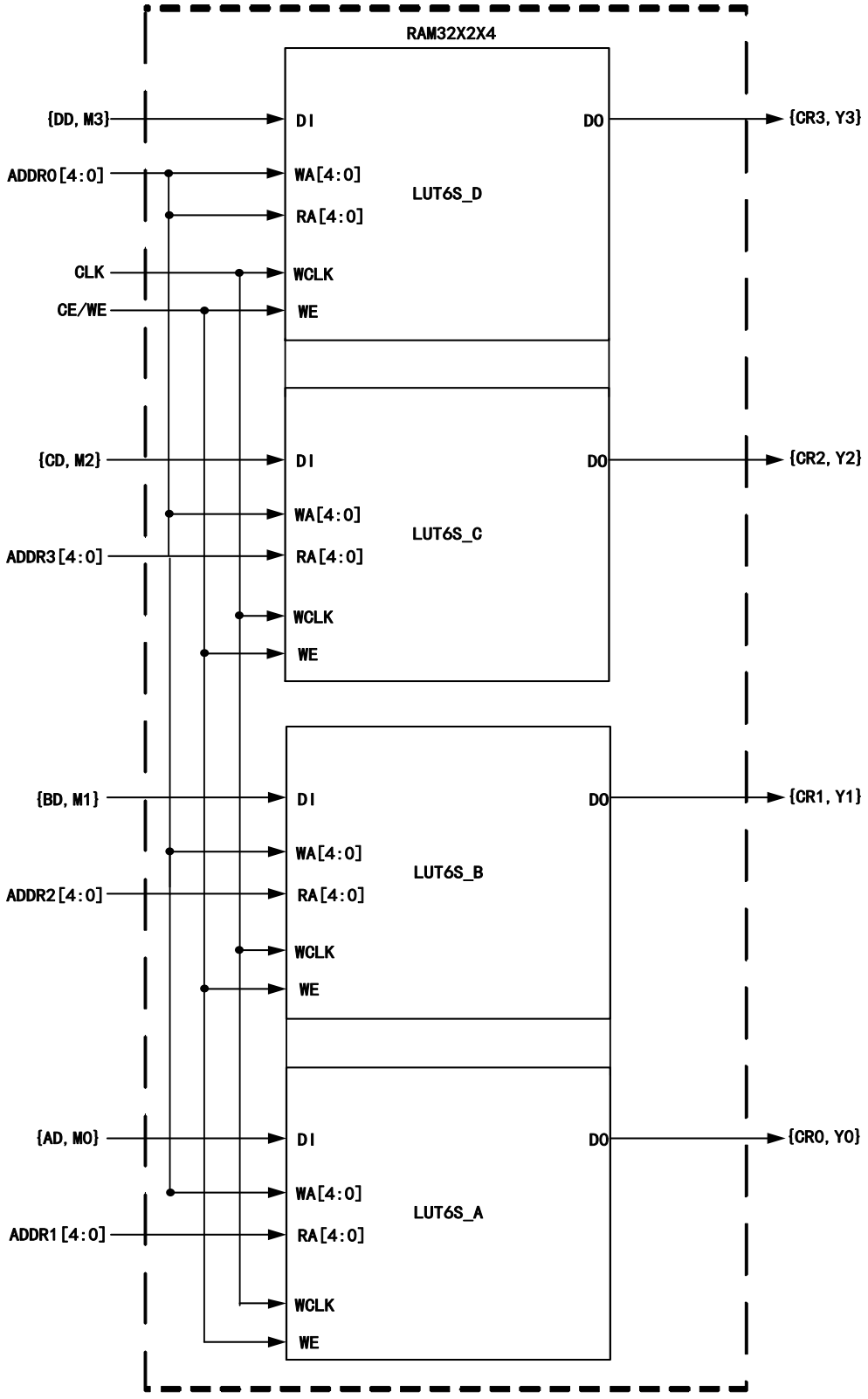


Figure 2-13 Logical Diagram of the RAM32X2X4 Implementation

2.2.5.11 RAM64X1X4

Figure 2-14 is the diagram of a RAM64X1X4, which requires four LUT6Ss to implement. There are four read/write channels, each writing to the memory space of the four LUT6Ss. The read and write addresses of LUT6S_D need to be connected together. The write addresses of LUT6S_A, LUT6S_B, and LUT6S_C multiplex the read/write addresses of LUT6S_D. One CLMS can instantiate one RAM64X1X4.

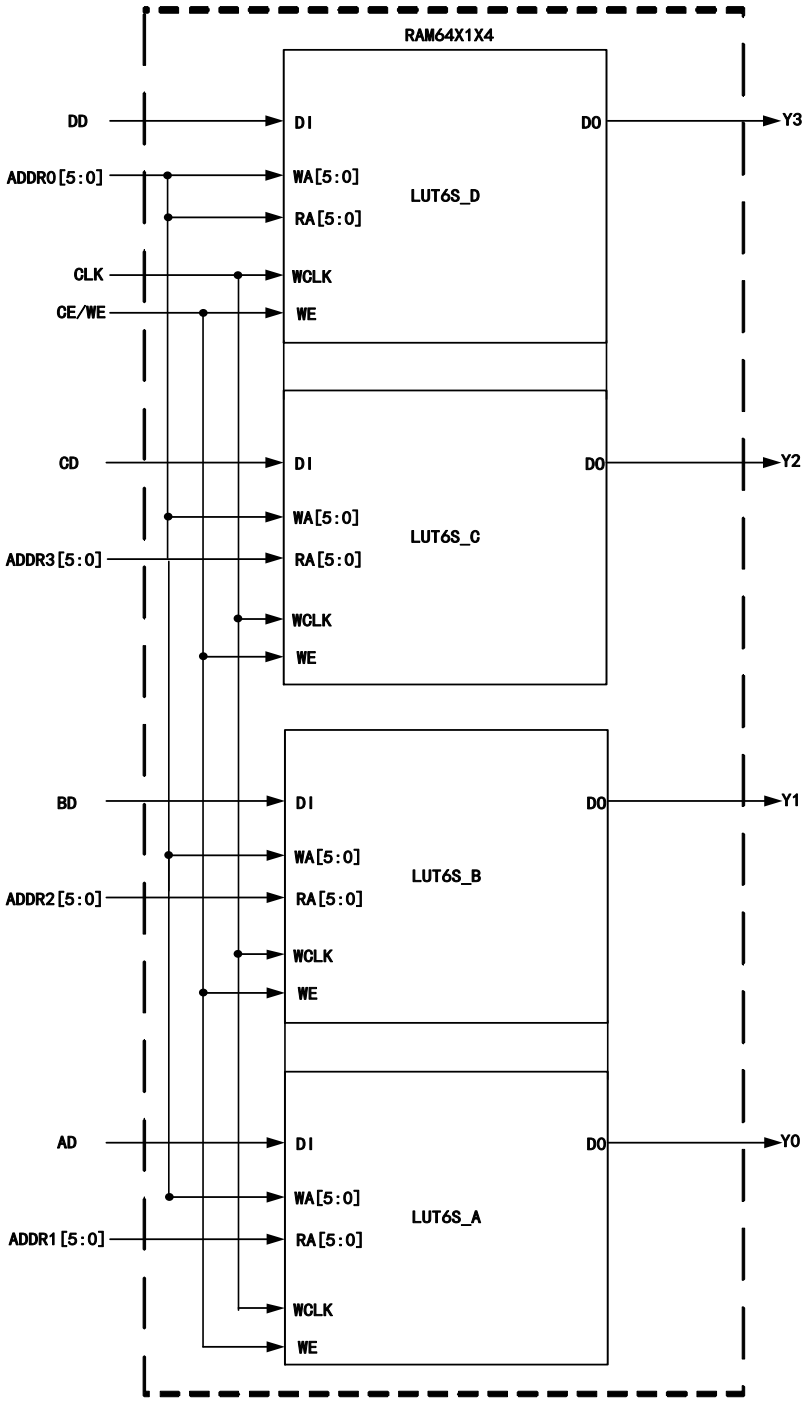


Figure 2-14 Logical Diagram of the RAM64X1X4 Implementation

## 2.3 Operating Modes of the CLM Register

The CLM of Logos2 family FPGAs has eight registers, whose configurable attributes mainly include:

- Flexible data input selection;
- Support for synchronous reset, synchronous set, asynchronous reset, or asynchronous set modes;
- The register's clock (CLK), clock enable (CE), and local reset/set (RS) signals all support polarity selection;
- Clock enable (CE) and local reset/set (RS) signals both support the fast cascade chain;
- Fast cascade chain for the shift register supported.

Registers are divided into two categories according to the data input source of the register, including 4 main registers and 4 additional registers. These two types of registers will be introduced separately.

### 2.3.1 Main Register

The main register primarily stores the output data of LUT6. Its structure is shown in Figure 2-15.

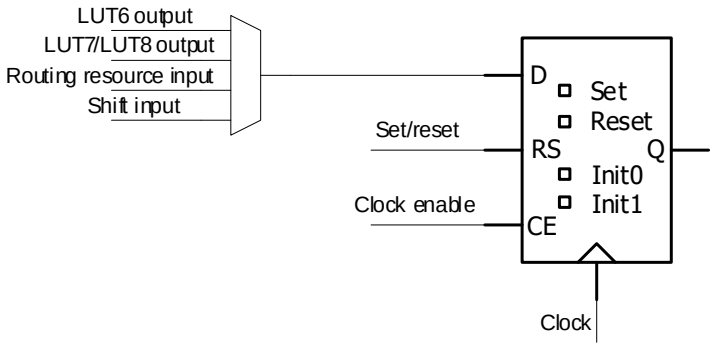


Figure 2-15 Diagram of the Main Register

The input sources for the main register are: LUT6 output, LUT7/LUT8 output, routing resource output (M0/M1/M2/M3), and shift register chain output. The main register outputs to the routing resources.

The main register's features are as follows:

- Data input from LUT, routing resource output, shift register output
- Programmable reset/set (synchronous/asynchronous, CLM global configuration mode)
- Programmable clock rising/falling edge trigger (CLM global configuration mode)
- Can be used for logic function mode, arithmetic function mode, ROM mode, distributed RAM mode
- Programmable clock/clock enable/local reset/local set control signal polarity (CLM global configuration mode)
- Global reset/set (GRS) asynchronously initializes to 0/1

- Can be configured as a shift register (comprised of four main registers and four additional registers)

### 2.3.2 Additional Register

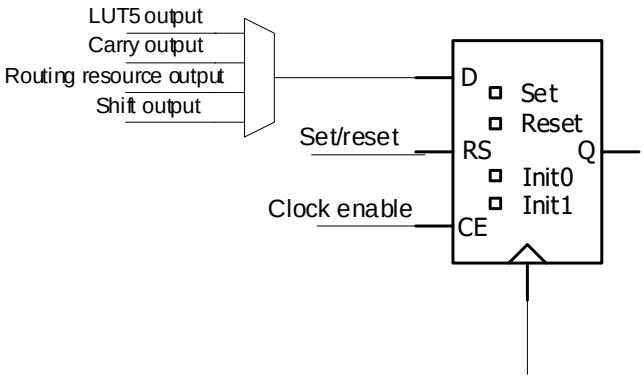


Figure 2-16 Diagram of the Additional Register

The input sources for the additional register are: LUT5 output, carry output, routing resource output (PG2L100H/PG2L200H use M0/M1/M2/M3, PG2L25H/PG2L50H use AD/BD/CD/DD), and shift register chain output. The additional register outputs to the combined output CR, which then enters the routing resources.

The additional register's features are as follows:

- Data input from LUT, routing resource output, fast carry chain, shift register output
- Programmable reset/set (synchronous/asynchronous, CLM global configuration mode)
- Programmable clock rising/falling edge trigger (CLM global configuration mode)
- Can be used for logic function mode, arithmetic function mode, ROM mode, distributed RAM mode
- Programmable clock/clock enable/local reset/local set control signal polarity (CLM global configuration mode)
- Global reset/set (GRS) asynchronously initializes to 0/1
- Can be configured as a shift register (comprised of four main registers and four additional registers)

### 2.3.3 Register Control Signal

In each CLM, all registers share the clock (CLK), clock enable (CE), and local reset/set (RS) signals. The polarity of the CLK, CE and RS signals is globally controlled by the CLM. The register control

signals also have the following characteristics:

- CE port input can be disabled, meaning the internal clock is always valid after CE port input is disabled
- RS port input can be disabled, meaning the internal register local reset/set is always invalid
- Eight registers form a 8-bit shift register chain, with the clock coming from the local CLK
- CE and RS have built-in control cascade chains (the CLM in the same column are cascaded in an order from bottom to top)

Notably, the CLM structure of PG2L200H differs from other devices in the Logos2 family. The CLM of PG2L200H has two clock enable (CE) ports (namely CE0 and CE1), each controlling four registers. Additionally, a 2-to-1 MUX is added. When the CE signals for the 8 registers are the same, signals can be input from CE0 and delivered to the 8 registers via the added MUX. The differences between the CE signals of PG2L200H and other devices in the Logos2 family are illustrated in Figure 2-17.

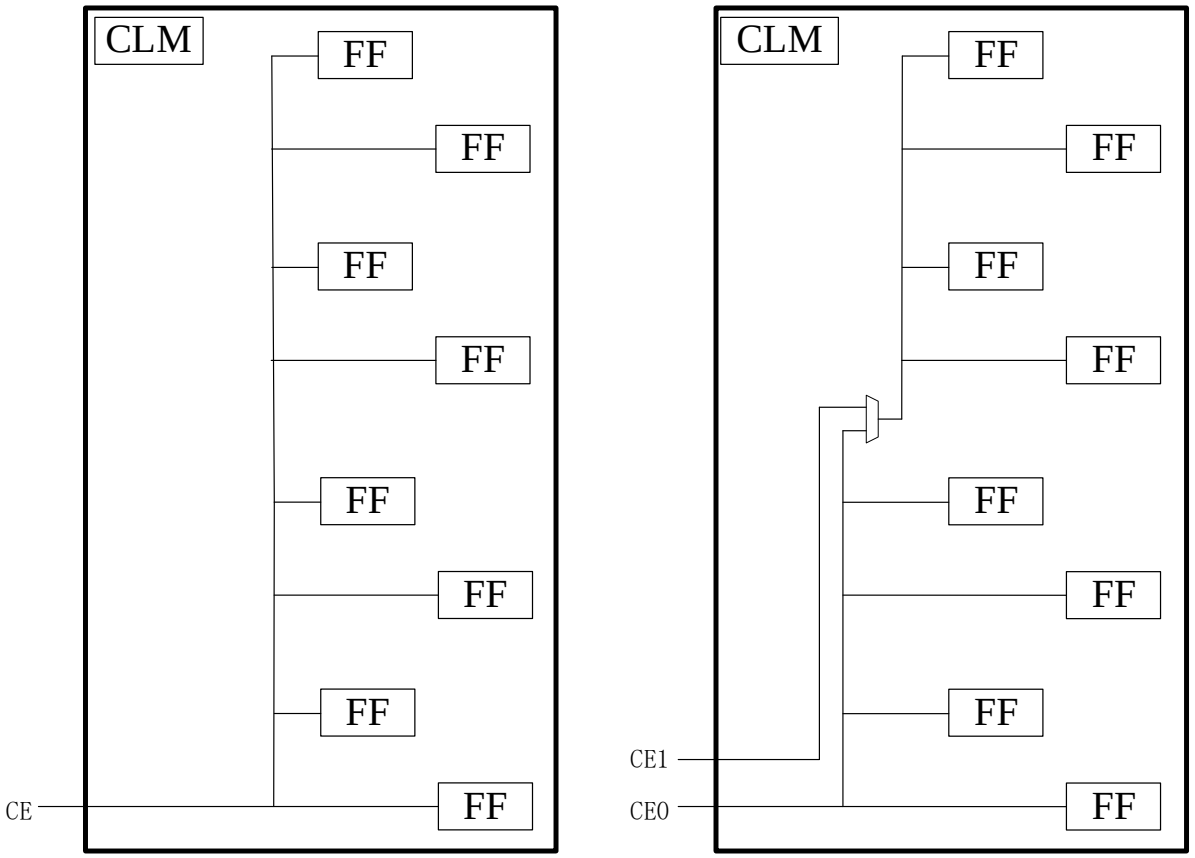


Diagram of the CE signals of other Logos2 devices

Diagram of the CE signals of PG2L200H

Figure 2-17 Diagram of the Differences Between the CE Signals of PG2L200H and Other Devices in the Logos2 Family

## 2.4 CLM Timing Parameters

Table 2-5 describes the typical timing parameters of the CLM:

Table 2-5 Typical Timing Parameters of the CLM

| Parameter | Control Signals | Description |
|---|---|---|
| Tceck | CE | The setup of CE with respect to DFF |
| Tsu/Thd | Ax/Bx/Cx/Dx | The setup/hold of Ax/Bx/Cx/Dx with respect to DFF |
| Titoy6 | Y0/Y1/Y2/Y3 | The delay from LUT6 input Ax/Bx/Cx/Dx to Y0/Y1/Y2/Y3 |
| Trsck | RS | The setup of RS with respect to DFF |
| Tco | Q0/Q1/Q2/Q3 | The TCO of CLK input with respect to Q0/Q1/Q2/Q3 |
| Tctocr | CR0/CR1/CR2/CR3 | The TCO of CLK input with respect to CR0/CR1/CR2/CR3 |

Note: For detailed descriptions of timing parameters, please refer to ***"DS04001_Logos2 Family FPGAs Device Data Sheet"***.

Figure 2-18 is the Timing Diagram of the CLM.

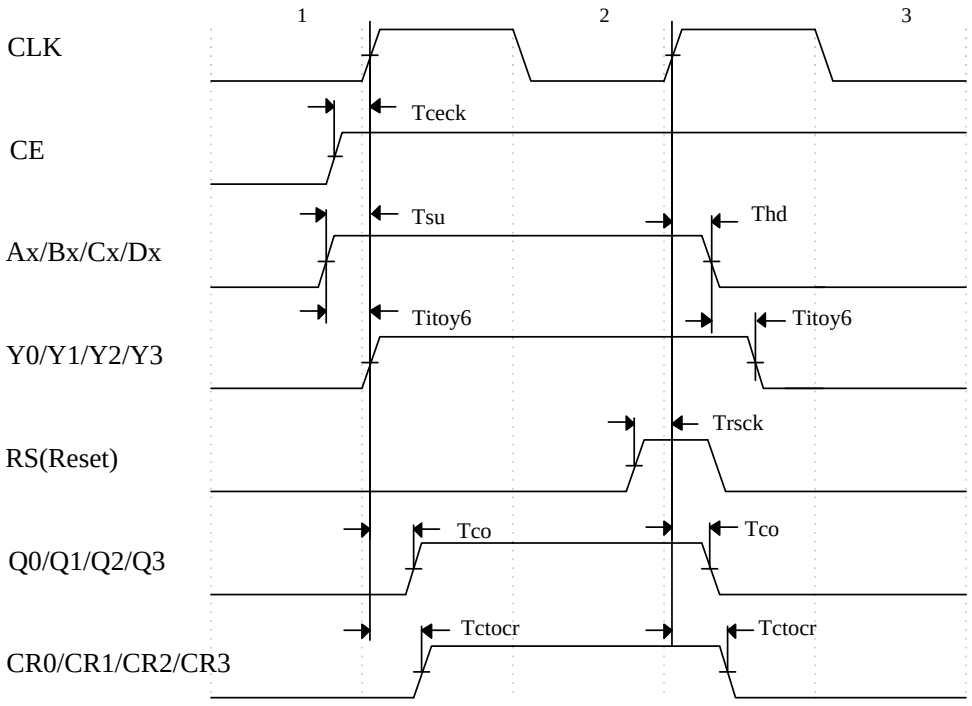


Figure 2-18 CLM Timing Diagram

Note: x = 0, 1, 2, 3, 4, 5.

# Chapter 3 Application Example

## 3.1 Commonly Used GTP

This section mainly introduces several commonly used CLM GTP primitives, which users can directly instantiate in their designs. The support status, ports and other parameters of each GTP are subject to the ***"UG040007_Logos2 Family Product GTP User Guide"***.

### 3.1.1 Multiplexer GTP

Table 3-1 Multiplexer GTP

| GTP | Input Source | Hardware Resource | Description |
|---|---|---|---|
| GTP_MUX2LUT7 | Output of LUT6 | L7ABMUX or L7CDMUX | Used for constructing LUT7, 8:1 multiplexer, and other logic |
| GTP_MUX2LUT8 | Output of L7ABMUX and L7CDMUX | L8MUX | Used for constructing LUT8, 16:1 multiplexer, and other logic |

GTP_MUX2LUT7 and GTP_MUX2LUT8 have the same ports, as shown in Figure 3-1.

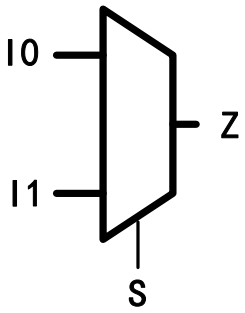


Figure 3-1 The Port Diagram of Multiplexer GTP

The port description of the multiplexer is shown in Table 3-2.

Table 3-2 Port Description of the Multiplexer

| Port | Input/Output | Function Description |
|---|---|---|
| I0 | Input | Input Signal |
| I1 | Input | Input Signal |
| S | Input | Selection Signal |
| Z | Output | Output Signal |

The multiplexer's truth table is shown in Table 3-3 (X represents any value). When the select signal S is 1'b0, the output is I0; when S is 1'b1, the output is I1.

Table 3-3 Truth Table of the Multiplexer

| **Input** | | | **Output** |
|---|---|---|---|
| **I1** | **I0** | **S** | **Z** |
| X | 0 | 0 | 0 |
| X | 1 | 0 | 1 |
| 0 | X | 1 | 0 |
| 1 | X | 1 | 1 |

### 3.1.2 Carry Chain GTP

The carry chain primitive GTP_LUT6CARRY represents the fast carry logic in the CLM, and is often used in constructing arithmetic logic such as adders and multipliers (refer to 2.2 Operating Modes of LUT6A and LUT6S).

GTP_LUT6CARRY is a fast carry logic, where different INIT parameter values correspond to different functions, and the corresponding INIT parameter must be specified when used.

Table 3-4 Parameter Description of Carry Chain GTP

| **Parameter Name** | **Parameter Type** | **Valid Values** | **Function Description** |
|---|---|---|---|
| INIT | <binary> | 0~2^64-1 | Output Configuration Parameter |
| I5_TO_CARRY | <string> | "TRUE" or "FALSE" | "TRUE": Select I5 to input to the carry logic;<br>"FALSE": Select LUT5B to input to the carry logic. |
| I5_TO_LUT | <string> | "TRUE" or "FALSE" | "TRUE": Select I5 to input to the LUT logic;<br>"FALSE": Select CIN to input to the LUT logic. |

The port diagram of GTP_LUT6CARRY primitive is shown in Figure 3-2.

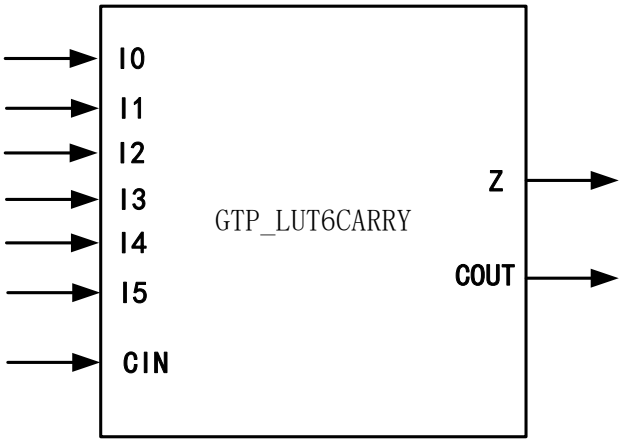


Figure 3-2 The Port Diagram of Carry Chain GTP

The port description of GTP_LUT6CARRY primitive is shown in Table 3-5.

Table 3-5 The Port Description of Carry Chain GTP

| Port | Input/Output | Function Description |
|---|---|---|
| I0 | Input | Input Signal |
| I1 | Input | Input Signal |
| I2 | Input | Input Signal |
| I3 | Input | Input Signal |
| I4 | Input | Input Signal |
| I5 | Input | Input Signal |
| CIN | Input | Input Signal, Cascaded Input |
| COUT | Output | Cascade Output |
| Z | Output | Output Signal |

The internal structure diagram of GTP_LUT6CARRY primitive is shown in Figure 3-3. The GTP_LUT6CARRY consists of 2 lut5s and 4 two-input multiplexers (mux). Among these, mux0 and mux2 dynamically select internal signals to output to COUT and Z, while mux1 and mux3 are statically selected based on parameters I5_TO_CARRY and I5_TO_LUT for fixed internal signals.

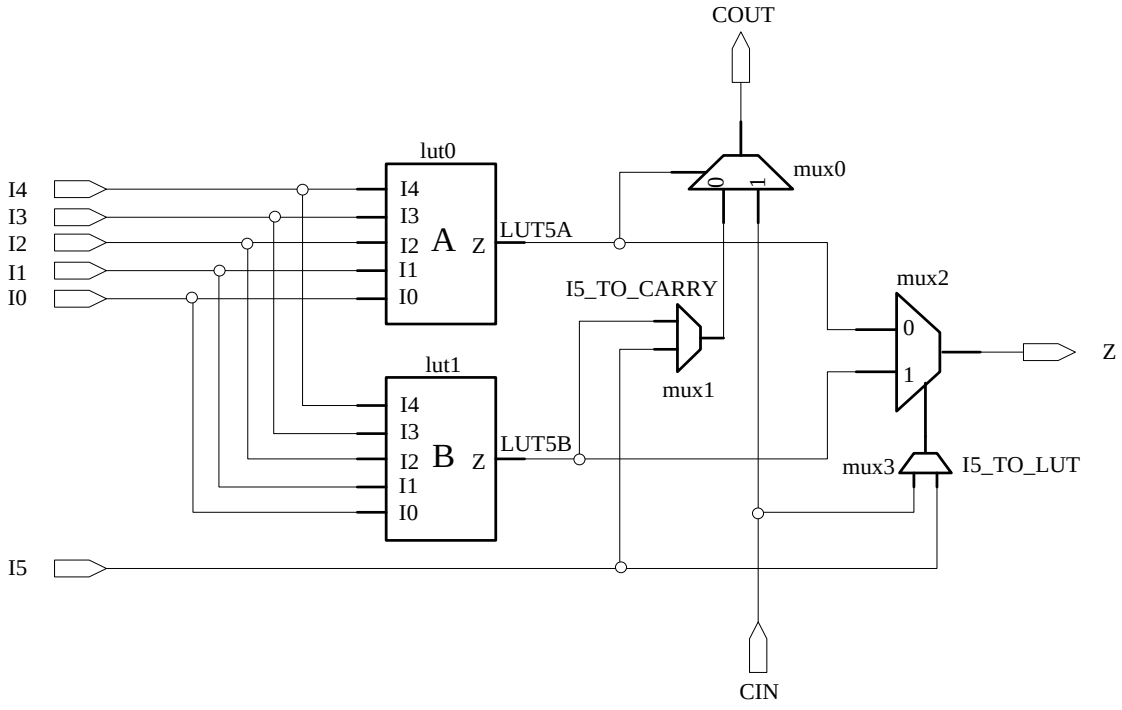


Figure 3-3 The Internal Structure Diagram of GTP_LUT6CARRY

The truth table of GTP_LUT6CARRY can be found in Table 3-6, Table 3-7 and Table 3-8.

Table 3-6 Truth Table of Internal Signals LUT5A and LUT5B

| **Input Signal** | | | | | **Output Signal** | |
|---|---|---|---|---|---|---|
| **I4** | **I3** | **I2** | **I1** | **I0** | **LUT5B** | **LUT5A** |
| 0 | 0 | 0 | 0 | 0 | INIT[32] | INIT[0] |
| 0 | 0 | 0 | 0 | 1 | INIT[33] | INIT[1] |
| 0 | 0 | 0 | 1 | 0 | INIT[34] | INIT[2] |
| 0 | 0 | 0 | 1 | 1 | INIT[35] | INIT[3] |
| 0 | 0 | 1 | 0 | 0 | INIT[36] | INIT[4] |
| 0 | 0 | 1 | 0 | 1 | INIT[37] | INIT[5] |
| 0 | 0 | 1 | 1 | 0 | INIT[38] | INIT[6] |
| 0 | 0 | 1 | 1 | 1 | INIT[39] | INIT[7] |
| 0 | 1 | 0 | 0 | 0 | INIT[40] | INIT[8] |
| 0 | 1 | 0 | 0 | 1 | INIT[41] | INIT[9] |
| 0 | 1 | 0 | 1 | 0 | INIT[42] | INIT[10] |
| 0 | 1 | 0 | 1 | 1 | INIT[43] | INIT[11] |
| 0 | 1 | 1 | 0 | 0 | INIT[44] | INIT[12] |
| 0 | 1 | 1 | 0 | 1 | INIT[45] | INIT[13] |
| 0 | 1 | 1 | 1 | 0 | INIT[46] | INIT[14] |
| 0 | 1 | 1 | 1 | 1 | INIT[47] | INIT[15] |
| 1 | 0 | 0 | 0 | 0 | INIT[48] | INIT[16] |
| 1 | 0 | 0 | 0 | 1 | INIT[49] | INIT[17] |
| 1 | 0 | 0 | 1 | 0 | INIT[50] | INIT[18] |
| 1 | 0 | 0 | 1 | 1 | INIT[51] | INIT[19] |
| 1 | 0 | 1 | 0 | 0 | INIT[52] | INIT[20] |
| 1 | 0 | 1 | 0 | 1 | INIT[53] | INIT[21] |
| 1 | 0 | 1 | 1 | 0 | INIT[54] | INIT[22] |
| 1 | 0 | 1 | 1 | 1 | INIT[55] | INIT[23] |
| 1 | 1 | 0 | 0 | 0 | INIT[56] | INIT[24] |
| 1 | 1 | 0 | 0 | 1 | INIT[57] | INIT[25] |
| 1 | 1 | 0 | 1 | 0 | INIT[58] | INIT[26] |
| 1 | 1 | 0 | 1 | 1 | INIT[59] | INIT[27] |
| 1 | 1 | 1 | 0 | 0 | INIT[60] | INIT[28] |
| 1 | 1 | 1 | 0 | 1 | INIT[61] | INIT[29] |
| 1 | 1 | 1 | 1 | 0 | INIT[62] | INIT[30] |
| 1 | 1 | 1 | 1 | 1 | INIT[63] | INIT[31] |

Table 3-7 Truth Table of Output Signal COUT

| **Parameter** | **Input Signal** | | | | **Output signal** |
|---|---|---|---|---|---|
| **I5_TO_CARRY** | **LUT5A** | **LUT5B** | **I5** | **CIN** | **COUT** |

| Parameter | Input Signal | | | | Output signal |
|---|---|---|---|---|---|
| **I5_TO_CARRY** | **LUT5A** | **LUT5B** | **I5** | **CIN** | **COUT** |
| "FALSE" | 0 | 0 | X | 0 | 0 |
| "FALSE" | 0 | 0 | X | 1 | 0 |
| "FALSE" | 0 | 1 | X | 0 | 1 |
| "FALSE" | 0 | 1 | X | 1 | 1 |
| "FALSE" | 1 | 0 | X | 0 | 0 |
| "FALSE" | 1 | 0 | X | 1 | 1 |
| "FALSE" | 1 | 1 | X | 0 | 0 |
| "FALSE" | 1 | 1 | X | 1 | 1 |
| "TRUE" | 0 | X | 0 | 0 | 0 |
| "TRUE" | 0 | X | 0 | 1 | 0 |
| "TRUE" | 0 | X | 1 | 0 | 1 |
| "TRUE" | 0 | X | 1 | 1 | 1 |
| "TRUE" | 1 | X | 0 | 0 | 0 |
| "TRUE" | 1 | X | 0 | 1 | 1 |
| "TRUE" | 1 | X | 1 | 0 | 0 |
| "TRUE" | 1 | X | 1 | 1 | 1 |

Table 3-8 Truth Table of Output Signal Z

| Parameter | Input Signal | | | | Output signal |
|---|---|---|---|---|---|
| **I5_TO_LUT** | **I5** | **CIN** | **LUT5A** | **LUT5B** | **Z** |
| "FALSE" | X | 0 | 0 | 0 | 0 |
| "FALSE" | X | 0 | 0 | 1 | 0 |
| "FALSE" | X | 0 | 1 | 0 | 1 |
| "FALSE" | X | 0 | 1 | 1 | 1 |
| "FALSE" | X | 1 | 0 | 0 | 0 |
| "FALSE" | X | 1 | 0 | 1 | 1 |
| "FALSE" | X | 1 | 1 | 0 | 0 |
| "FALSE" | X | 1 | 1 | 1 | 1 |
| "TRUE" | 0 | X | 0 | 0 | 0 |
| "TRUE" | 0 | X | 0 | 1 | 0 |
| "TRUE" | 0 | X | 1 | 0 | 1 |
| "TRUE" | 0 | X | 1 | 1 | 1 |
| "TRUE" | 1 | X | 0 | 0 | 0 |
| "TRUE" | 1 | X | 0 | 1 | 1 |
| "TRUE" | 1 | X | 1 | 0 | 0 |
| "TRUE" | 1 | X | 1 | 1 | 1 |

### 3.1.3 Distributed RAM GTP

To use the distributed RAM resources in CLMS, users can instantiate distributed RAM primitives. The Logos2 family FPGAs supports the distributed RAM GTP shown in Table 3-9.

Table 3-9 RAM GTP

| GTP | Density | Type | Address Inputs |
|---|---|---|---|
| GTP_RAM32X1SP | 32-bit | Single port | A0–4/B0–4/C0–4/D0–4 (write/read) |
| GTP_RAM32X1DP | 32-bit | Simple dual port | D0~4 (write)<br>A0~4/B0~4/C0~4 (read) |
| GTP_RAM64X1SP | 64-bit | Single port | A0~5/B0~5/C0~5/D0~5 (write/read) |
| GTP_RAM64X1DP | 64-bit | Simple dual port | D0~5 (write)<br>A0~5/B0~5/C0~5 (read) |
| GTP_RAM128X1SP | 128-bit | Single port | A0~5/B0~5/C0~5/D0~5 are used as lower 6-bit address, M0/M1 is used as the highest bit address |
| GTP_RAM128X1DP | 128-bit | Simple dual port | D0~5 are used as the lower 6-bit write addresses, and M1 is used as the 7th bit write address;<br>A0~5/B0~5/C0~5 are used as the lower 6-bit read addresses, and M0 is used as the 7th bit read address |
| GTP_RAM256X1SP | 256-bit | Single port | A0~5/B0~5/C0~5/D0~5 are used as the lower 6-bit addresses, M0/M1 as the 7th bit address, and M2 as the highest bit address (write/read) |
| GTP_RAM32X2SP | 64-bit | Single port | A0–4/B0–4/C0–4/D0–4 (write/read) |
| GTP_RAM32X2DP | 64-bit | Simple dual port | D0~4 (write)<br>A0~4/B0~4/C0~4 (read) |
| GTP_RAM32X2X4 | 32-bitX4 | Configurable | D0~4 (write)<br>A0–4/B0–4/C0–4/D0–4 (read) |
| GTP_RAM64X1X4 | 64-bitX4 | Configurable | D0~5 (write)<br>A0~5/B0~5/C0~5 (/D0~5 read) |

Distributed RAM can set initial value parameters. Taking 256-bit RAM as an example, please refer to Table 3-10. For detailed parameters of other distributed RAM GTP, please refer to ***"UG040007_Logos2 Family Products GTP User Guide"***.

Table 3-10 Description of Distributed RAM GTP Parameters

| Parameter Name | Parameter Type | Valid Values | Function Description |
|---|---|---|---|
| INIT | <binary> | 256'h0~256'hffff_ffff_ffff_ffff_ffff_ffff_ffff_ffff_ffff_ffff_ffff_ffff_ffff_ffff_ffff_ffff | Memory initialization configuration parameters |

The port diagram of the distributed single-port and simple dual-port RAMs supported by the Logos2 family FPGAs is shown in Figure 3-4.

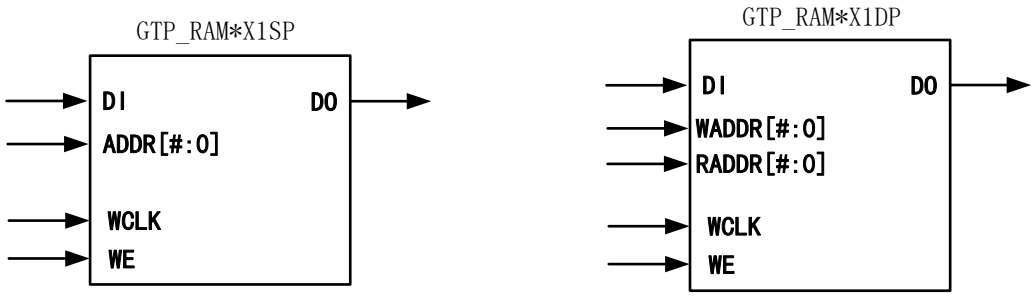


Figure 3-4 The Port Diagram of RAM GTP

The description of the ports of the distributed single-port and simple dual-port RAMs supported by the Logos2 Family FPGAs is shown in Table 3-11.

Table 3-11 The Port Description of Distributed RAM GTP

| **Port Name** | **Input/Output** | **Description** |
|---|---|---|
| DI | Input | Write data signal |
| DO | Output | Read data signal |
| [R/W]ADDR | Input | **Single port:**<br>ADDR: Address signal, shared for read and write;<br>**Simple dual port:**<br>WADDR: write address<br>RADDR: read address |
| WCLK | Input | Clock signal |
| WE | Input | Write enable signal, active-high |

The port timing diagram for the distributed single-port RAM supported by the Logos2 family FPGAs is shown in Figure 3-5, where MEM is the data stored at the corresponding address. The distributed single-port RAM operates with synchronous write and asynchronous read. Before data is written on the rising edge of the third clock pulse, the address has already changed to A0, and DO outputs the old data MEM(A0) due to the address change; data D0 is written to the RAM's A0 address, then outputs to the DO port after a certain delay; when the address changes to A1 and WE is 0, DO outputs MEM (A1).

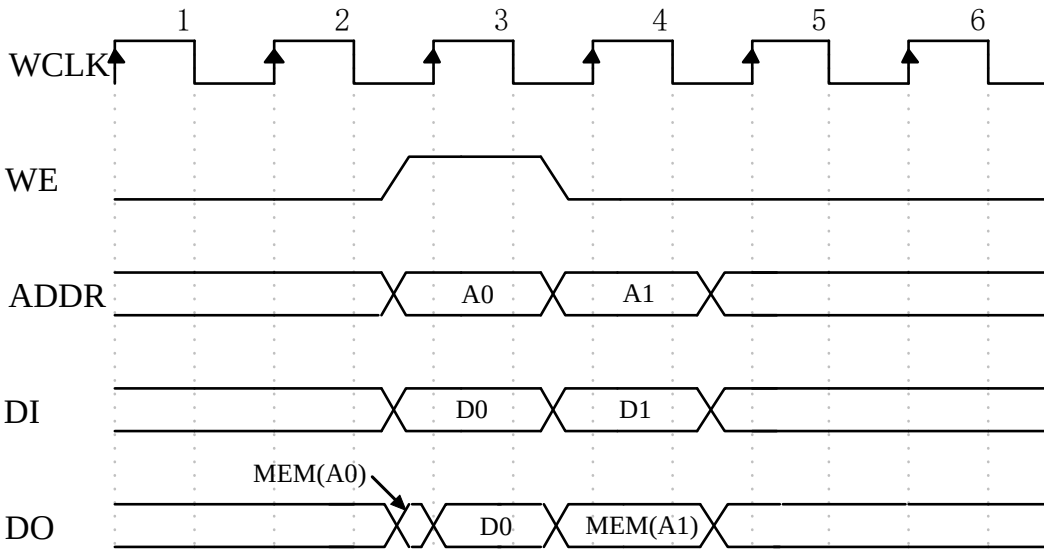


Figure 3-5 Distributed Single-Port RAM Timing Diagram

The port timing diagram of the distributed simple dual-port RAM supported by the Logos2 family FPGAs is shown in Figure 3-6, where MEM is the data stored at the corresponding address. The RAM operates with synchronous write and asynchronous read. During the third and fourth clock cycles, with WE high, data D0 and D1 are written to addresses A0 and A1 on the clock's rising edge; when the read address RADDR changes to A0 and A1, the read data at DO is D0 and D1 respectively.

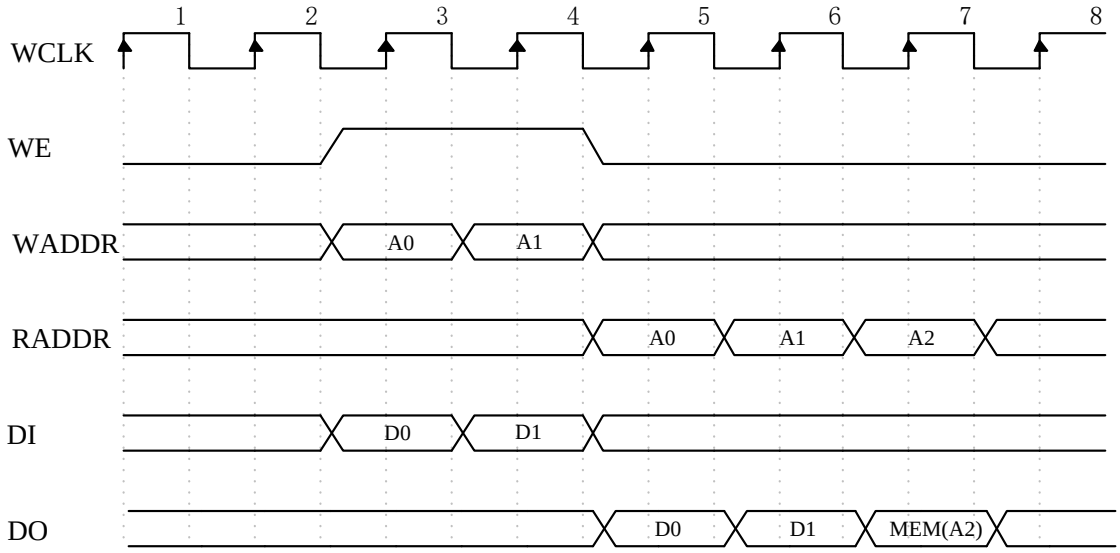


Figure 3-6 Distributed Simple Dual-Port RAM Timing Diagram

### 3.1.4 ROM GTP

To use CLM as a read-only memory, users can instantiate ROM primitives, as the Logos2 family FPGAs supports the ROM GTP shown in Table 3-12.

Table 3-12 ROM GTP

| GTP | Density | Address Inputs |
|---|---|---|
| GTP_ROM32X1 | 32-bit | A0–4/B0–4/C0–4/D0–4 |
| GTP_ROM32X2 | 64-bit | A0–4/B0–4/C0–4/D0–4 |
| GTP_ROM64X1 | 64-bit | A0~5/B0~5/C0~5/D0~5 |
| GTP_ROM128X1 | 128-bit | A0~5/B0~5/C0~5/D0~5 are used as the lower 6-bit address, M0/M1 are used as the highest bit address |
| GTP_ROM256X1 | 256-bit | A0~5/B0~5/C0~5/D0~5 are used as the lower 6-bit addresses, M0/M1 as the 7th bit address, and M2 as the highest bit address |

Distributed ROM can set initial value parameters, taking 256bit ROM as an example, please refer to Table 3-13.

Table 3-13 The Parameter Description of Distributed ROM GTP

| Parameter Name | Parameter Type | Valid Values | Function Description |
|---|---|---|---|
| INIT | <binary> | 256'h0~256'hffff_ffff_ffff_ffff_ffff_ffff_ffff_ffff_ffff_ffff_ffff_ffff_ffff_ffff_ffff_ffff | ROM Initialization Configuration Parameters |

Taking GTP_ROM256X1 as an example, the port diagram of ROM GTP is shown in Figure 3-7.

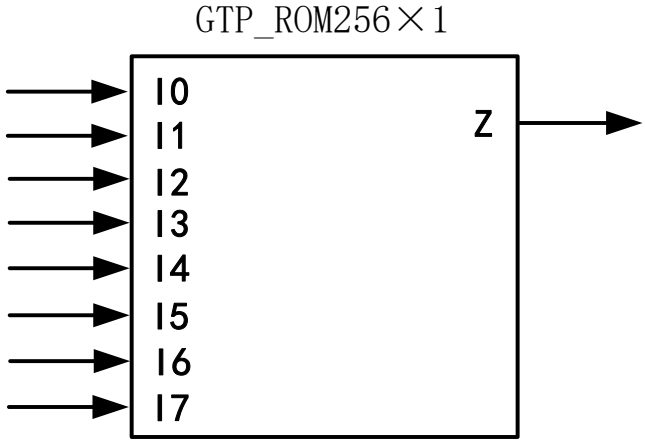


Figure 3-7 The Port Diagram of ROM GTP

The port description of GTP_ROM256X1 is shown in Table 3-14.

Table 3-14 Description of Distributed ROM GTP Ports

| Port | Input/Output | Function Description |
|---|---|---|
| I0 | Input | ROM read address addr[0] |
| I1 | Input | ROM read address addr[1] |
| I2 | Input | ROM read address addr[2] |

| Port | Input/Output | Function Description |
|---|---|---|
| I3 | Input | ROM read address addr[3] |
| I4 | Input | ROM read address addr[4] |
| I5 | Input | ROM read address addr[5] |
| I6 | Input | ROM read address addr[6] |
| I7 | Input | ROM read address addr[7] |
| Z | Output | Read data |

This GTP implements the ROM storage function. Inputs I7–I0 constitute the read data address, reading the value of the specified bit of the ROM's initial configuration parameters.

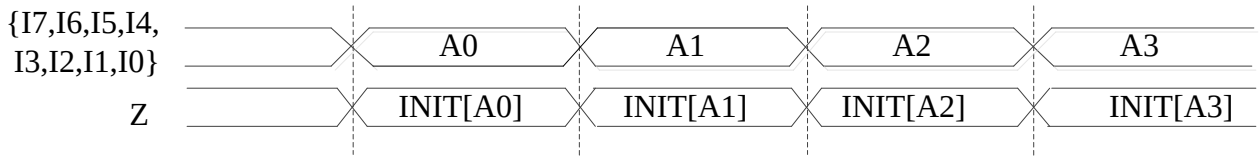


Figure 3-8 GTP_ROM256X1 Waveform Diagram

For details about other distributed ROM GTPs, please refer to ***"UG040007_Logos2 Family Products GTP User Guide"***.

### 3.1.5 Flip-flop GTP

The Logos2 family FPGAs supports the flip-flop GTP as shown in Table 3-15.

Table 3-15 Flip-flop GTP

| GTP | Description |
|---|---|
| GTP_DFF | Flip-flop |
| GTP_DFF_C | Asynchronous clear flip-flop |
| GTP_DFF_E | Flip-flop with enable |
| GTP_DFF_CE | Asynchronous clear flip-flop with enable |
| GTP_DFF_P | Asynchronous set flip-flop |
| GTP_DFF_PE | Asynchronous set flip-flop with enable |
| GTP_DFF_R | Synchronous clear flip-flop |
| GTP_DFF_RE | Synchronous clear flip-flop with enable |
| GTP_DFF_S | Synchronous set flip-flop |
| GTP_DFF_SE | Synchronous set flip-flop with enable |

Taking GTP_DFF_CE as an example, the port diagram of the flip-flop GTP is shown in Figure 3-9.

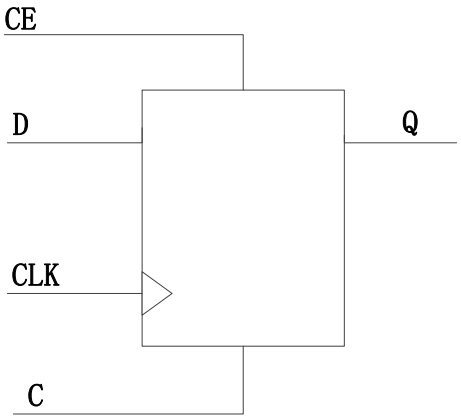


Figure 3-9 The Port Diagram of GTP_DFF_CE

The port description of the flip-flop GTP supported by the Logos2 family FPGAs is shown in Table 3-16.

Table 3-16 The Port Description of Flip-flop GTP

| Port Name | Input/Output | Description |
|---|---|---|
| CLK | Input | Input Clock |
| D | Input | Input signal of trigger |
| Q | Output | Output signal of trigger |
| C | Input | Asynchronous clear signal, sets output Q to 0 when C is at a high level |
| CE | Input | Active-high enable signal |
| P | Input | Asynchronous set signal, sets output Q to 1 when P is at a high level |
| R | Input | Synchronous clear signal, sets output Q to 0 when P is at a high level |
| S | Input | Synchronous set signal, sets output Q to 1 when S is at a high level |

Taking GTP_DFF_CE as an example, the parameter description of the flip-flop GTP is shown in Table 3-17.

Table 3-17 GTP_DFF_CE Parameter Description

| Parameter Name | Parameter Type | Setting Value | Function Description |
|---|---|---|---|
| GRS_EN | <string> | "TRUE", "FALSE" | Global reset signal enable |
| INIT | <binary> | 1'b0, 1'b1 | Initial value, asynchronously triggered by the global reset signal when GRS_EN = "TRUE" |

For details about other flip-flop GTP, please refer to ***"UG040007_Logos2 Family Products GTP User Guide".***

### 3.1.6 Latch GTP

The Logos2 family FPGAs supports the latch GTP as shown in Table 3-18.

Table 3-18 3.1.6 Latch GTP

| GTP | Description |
|---|---|
| GTP_DLATCH | Latch |
| GTP_DLATCH_C | Asynchronous clear latch |
| GTP_DLATCH_E | Latch with enable |
| GTP_DLATCH_CE | Asynchronous clear latch with enable |
| GTP_DLATCH_P | Asynchronous set latch |
| GTP_DLATCH_PE | Asynchronous set latch with enable |

Taking GTP_DLATCH_CE as an example, the port diagram of the latch GTP as shown in Figure 3-10.

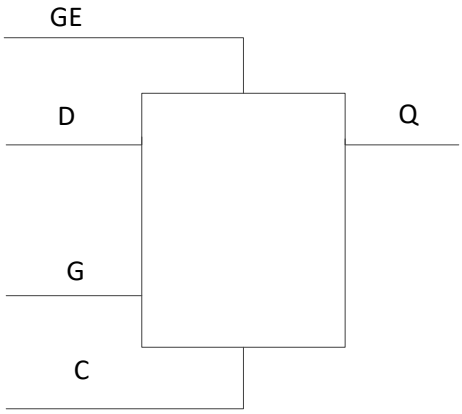


Figure 3-10 The Port Diagram of GTP_DLATCH_CE

The port description of the latch GTP supported by the Logos2 family FPGAs is shown in Table 3-19.

Table 3-19 Port Description of Latch GTP

| Port Name | Input/Output | Description |
|---|---|---|
| G | Input | Input level, the input signal is transmitted to the output signal when G is at high level |
| D | Input | Latch input signal |
| Q | Output | Latch output signal |
| P | Input | Asynchronous set signal, sets output Q to 1 when at a high level |
| GE | Input | Active-high enable signal |
| C | Input | Asynchronous clear signal, sets output Q to 0 when at a high level |

Taking GTP_DLATCH_CE as an example, the parameter description of the latch GTP is shown in Table 3-20.

Table 3-20 GTP_DLATCH_CE Parameter Description

| Parameter Name | Parameter Type | Setting Value | Function Description |
|---|---|---|---|
| GRS_EN | <string> | "TRUE", "FALSE" | Global reset signal enable |
| INIT | <binary> | 1'b0, 1'b1 | Initial value, asynchronously triggered by the global reset signal when GRS_EN = "TRUE" |

For details about other latch GTP, please refer to ***"UG040007_Logos2 Family Products GTP User Guide"***.

### 3.2 Using CLM

The use of CLMs can be facilitated through the Pango Design Suite software by Shenzhen Pango Microsystems Co., Ltd. CLM can be used in the following three ways.

- Generate distributed RAM IP using the IP Compiler tool embedded in Pango Design Suite, please refer to the "***UG061001 Distributed RAM IP User Guide***" document attached in the IP Compiler tool;
- Using CLM by instantiating GTP primitives in the design; an introduction to commonly used CLM primitives can be found in the Commonly Used GTP section, detailed information can be found by referring to the GTP manual. In the arch\vendor\pango\verilog\simulation subdirectory of the software installation directory, there are simulation models for each GTP, which can be used for reference during design;
- Users can add attributes in the design code to constrain the tool to map the corresponding instance to the CLM.

For the design purpose of data storage, distributed memory, block memory, registers, and other resources can be used. Parameters can be added to the code to inform the tool designer which resources are expected to be used to implement the design. For detailed description, please refer to the documentation "***ADS_Synthesis_User_Guide***" in the software installation directory.

Taking a 128×1 RAM as an example, see the following code for a scenario where the constraint object is a module.

```
module ipm_distributed_spram_v1_2_ram128x1
#(
parameter    ADDR_WIDTH      = 4 ,
parameter    DATA_WIDTH      = 4 ,
```

```
……
    )
    (
        input       wire    [DATA_WIDTH-1:0] wr_data ,
        input       wire    [ADDR_WIDTH-1:0] addr ,
        ……
    )/* synthesis syn_ramstyle = "select_ram" */;
     reg     [DATA_WIDTH-1:0]        mem [2**ADDR_WIDTH-1:0];
……
endmodule
```

Taking a 128×1 RAM as an example, see the following code for an application scenario where the constraint object is a RAM signal.

```
module ipm_distributed_spram_v1_2_ram128x1
#(
parameter    ADDR_WIDTH     = 4 ,
parameter    DATA_WIDTH     = 4 ,
……
    )
    (
        input       wire    [DATA_WIDTH-1:0]            wr_data ,
        input       wire    [ADDR_WIDTH-1:0]             addr ,
        ……
    );
    reg [DATA_WIDTH-1:0] mem [2**ADDR_WIDTH-1:0]/* synthesis syn_ramstyle = "select_ram" */;
……
endmodule
```

# Disclaimer

## Copyright Notice



## Disclaimer

1. This document only provides information in stages and may be updated at any time based on the actual situation of the products without further notice. The Company assumes no legal responsibility for any direct or indirect losses caused by improper use of this document.
2. This document is provided "as is" without any warranties, including but not limited to warranties of merchantability, fitness for a particular purpose, non-infringement, or any other warranties mentioned in proposals, specifications, or samples. This document does not grant any explicit or implied intellectual property usage license, whether by estoppel or otherwise.
3. The Company reserves the right to modify any documents related to its family products at any time without prior notice.