现代计算机组成原理

潘 明　潘 松 编著

科学出版社

第 4 章

CPU功能模块设计

4.1 8位CPU功能与结构

1．CPU的功能

（1）指令控制

（2）操作控制

（3）时序控制

（4）数据加工

4.1 8位CPU功能与结构

2．CPU的组成结构

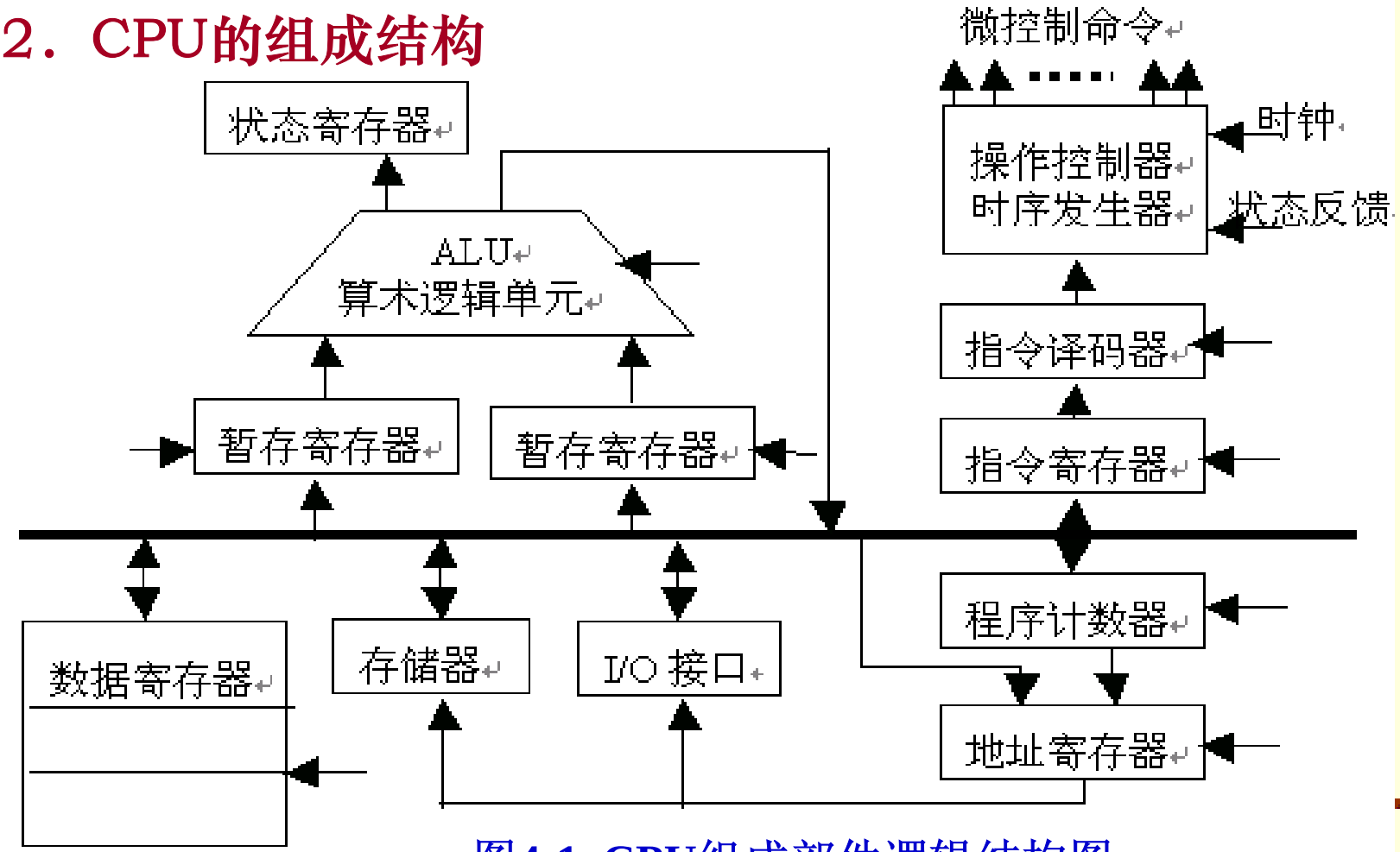

图4-1 CPU组成部件逻辑结构图

4.2 CPU中的基本部件

4.2.1. 算术逻辑单元（ALU）

算术逻辑单元（ALU）是计算机的核心部件之一，它能执行加法和减法等算术运算，也能执行“与”、“或”、“非”等逻辑运算。

算术逻辑单元的基本功能可以根据74LS181的功能用VHDL编辑而成。

表4-1　ALU181的运算功能

选择端	高电平作用数据		
	M=H	M=L 算术操作	
S3 S2 S1 S0	逻辑功能	Cn=L（无进位）	Cn=H（有进位）
0 0 0 0	$F = \overline{A}$	$F = A$	$F = A$加1
0 0 0 1	$F = \overline{A + B}$	$F = A + B$	$F=(A+B)$加1
0 0 1 0	$F = \overline{A}B$	$F = A + \overline{B}$	$F = A + \overline{B}$加1
0 0 1 1	$F = 0$	F减1（2的补码）	$F = 0$
0 1 0 0	$F = \overline{AB}$	$F = A$加$A\overline{B}$	$F = A$加$A\overline{B}$加1
0 1 0 1	$F = \overline{B}$	$F=(A+B)$加$A\overline{B}$	$F=(A+B)$加$A+\overline{B}1$
0 1 1 0	$F = A \oplus B$	$F = A$减B	$F = A$减B减1
0 1 1 1	$F = A\overline{B}$	$F = A + \overline{B}$	$F = (A + \overline{B})$减1
1 0 0 0	$F = \overline{A} + B$	$F = A$加AB	$F = A$加AB加1
1 0 0 1	$F = \overline{A \oplus B}$	$F = A$加B	$F = A$加B加1
1 0 1 0	$F = B$	$F =(A + \overline{B})$ 加AB	$F = (A + \overline{B})$加AB加1
1 0 1 1	$F = AB$	$F = AB$	$F = AB$减1
1 1 0 0	$F = 1$	$F = A$加$^*A$	$F = A$加A加1
1 1 0 1	$F = A + \overline{B}$	$F =(A + B)$ 加A	$F =(A + B)$ 加A加1
1 1 1 0	$F = A + B$	$F =(A + \overline{B})$ 加A	$F =(A + \overline{B})$ 加A加1
1 1 1 1	$F = A$	$F = A$	$F = A$减1

4.2 CPU中的基本部件

4.2.1. 算术逻辑单元（ALU）

【例4-1】

```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
USE IEEE.STD_LOGIC_UNSIGNED.ALL;
ENTITY ALU181 IS
PORT ( S  : IN  STD_LOGIC_VECTOR(3 DOWNTO 0 );
   A, B  : IN  STD_LOGIC_VECTOR(7 DOWNTO 0);
   F   : OUT STD_LOGIC_VECTOR(7 DOWNTO 0);
   COUT : OUT STD_LOGIC_VECTOR(3 DOWNTO 0);
   M, CN  : IN  STD_LOGIC;
   CO, FZ : OUT STD_LOGIC  );
END ALU181;
ARCHITECTURE behav OF ALU181 IS
SIGNAL A9, B9, F9 : STD_LOGIC_VECTOR(8 DOWNTO 0);
```

（接下页）

```
    BEGIN
    A9 <= '0' & A ;  B9 <= '0' & B ;
    PROCESS(M,CN,A9,B9)
    BEGIN
    CASE S  IS
    WHEN "0000"=>IF M='0' THEN F9<=A9 + CN      ; ELSE  F9<=NOT A9; END IF;
    WHEN "0001"=>IF M='0' THEN F9<=(A9 OR B9)+CN; ELSE  F9<=NOT(A9 OR B9);
END IF;
    WHEN "0010"=>IF M='0' THEN F9<=(A9 OR (NOT B9))+CN;
    ELSE F9<=(NOT A9) AND B9; END IF;
    WHEN  "0011"=>IF  M='0'  THEN  F9<=  "000000000"-CN  ;  ELSE  F9<="000000000";
END IF;
    WHEN "0100"=>IF M='0' THEN F9<=A9+(A9 AND NOT B9)+CN ;
     ELSE F9<=NOT (A9 AND B9); END IF;
    WHEN "0101"=>IF M='0' THEN F9<=(A9 OR B9)+(A9 AND NOT B9)+CN ;
     ELSE  F9<=NOT B9; END IF;
    WHEN "0110"=>IF M='0' THEN F9<=A9 -B9 - CN   ; ELSE F9<=A9 XOR B9; END IF;
    WHEN "0111"=>IF M='0' THEN F9<=(A9 AND (NOT B9))-CN ;
     ELSE F9<=A9 AND (NOT B9); END IF;
    WHEN "1000" =>IF M='0' THEN F9<=A9 + (A9 AND B9)+CN;
     ELSE  F9<=(NOT A9) OR B9;END IF;
    WHEN "1001" =>IF M='0' THEN F9<=A9 + B9 + CN; ELSE  F9<=NOT(A9 XOR B9);
```

（接下页）

```
    END IF;
    WHEN "1010" =>IF M='0' THEN F9<=(A9 OR (NOT B9))+(A9 AND B9)+CN ;
     ELSE  F9<=B9; END IF;
    WHEN "1011" =>IF M='0' THEN F9<=(A9 AND B9)- CN ; ELSE  F9<=A9 AND B9;
END IF;
    WHEN "1100" =>IF M='0' THEN F9<=A9 + A9 + CN; ELSE  F9<= "000000001"; END
IF;
    WHEN "1101" =>IF M='0' THEN F9<=(A9 OR B9)+A9 + CN ;
    ELSE  F9<=A9 OR (NOT B9); END IF;
    WHEN "1110" =>IF M='0' THEN F9<=(A9 OR(NOT B9))+A9+CN; ELSE F9<=A9 OR
B9;END IF;
    WHEN "1111" =>IF M='0' THEN F9<=A9-CN; ELSE  F9<=A9 ; END IF;
    WHEN OTHERS  =>F9<= "000000000" ;
    END CASE;
    IF (A9= B9) THEN FZ <= '0';END IF;
    END PROCESS;
    F<= F9(7 DOWNTO 0) ;   CO <= F9(8) ;
    COUT <= "0000" WHEN F9(8) = '0'   ELSE  "0001" ;
    END behav;
```

4.2 CPU中的基本部件

4.2.1. 算术逻辑单元（ALU）

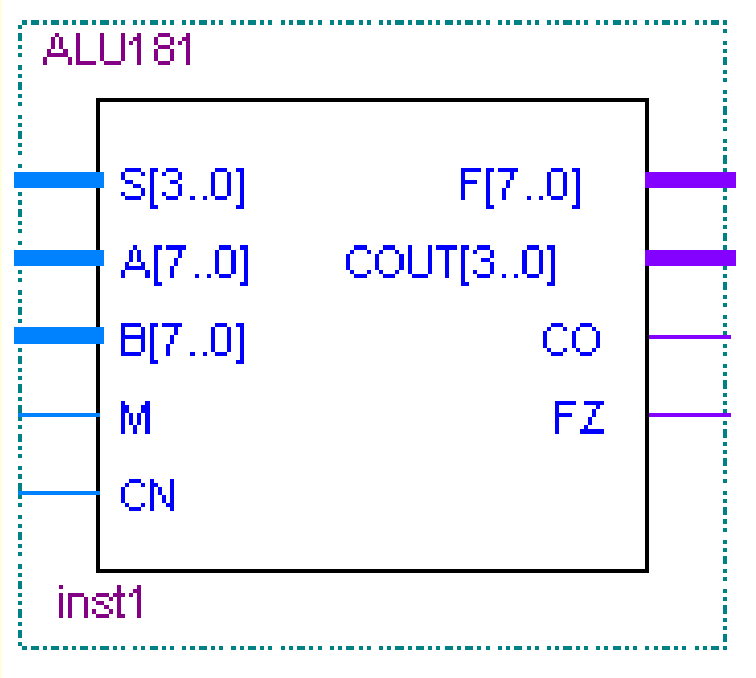

图4-2 ALU逻辑结构图

4.2 CPU中的基本部件

4.2.2 数据缓冲寄存器

【例4-2】

```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
ENTITY LATCH8 IS
    PORT (    GATE :  IN STD_LOGIC;
              DIN :  IN STD_LOGIC_VECTOR(7 DOWNTO 0);
              DOUT : OUT STD_LOGIC_VECTOR(7 DOWNTO 0) );
END LATCH8;
ARCHITECTURE behav OF LATCH8 IS
BEGIN
 PROCESS(GATE, DIN)
   BEGIN
    IF GATE = '1' THEN  DOUT <= DIN ;    END IF;
 END PROCESS;
END behav;
```

4.2 CPU中的基本部件

4.2.3 移位运算器

表4-2 移位运算器的功能

G	S1	S0	M	功 能
0	0	0	任意	保持
0	1	0	0	循环右移
0	1	0	1	带进位循环右移
0	0	1	0	循环左移
0	0	1	1	带进位循环左移
任意	1	1	任意	加载待移位数

4.2 CPU中的基本部件

4.2.3 移位运算器

【例4-3】

```
Library IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
ENTITY SHIFTER IS
PORT (CLK, M, C0 : IN STD_LOGIC;
S     : IN STD_LOGIC_VECTOR(1 DOWNTO 0);
D     : IN STD_LOGIC_VECTOR(7 DOWNTO 0);
QB    : OUT STD_LOGIC_VECTOR(7 DOWNTO 0);
CN        : OUT STD_LOGIC);
END ENTITY;
ARCHITECTURE BEHAV OF SHIFTER IS
SIGNAL ABC  : STD_LOGIC_VECTOR(2 DOWNTO 0);
SIGNAL REG  : STD_LOGIC_VECTOR(7 DOWNTO 0);
SIGNAL   CY : STD_LOGIC ;
```

（接下页）

```
    BEGIN
    PROCESS (CLK,ABC,C0)
    BEGIN
    IF CLK'EVENT AND CLK = '1' THEN
    CASE ABC IS
    WHEN "011" =>REG(0)<= C0; REG(7 DOWNTO 1) <= REG(6 DOWNTO 0); CY <=
REG(7);
    --带进位循--环左移
    WHEN "010" =>REG(0)<= REG(7);REG(7 DOWNTO 1)<=REG(6 DOWNTO 0);--循环
左移
    WHEN "100" =>REG(7)<= REG(0); REG(6 DOWNTO 0)<=REG(7 DOWNTO 1);--循
环右移
    WHEN "101" =>REG(7)<= C0 ; REG(6 DOWNTO 0) <= REG(7 DOWNTO 1); CY <=
REG(0);
    --带进位--循环右移
    WHEN "110" =>REG(7 DOWNTO 0) <=   D(7 DOWNTO 0);          --加载待移位数
    WHEN "111" =>REG(7 DOWNTO 0) <=   D(7 DOWNTO 0);            --加载待移位数
    WHEN OTHERS => REG <= REG ;  CY <= CY ;                    --保持
    END CASE;
    END IF;
    END PROCESS;
    ABC <= S & M; QB(7 DOWNTO 0) <= REG(7 DOWNTO 0); CN <= CY;
    END BEHAV;
```

4.2 CPU中的基本部件

4.2.4 程序存储器与数据存储器

1. 工作原理

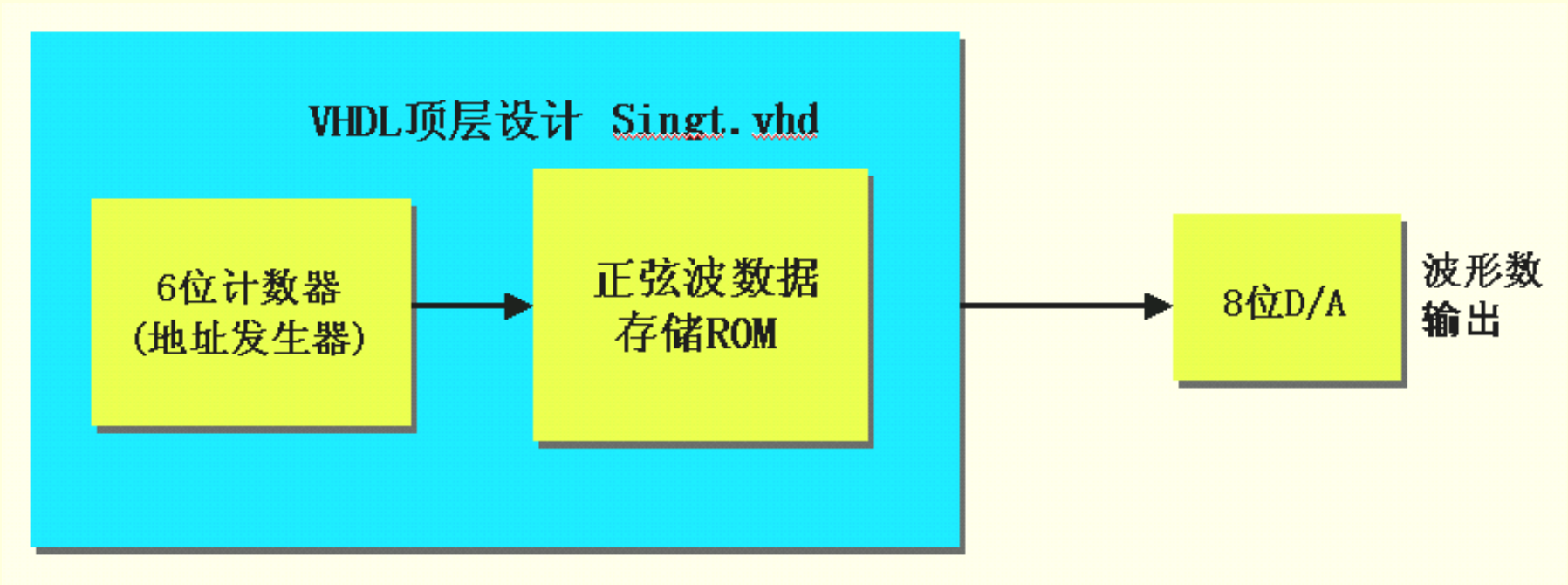

图4-3 正弦信号发生器结构框图

4.2 CPU中的基本部件

4.2.4 程序存储器与数据存储器

2. 定制初始化数据文件

【例4-4】

```
WIDTH = 8;
DEPTH = 64;
ADDRESS_RADIX = HEX;
DATA_RADIX = HEX;
CONTENT BEGIN
0       :       FF;
1       :       FE;
2       :       FC;
3       :       F9;
4       :       F5;
…（数据略去）
3E       :       FE;
3F       :       FF;
END;
```

4.2 CPU中的基本部件

4.2.4 程序存储器与数据存储器

2. 定制初始化数据文件

Addr	+0	+1	+2	+3	+4	+5	+6	+7
0	255	254	252	249	245	239	233	225
8	217	207	197	186	174	162	150	137
16	124	112	99	87	75	64	53	43
24	34	26	19	13	8	4	1	0
32	0	1	4	8	13	19	26	34
40	43	53	64	75	87	99	112	124
48	137	150	162	174	186	197	207	217
56	225	233	239	245	249	252	254	255

图4-4 将波形数据填入mif文件表中

4.2 CPU中的基本部件

4.2.4 程序存储器与数据存储器

2. 定制初始化数据文件

图4-5 ASM格式
建hex文件

4.2 CPU中的基本部件

4.2.4 程序存储器与数据存储器

2. 定制初始化数据文件

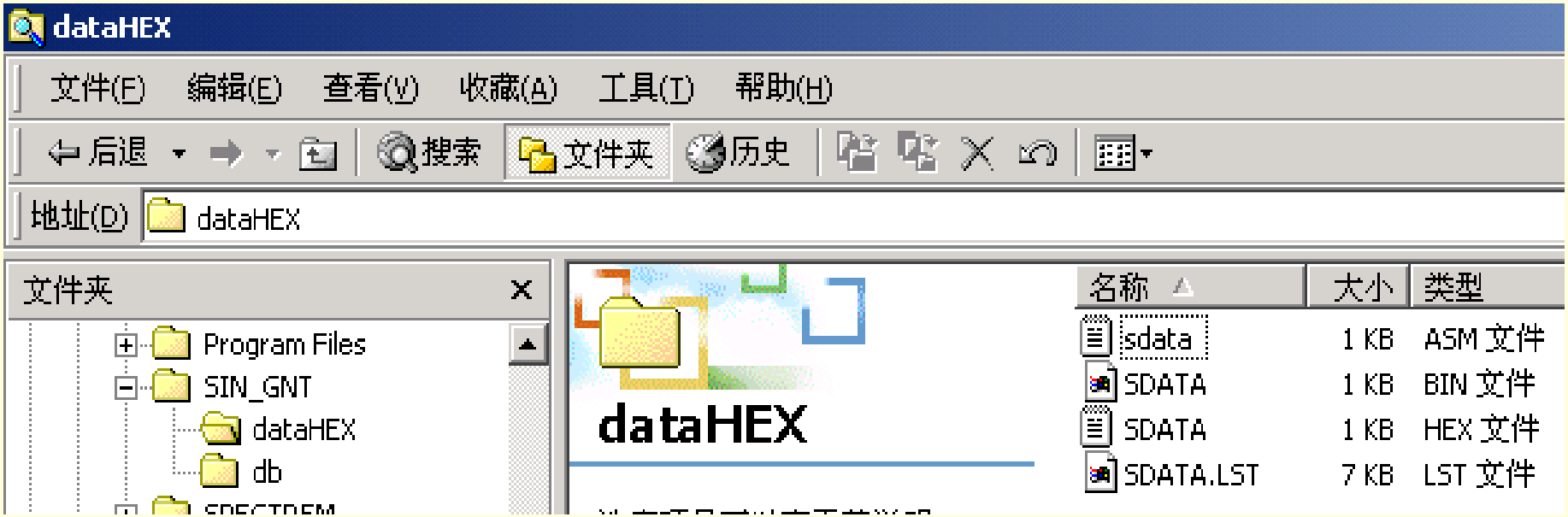

图4-6 sdata.hex文件的放置路径

4.2 CPU中的基本部件

4.2.4 程序存储器与数据存储器

3. 定制LPM_ROM元件

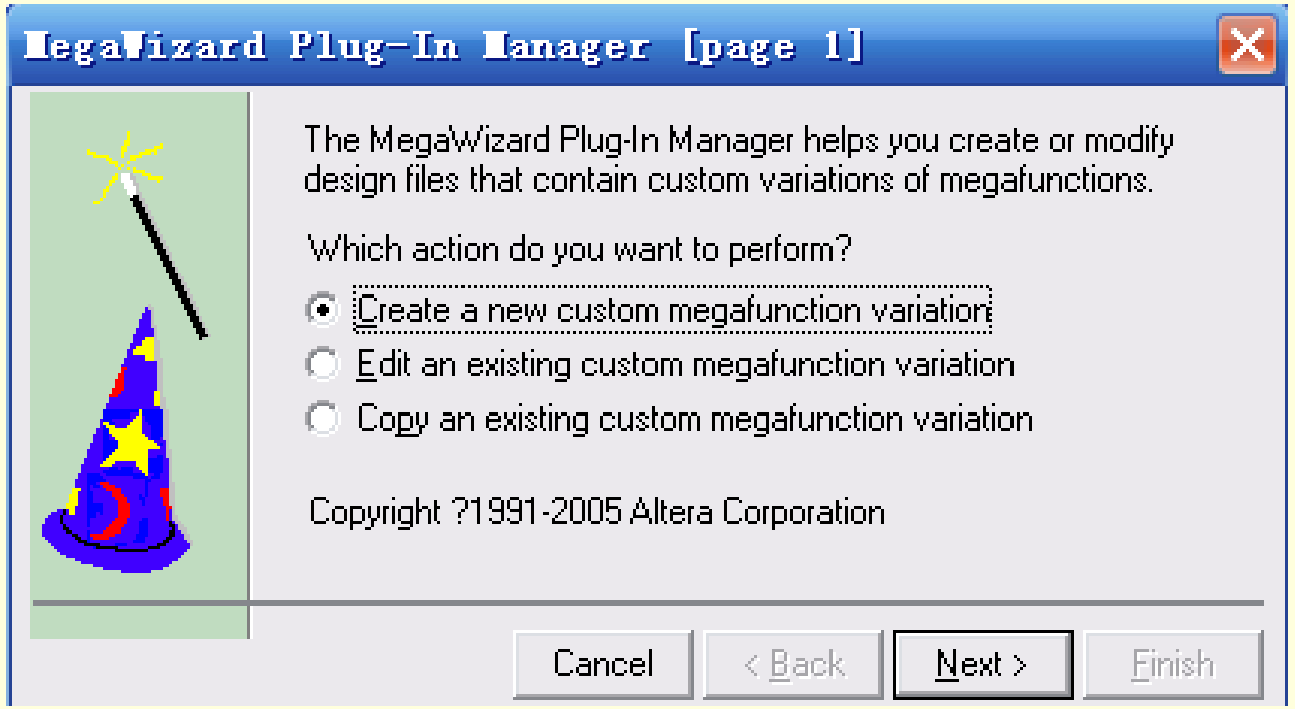

图4-7 定制新的宏功能块

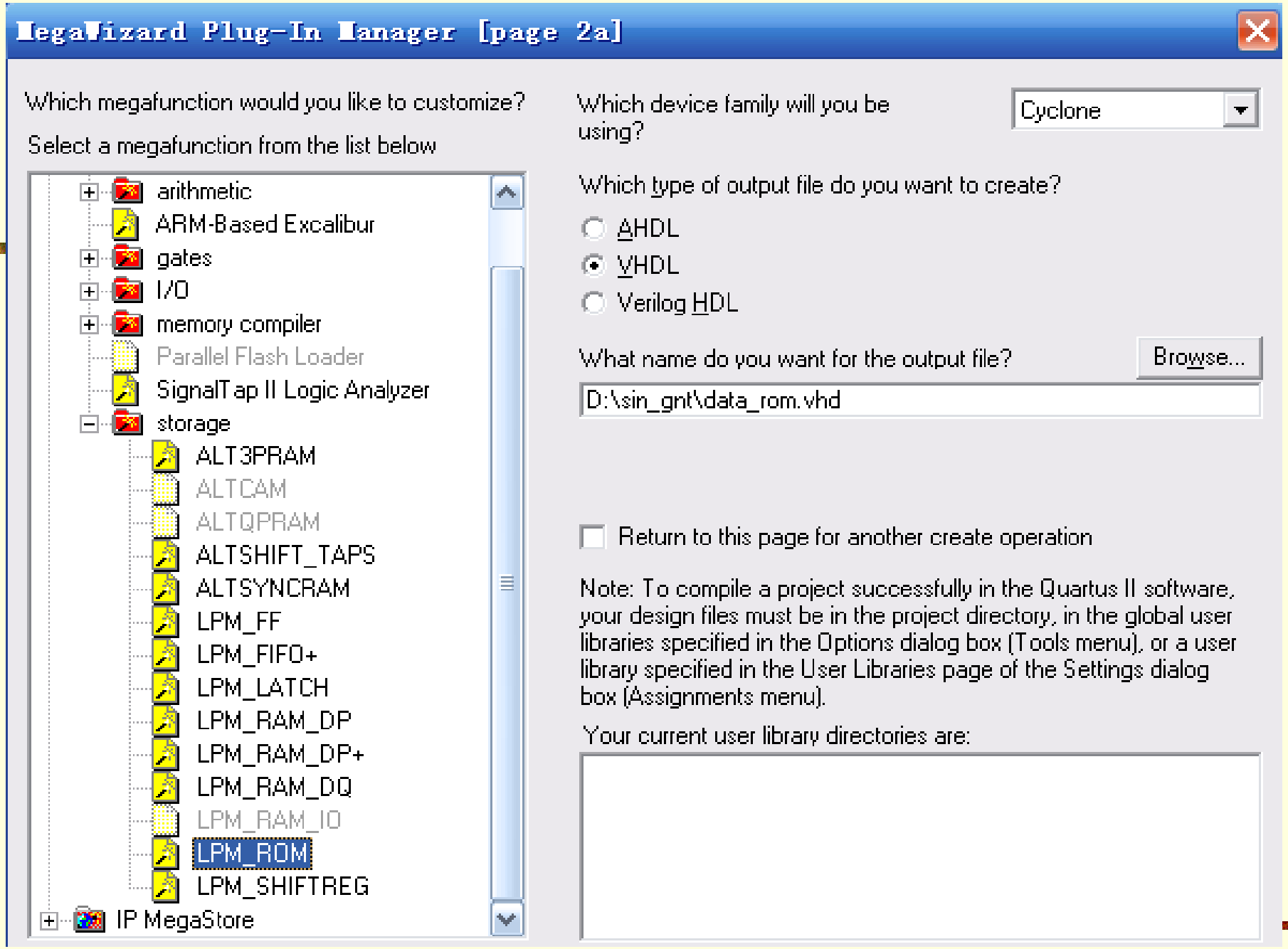

图4-8　LPM宏功能块设定

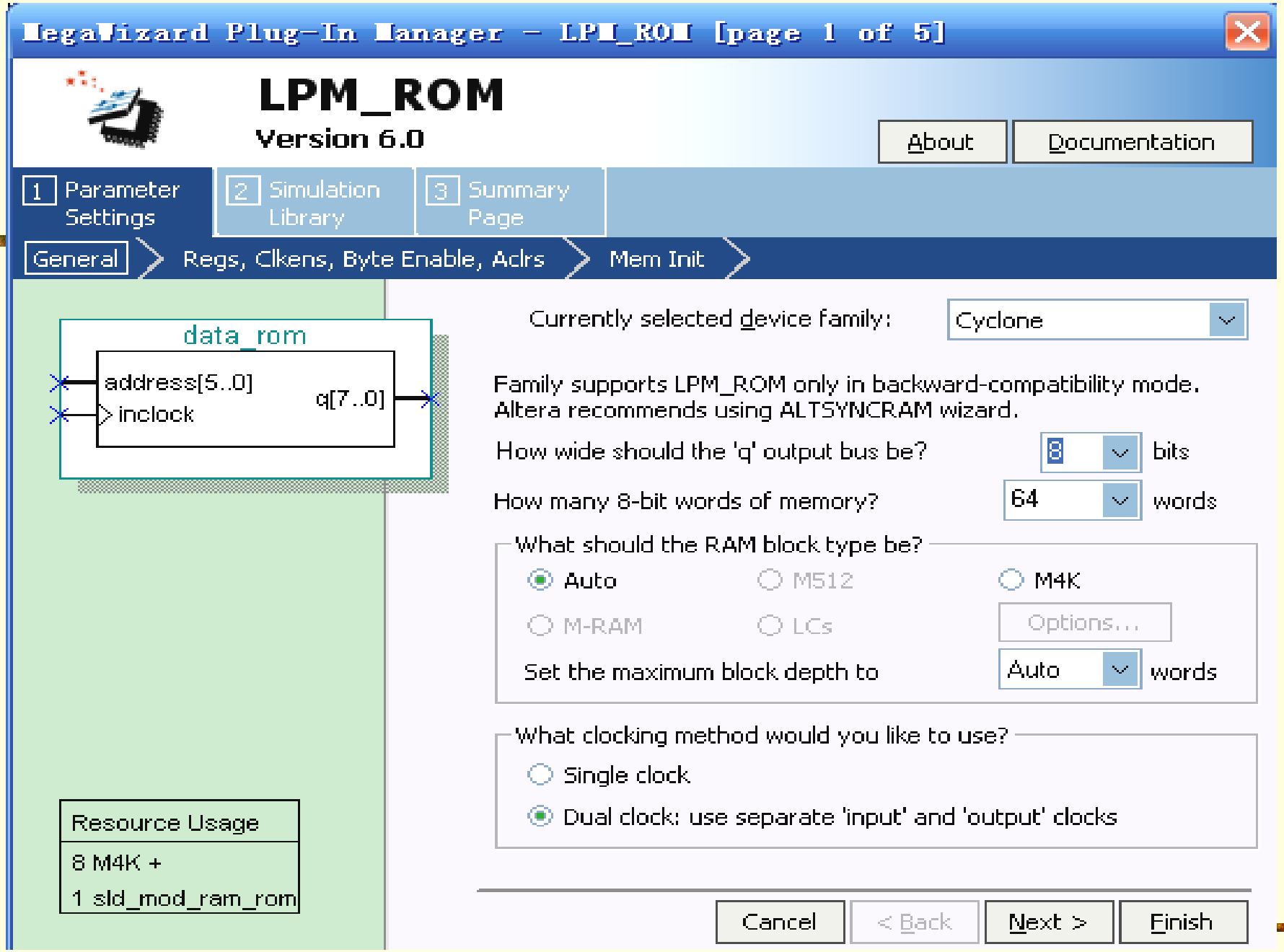

图4-9 选择data_rom模块数据线和地址线宽度

4.2 CPU中的基本部件

4.2.4 程序存储器与数据存储器

3. 定制LPM_ROM元件

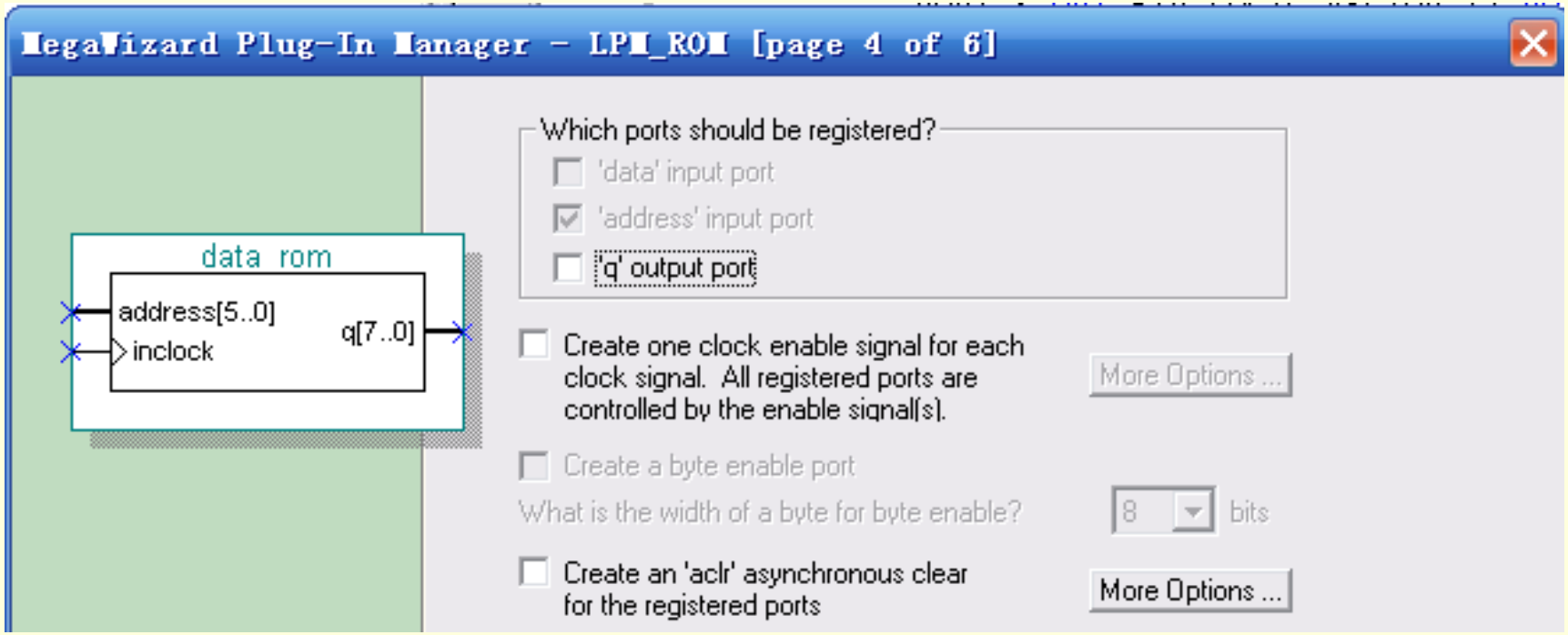

图4-10 选择地址锁存信号inclock

4.2 CPU中的基本部件

4.2.4 程序存储器与数据存储器

3. 定制LPM_ROM元件

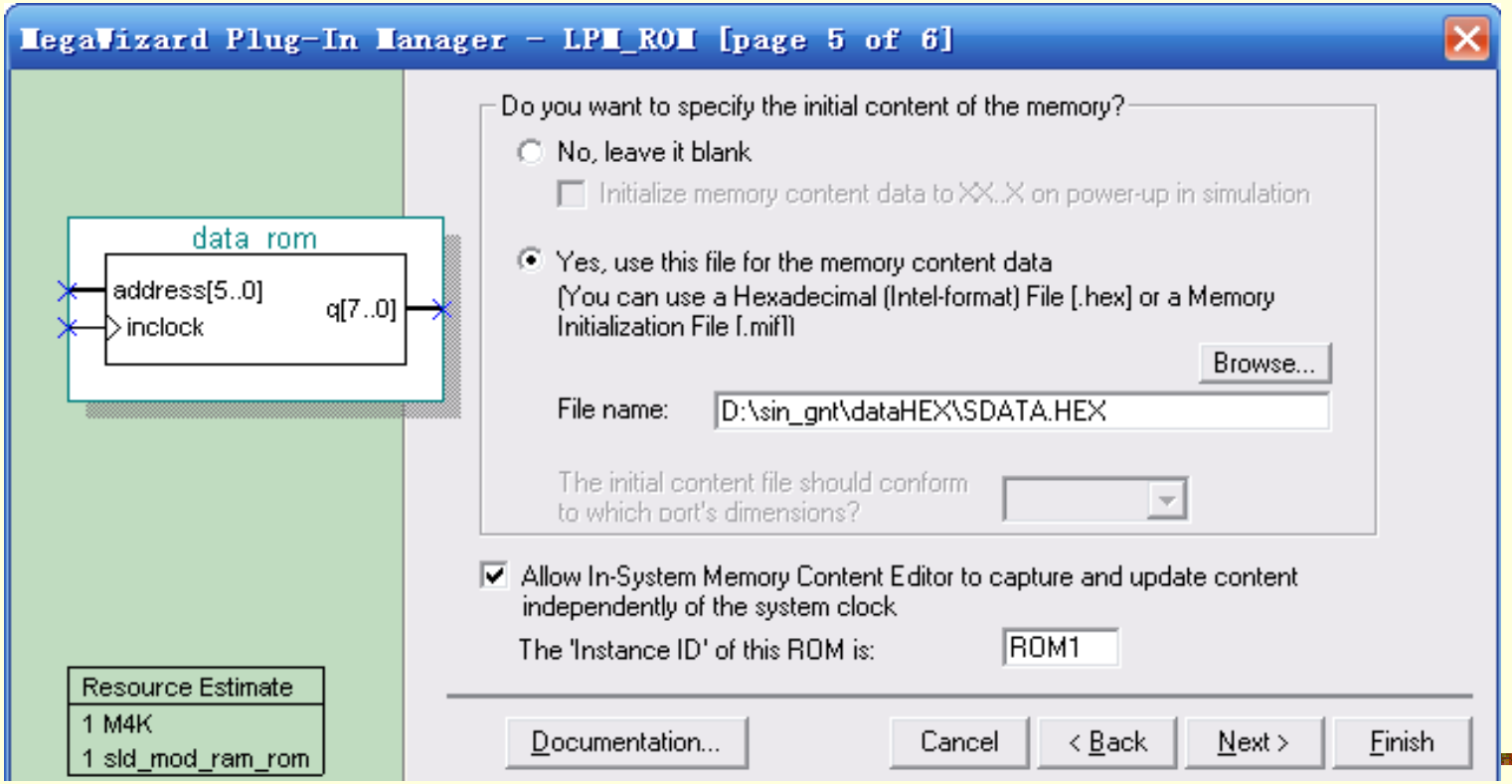

图4-11 调入ROM初始化数据文件并选择在系统读写功能

【例4-5】

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;
LIBRARY altera_mf;
USE altera_mf.altera_mf_components.all;          --使用宏功能库中的所有元件
ENTITY data_rom IS
        PORT (address           : IN STD_LOGIC_VECTOR (5 DOWNTO 0);
              inclock           : IN STD_LOGIC ;
              q                 : OUT STD_LOGIC_VECTOR (7 DOWNTO 0) );
END data_rom;
ARCHITECTURE SYN OF data_rom IS
        SIGNAL sub_wire0        : STD_LOGIC_VECTOR (7 DOWNTO 0);
        COMPONENT altsyncram                --例化altsyncram元件，调用了LPM模块altsyncram
        GENERIC (                           --参数传递语句
              intended_device_family    : STRING;   --类属参量数据类型定义
              width_a           : NATURAL;          widthad_a
        : NATURAL;
              numwords_a            : NATURAL;          operation_mode
              : STRING;
              outdata_reg_a         : STRING;       address_aclr_a
        : STRING;
              outdata_aclr_a        : STRING;       width_byteena_a
        : NATURAL;
```

（接下页）

```
                init_file              : STRING;                    lpm_hint : STRING;
                lpm_type                        : STRING            );
        PORT (  clock0     : IN STD_LOGIC ;              --altsyncram元件接口声明
                           address_a: IN STD_LOGIC_VECTOR (5 DOWNTO 0);
                           q_a        : OUT STD_LOGIC_VECTOR (7 DOWNTO 0) );
        END COMPONENT;
BEGIN
        q    <= sub_wire0(7 DOWNTO 0);
        altsyncram_component : altsyncram
        GENERIC MAP ( intended_device_family => "Cyclone", --参数传递映射
                width_a => 8,                                --数据线宽度8
                widthad_a => 6,                               --地址线宽度6
                numwords_a => 64,                               --数据数量64
                operation_mode => "ROM",                          --LPM模式ROM
                outdata_reg_a => "UNREGISTERED",                        --输出无锁存
                address_aclr_a => "NONE",                       --无异步地址清0
                outdata_aclr_a => "NONE",                     --无输出锁存异步清0
                width_byteena_a => 1,                      -- byteena_a输入口宽度1
                init_file => "./dataHEX/SDATA.hex", --ROM初始化数据文件，此处已修改过
                lpm_hint => "ENABLE_RUNTIME_MOD=YES,
INSTANCE_NAME=NONE",
                lpm_type => "altsyncram" )                        --LPM类型
        PORT MAP (clock0 => inclock, address_a => address,q_a => sub_wire0 );
END SYN;  SYN;
```

4.2 CPU中的基本部件

4.2.4 程序存储器与数据存储器

4. 完成顶层设计

【例4-6】 正弦信号发生器顶层设计

```
LIBRARY IEEE;  --正弦信号发生器源文件
USE IEEE.STD_LOGIC_1164.ALL;
USE IEEE.STD_LOGIC_UNSIGNED.ALL;
ENTITY SINGT IS
   PORT ( CLK  : IN STD_LOGIC;            --信号源时钟
          DOUT : OUT STD_LOGIC_VECTOR (7 DOWNTO 0) );--8位波形数据输出
END;
ARCHITECTURE DACC OF SINGT IS
COMPONENT data_rom --调用波形数据存储器LPM_ROM文件：data_rom.vhd声明
  PORT(address : IN STD_LOGIC_VECTOR (5 DOWNTO 0);--6位地址信号
             inclock : IN STD_LOGIC ;--地址锁存时钟
                   q : OUT STD_LOGIC_VECTOR (7 DOWNTO 0)            );
END COMPONENT;
```

（接下页）

```
SIGNAL Q1 : STD_LOGIC_VECTOR (5 DOWNTO 0); --设定内部节点作为地址计数器
   BEGIN
PROCESS(CLK )                    --LPM_ROM地址发生器进程
   BEGIN
IF CLK'EVENT AND CLK = '1' THEN  Q1<=Q1+1; --Q1作为地址发生器计数器
END IF;
END PROCESS;
u1 : data_rom PORT MAP(address=>Q1, q => DOUT,inclock=>CLK);--例化
END;
```

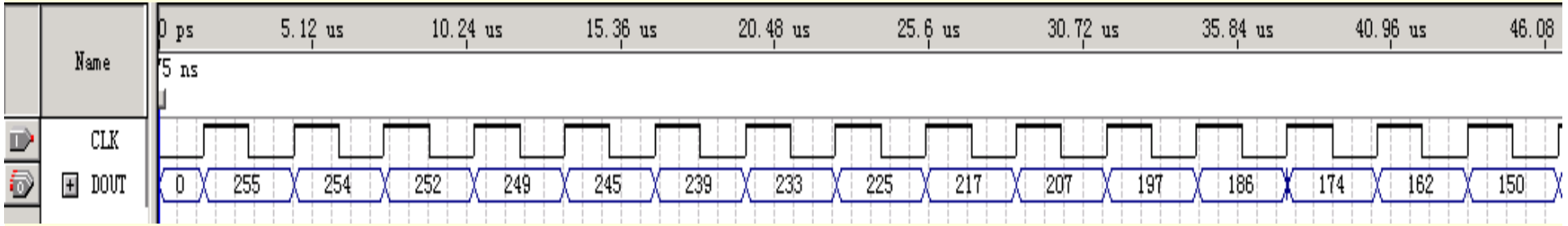

图4-13 仿真波形输出

4.2 CPU中的基本部件

4.2.4 程序存储器与数据存储器

4. 完成顶层设计

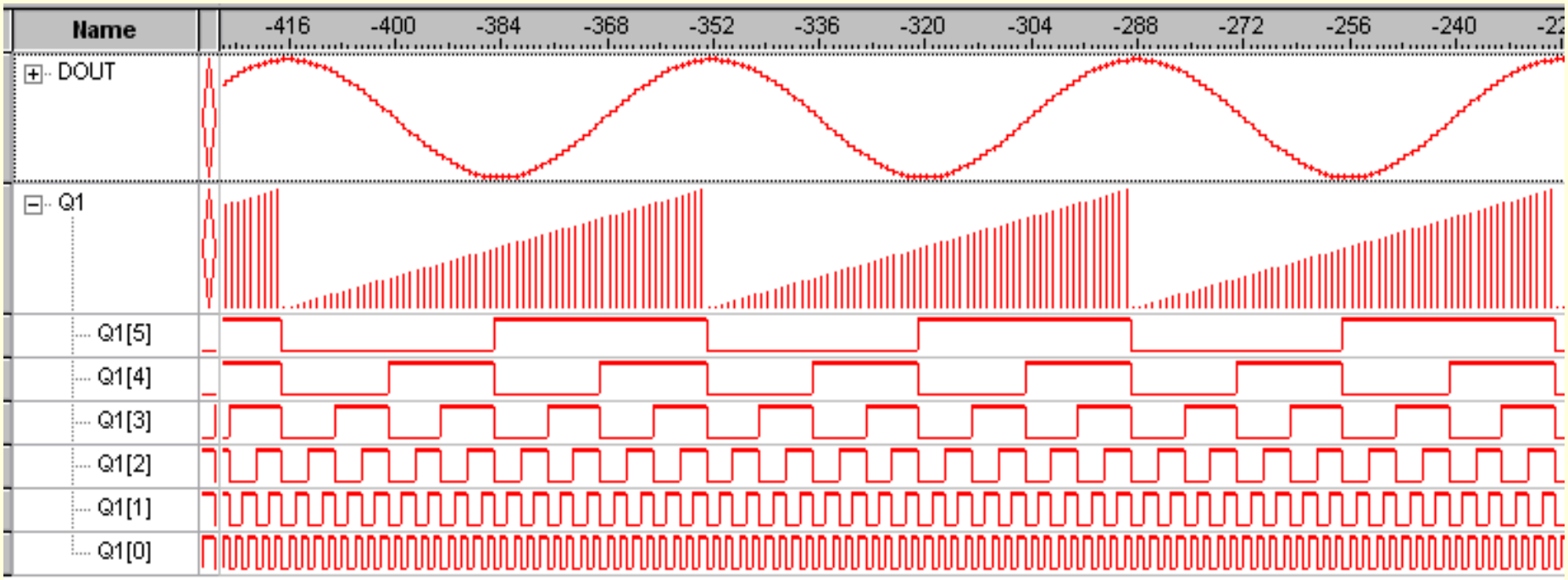

图4-14 嵌入式逻辑分析仪获得的波形

4.2 CPU中的基本部件

4.2.4 程序存储器与数据存储器

5. 微程序的LPM_ROM

图4-15 LPM_ROM的结构图

4.2 CPU中的基本部件

4.2.4 程序存储器与数据存储器

5. 微程序的LPM_ROM

Addr	+0	+1	+2	+3	+4	+5	+6	+7
00	018108	00ED82	00C050	00E004	00B005	01A206	959A01	00E00F
08	00ED8A	00ED8C	00A008	008001	062009	062009	070A08	038201
10	001001	00ED83	00ED87	00ED99	00ED9C	31821D	31821F	318221
18	318223	00E01A	00A01B	070A01	00D181	21881E	019801	298820
20	019801	118822	019801	198824	019801	018110	000002	000003
28	000004	000005	000006	000007	000008	000009	00000A	00000B
30	00000C	00000D	00000E	00000F	000010	000011	000012	000013
38	000014	000015	000016	000017	000018	000019	00001A	00001C

图4-16 rom_a.mif中的数据

4.2 CPU中的基本部件

4.2.4 程序存储器与数据存储器

6. LPM_RAM的调用和结构

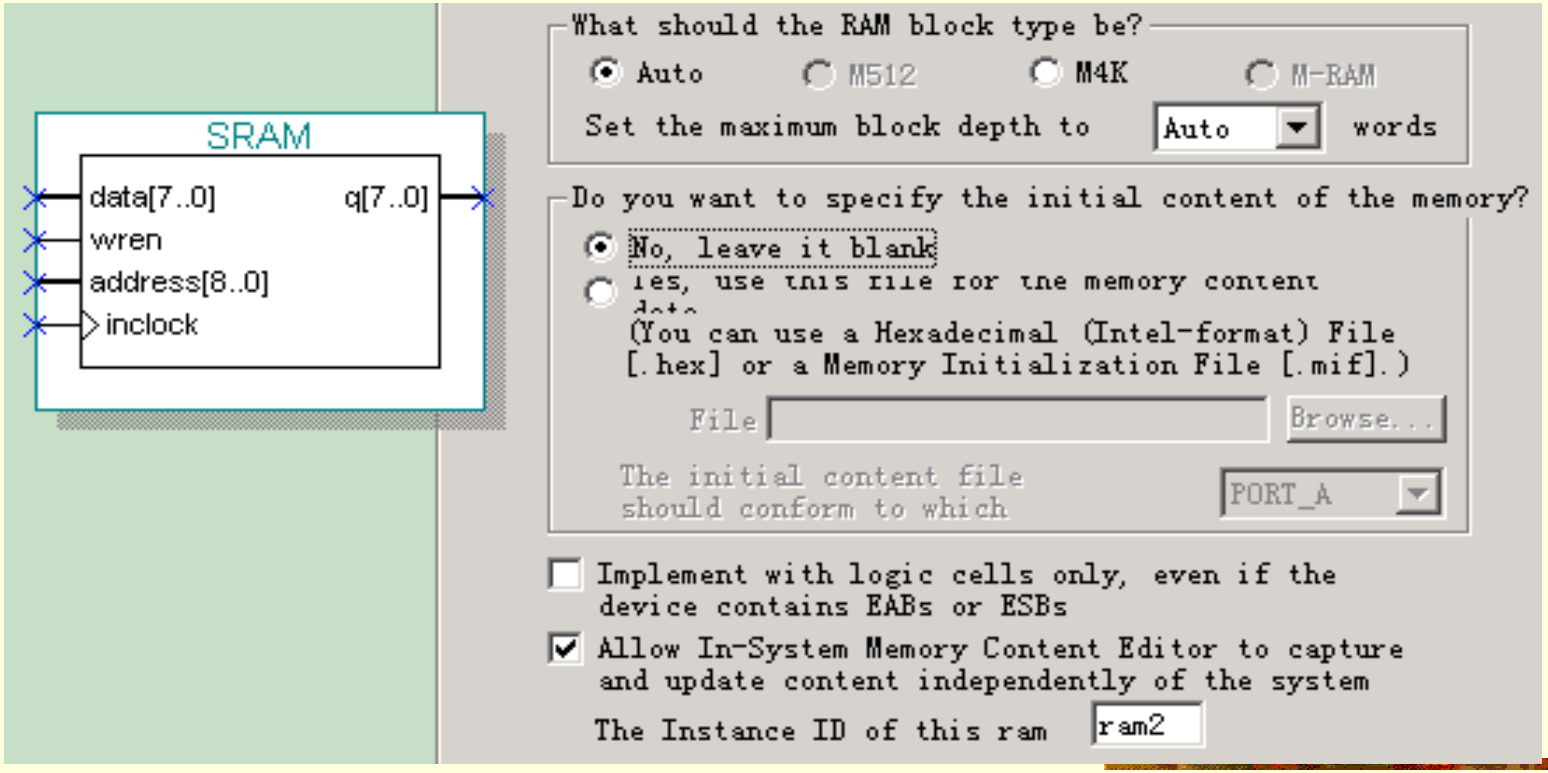

图4-17 编辑定制RAM

4.2 CPU中的基本部件

4.2.4 程序存储器与数据存储器

6. LPM_RAM的调用和结构

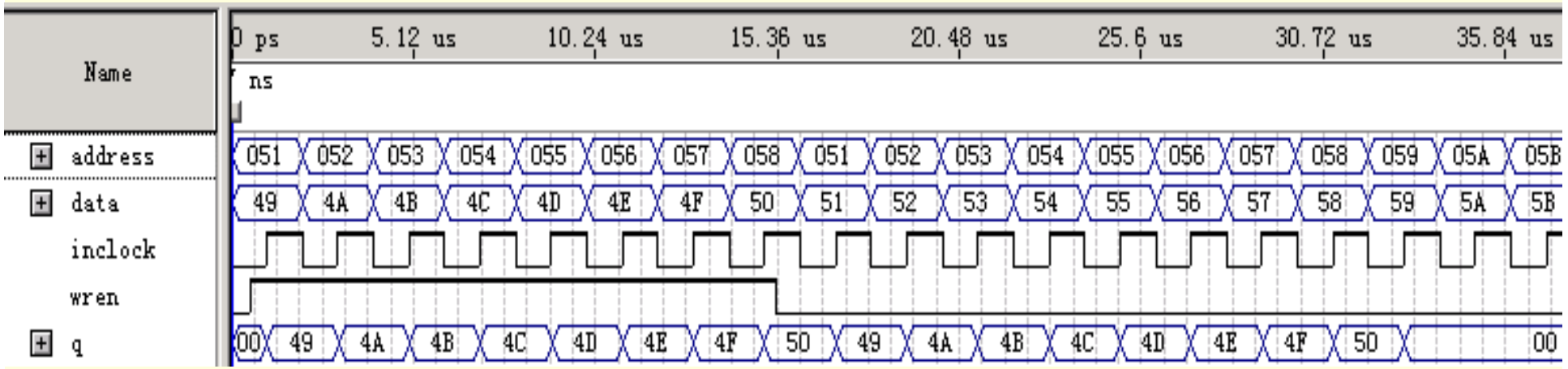

图4-18 LPM_RAM的仿真波形

4.2 CPU中的基本部件

4.2.4 程序存储器与数据存储器

6. LPM_RAM的调用和结构

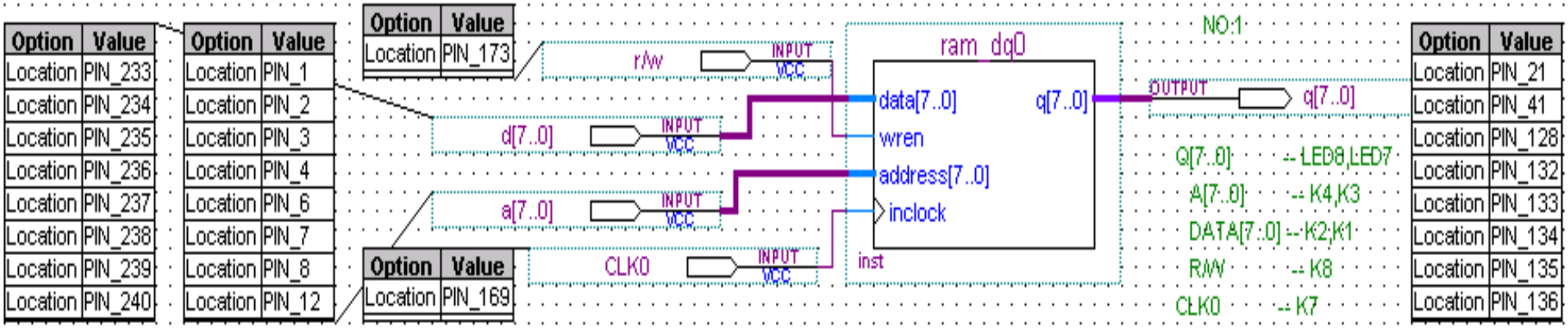

图4-19 8位LPM_RAM原理图模块

4.2 CPU中的基本部件

4.2.5 先进先出存储器FIFO

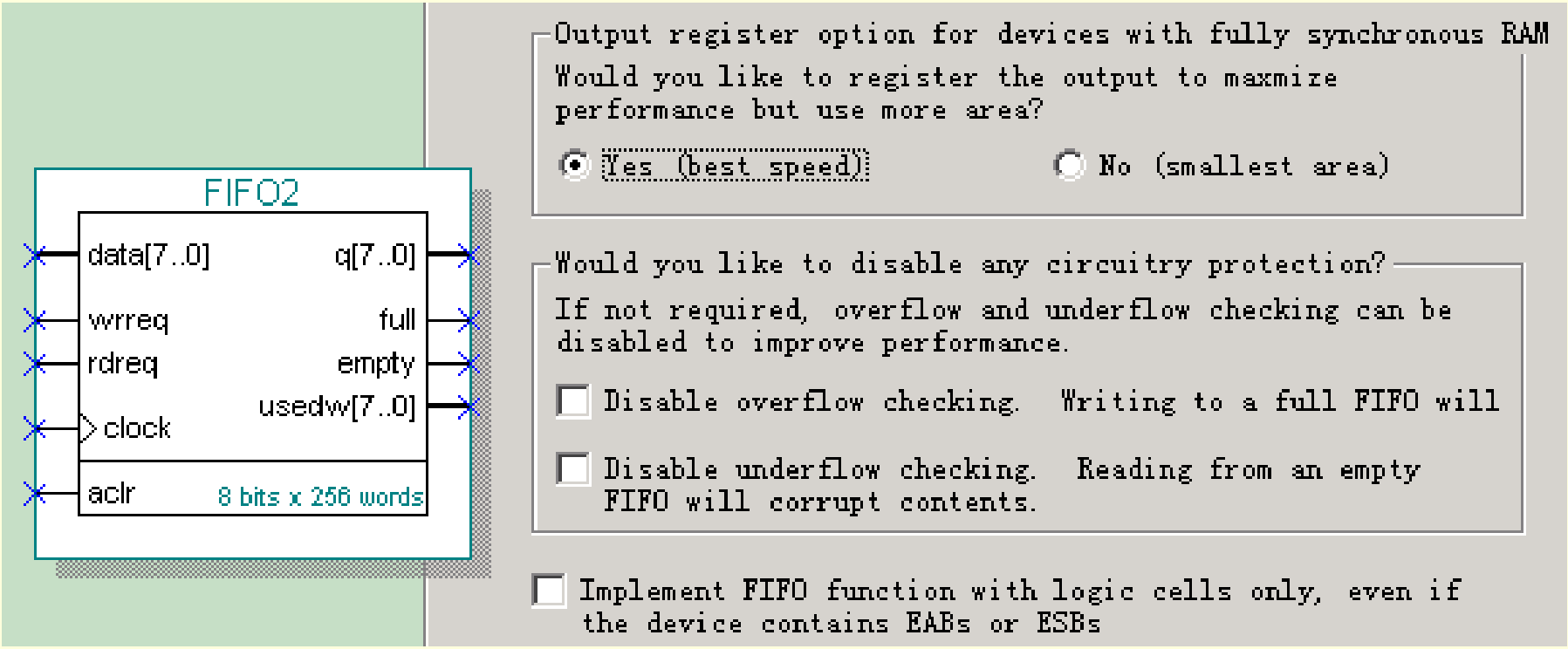

图4-20 FIFO编辑窗

4.2 CPU中的基本部件

4.2.5 先进先出存储器FIFO

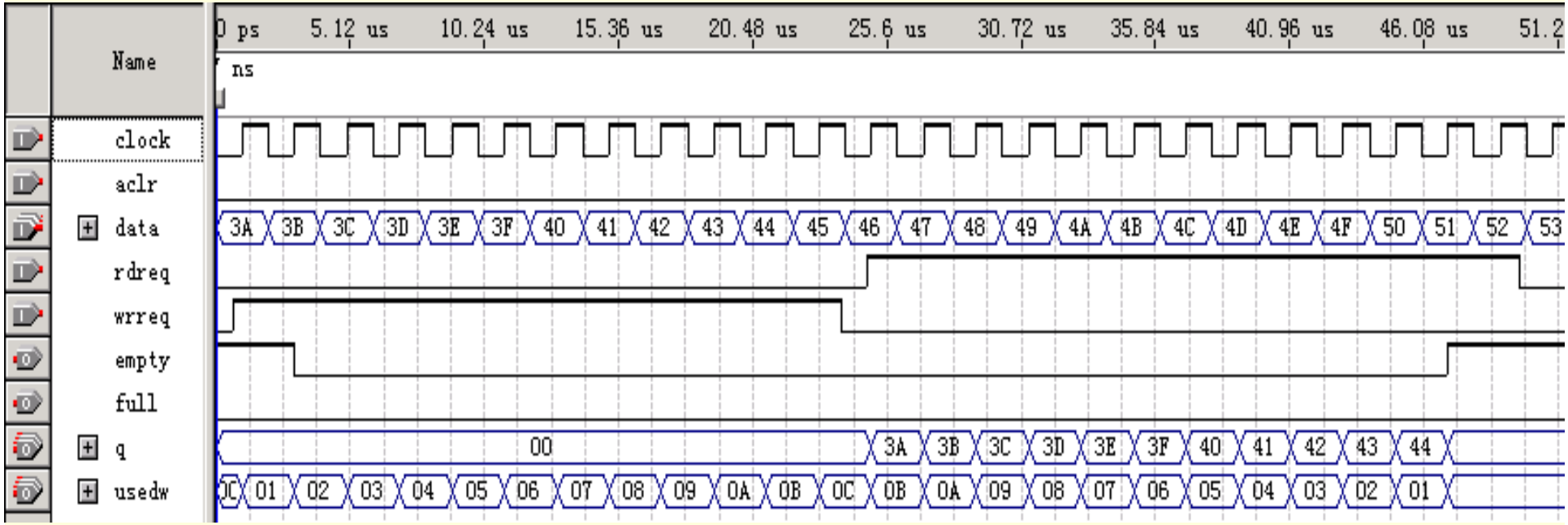

图4-21 FIFO的仿真波形

4.2 CPU中的基本部件

4.2.6 流水线乘法累加器

（1）用VHDL设计16位加法器

【例4-7】

```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
USE IEEE.STD_LOGIC_UNSIGNED.ALL;
ENTITY ADDER16B IS
   PORT ( CIN : IN STD_LOGIC;
          A，B : IN STD_LOGIC_VECTOR(15 DOWNTO 0);
          S : OUT STD_LOGIC_VECTOR(15 DOWNTO 0);
        COUT : OUT STD_LOGIC    );
END ADDER16B;
ARCHITECTURE behav OF ADDER16B IS
   SIGNAL SINT : STD_LOGIC_VECTOR(16 DOWNTO 0);
              SIGNAL AA,BB : STD_LOGIC_VECTOR(16 DOWNTO 0);
BEGIN
              AA<='0'&A;            BB<='0'& B;
   SINT <= AA + BB + CIN;   S <= SINT(15 DOWNTO 0);   COUT <= SINT(4);
END behav;
```

4.2 CPU中的基本部件

4.2.6 流水线乘法累加器

（2）顶层原理图文件设计

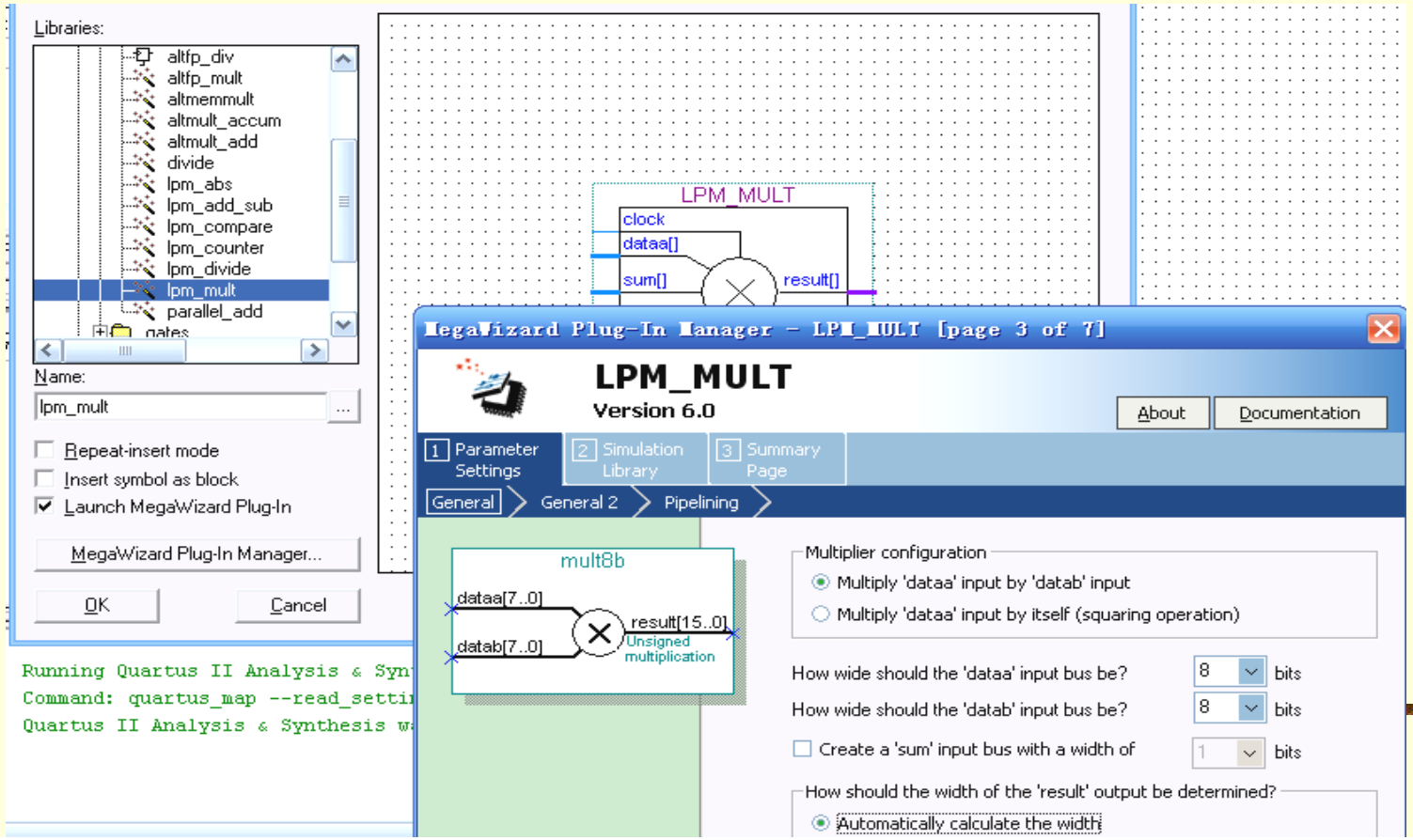

图4-22 在原理图编辑窗加入LPM元件

4.2 CPU中的基本部件

4.2.6 流水线乘法累加器

（2）顶层原理图文件设计

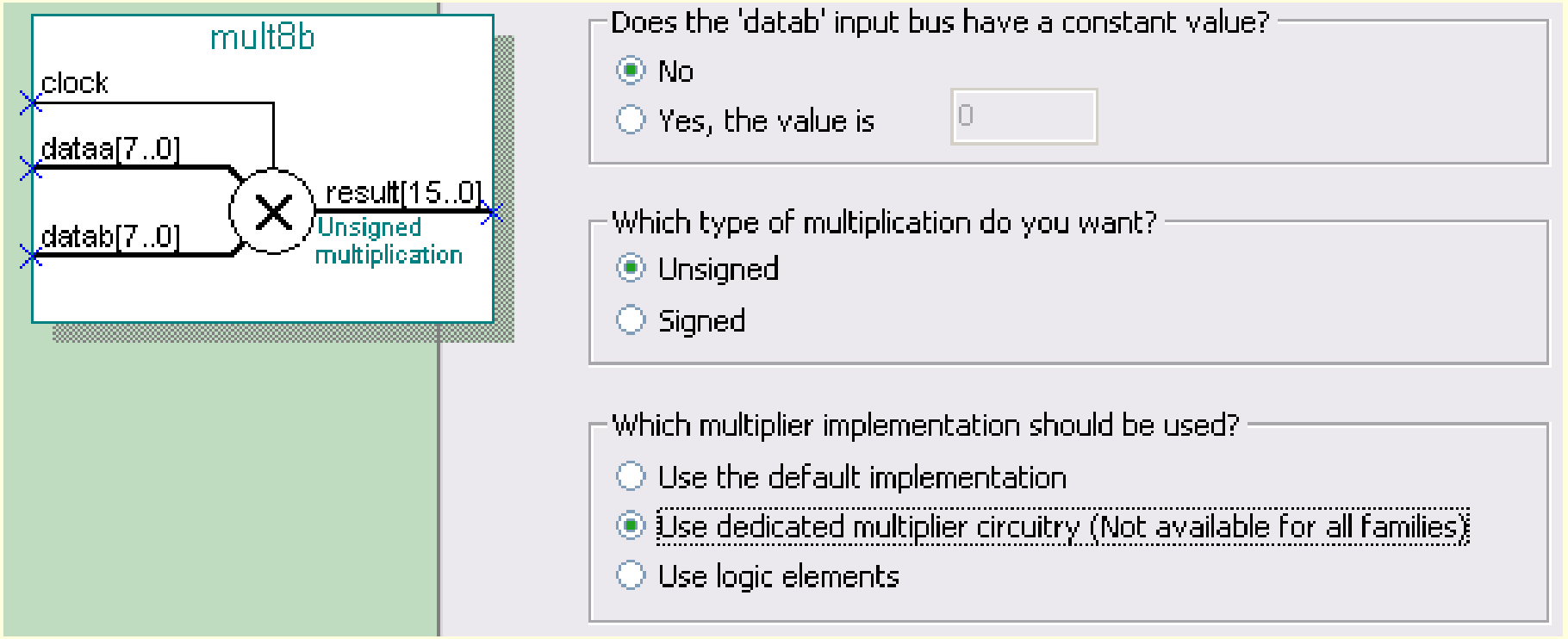

图4-23 将LPM乘法器设置为流水线工作方式

4.2 CPU中的基本部件

4.2.6 流水线乘法累加器

（3）仿真

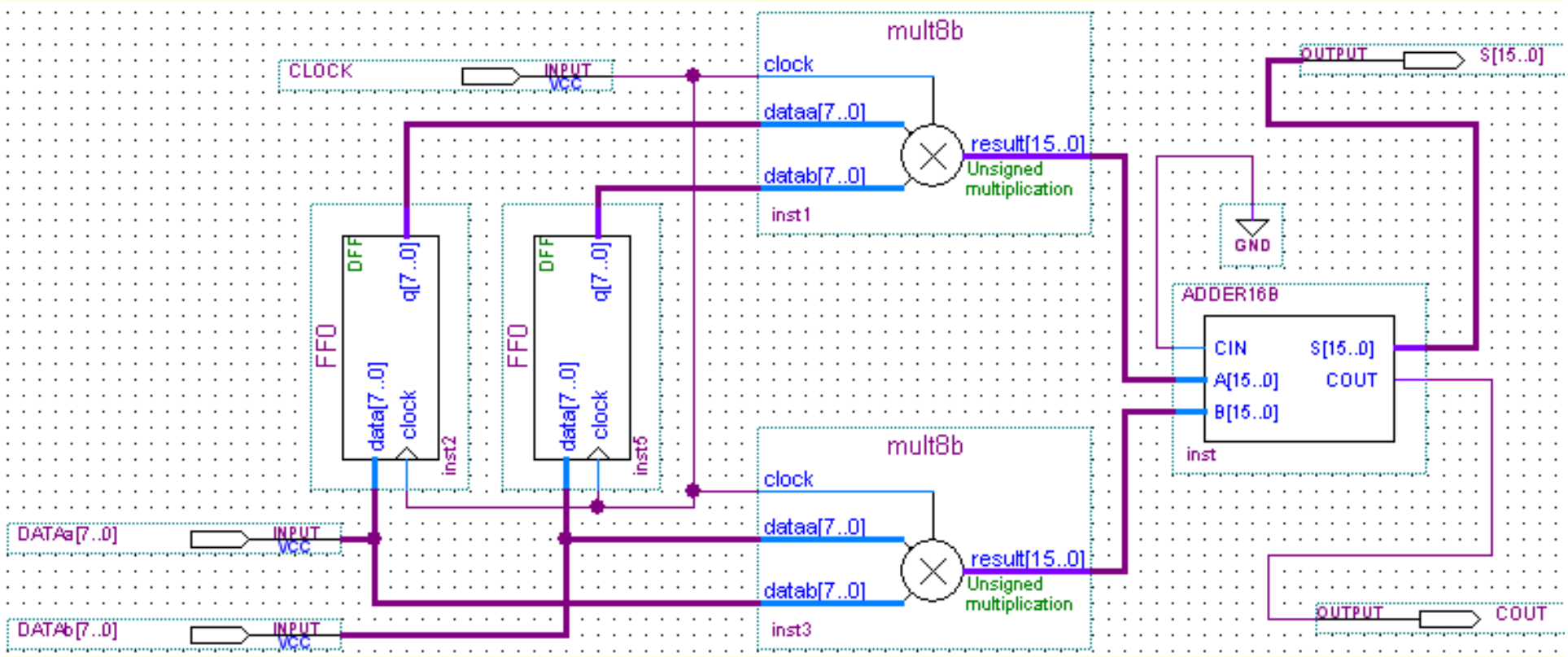

图4-24 乘法累加器电路

4.2 CPU中的基本部件

4.2.6 流水线乘法累加器

（3）仿真

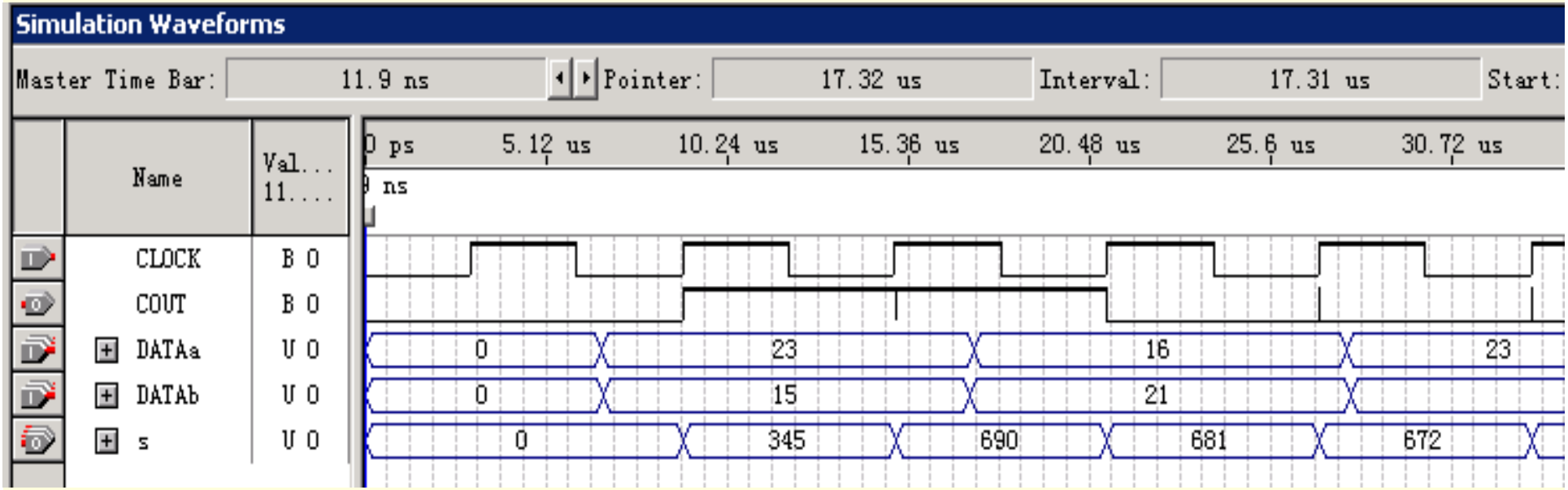

图4-25 muladd工程仿真波形

4.2 CPU中的基本部件

4.2.7 程序计数器与地址寄存器

1. 程序计数器

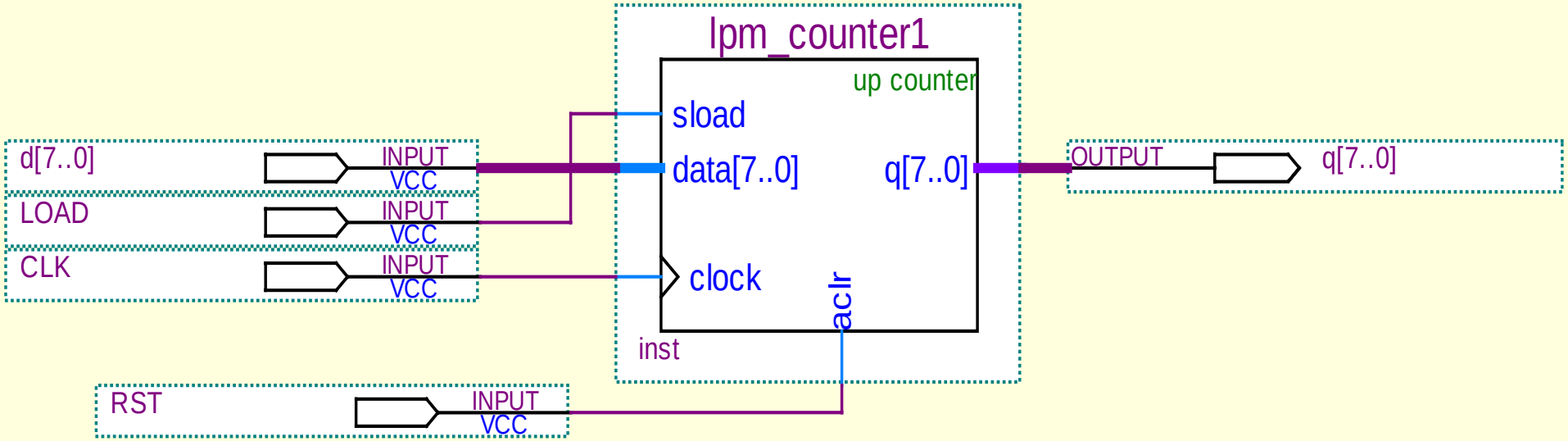

图4-26 程序计数器结构图

4.2 CPU中的基本部件

4.2.7 程序计数器与地址寄存器

2. 地址寄存器

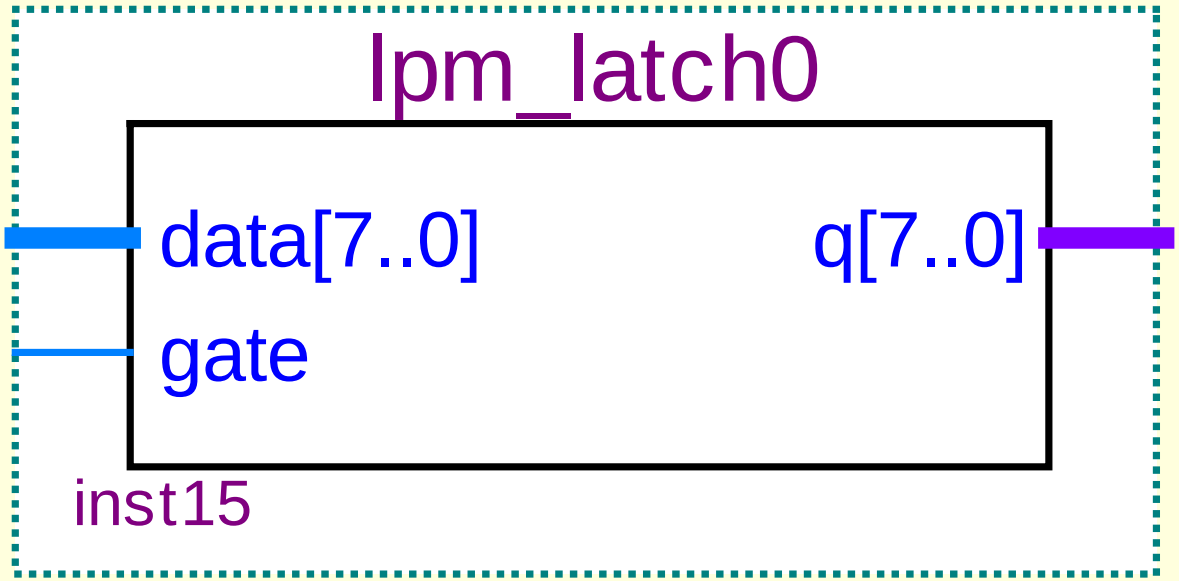

图4-27 地址寄存器结构图

4.2 CPU中的基本部件

4.2.8 指令寄存器

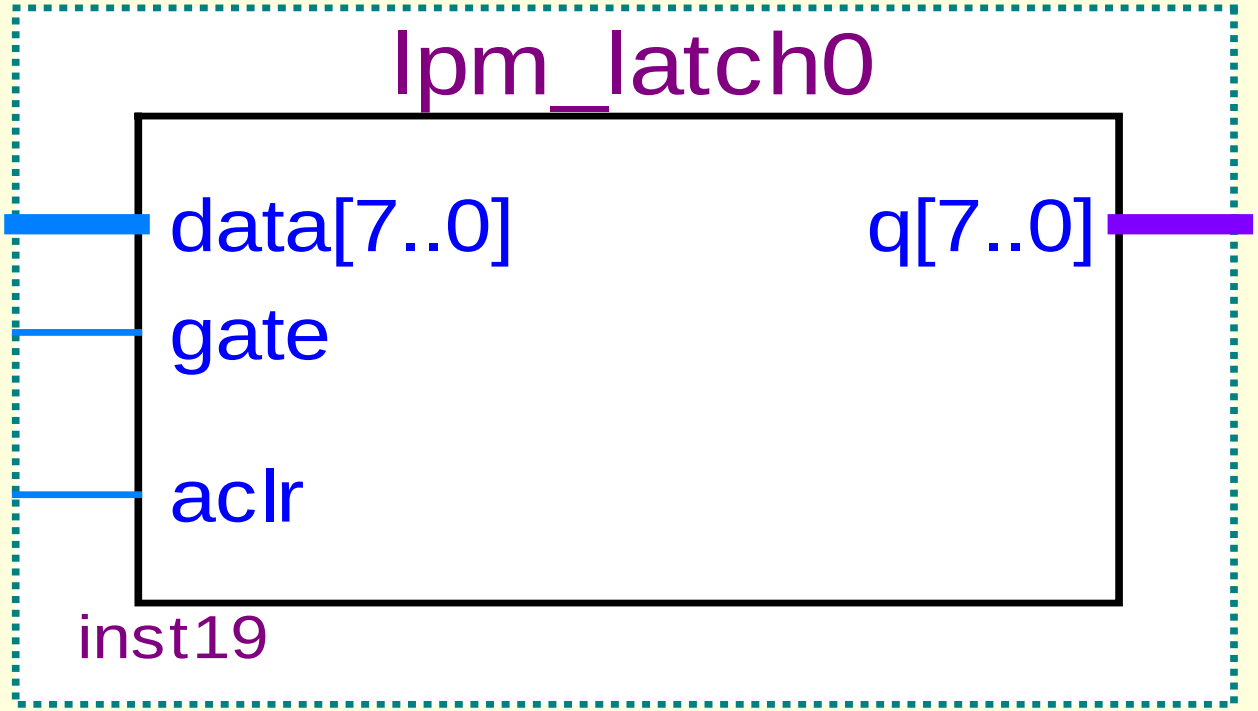

图4-28 指令寄存器结构图

4.2 CPU中的基本部件

4.2.9 指令译码器与控制器

1. 指令译码器（Instruction Decoder. ID）

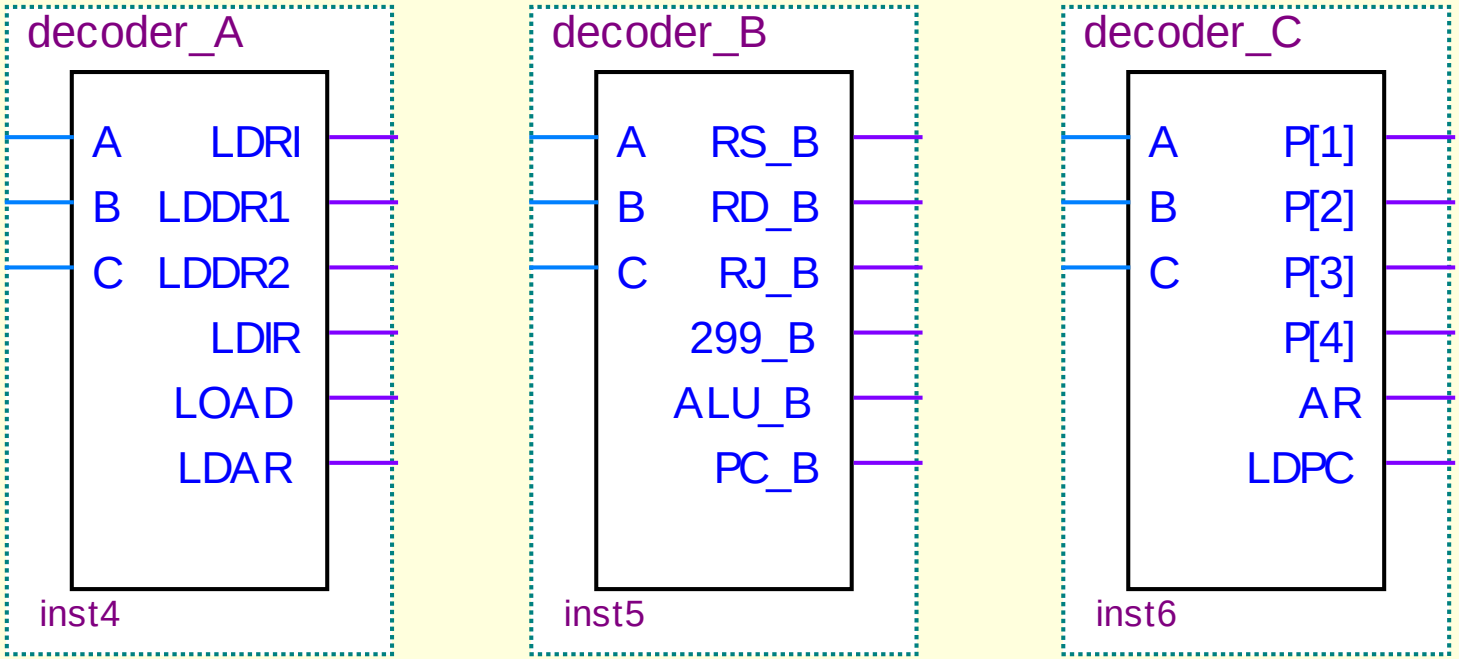

图4-29 指令译码器结构图

4.2 CPU中的基本部件

4.2.9 指令译码器与控制器

2. 微程序控制器

（1）微命令，微操作，微指令和微程序

（2）相容性微操作和相斥性微操作

4.2 CPU中的基本部件

4.2.9 指令译码器与控制器

3. 微程序控制的基本原理

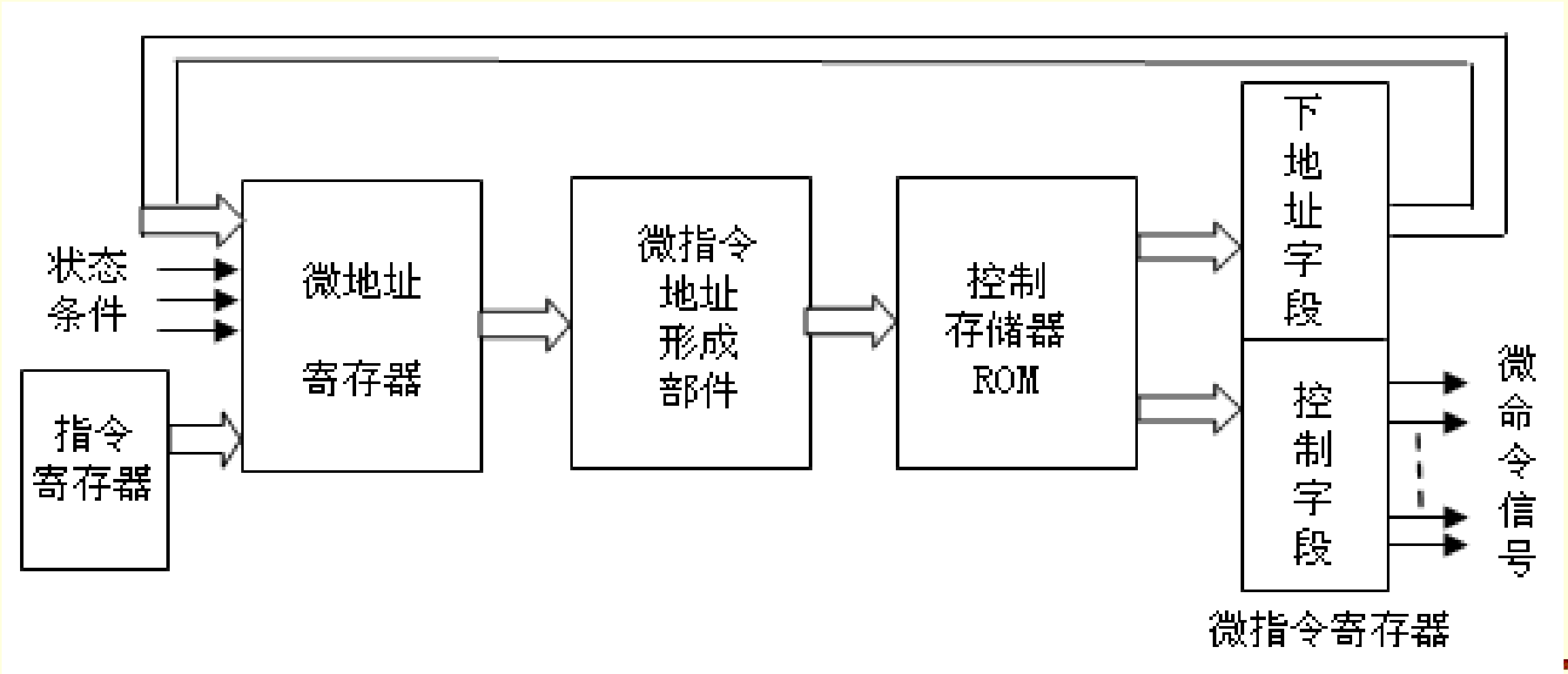

图4-30微程序控制的基本原理图

4.2 CPU中的基本部件

4.2.9 指令译码器与控制器

4. 微程序执行过程

（1）从控制存储器中取出一条“取指令”用的微指令，并送到微指令寄存器。

（2）指令操作码通过微地址形成线路，产生对应的微程序入口。

（3）逐条取出对应的微指令。

（4）执行完对应的一条指令的一段微程序后，返回0号或1号微地址单元，读取“取指令”的微指令，以便取下一条指令。

5. 微指令地址形成部件

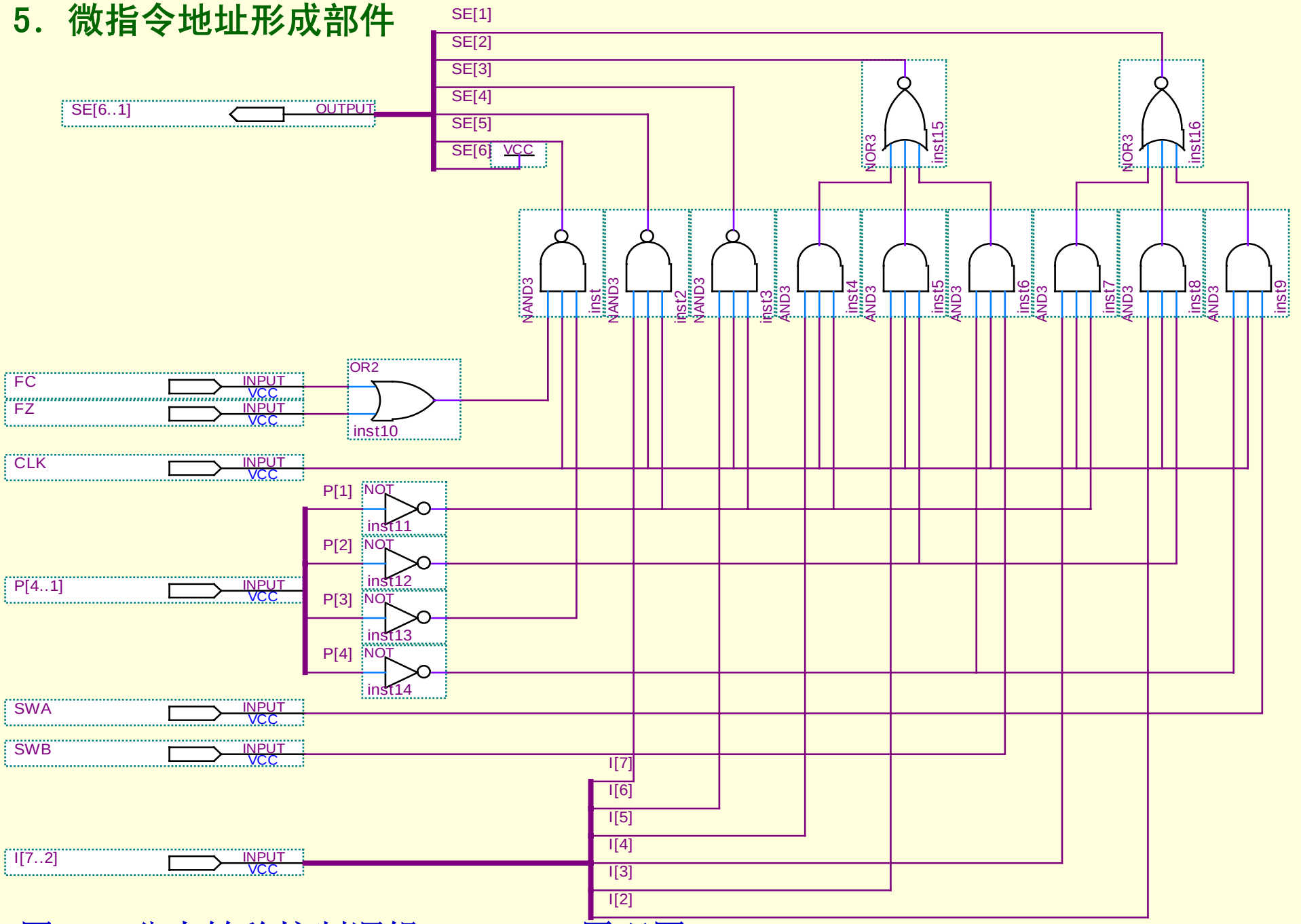

图4-31 分支转移控制逻辑QuartusII原理图

4.2 CPU中的基本部件

6. 数据寄存器存取控制逻辑

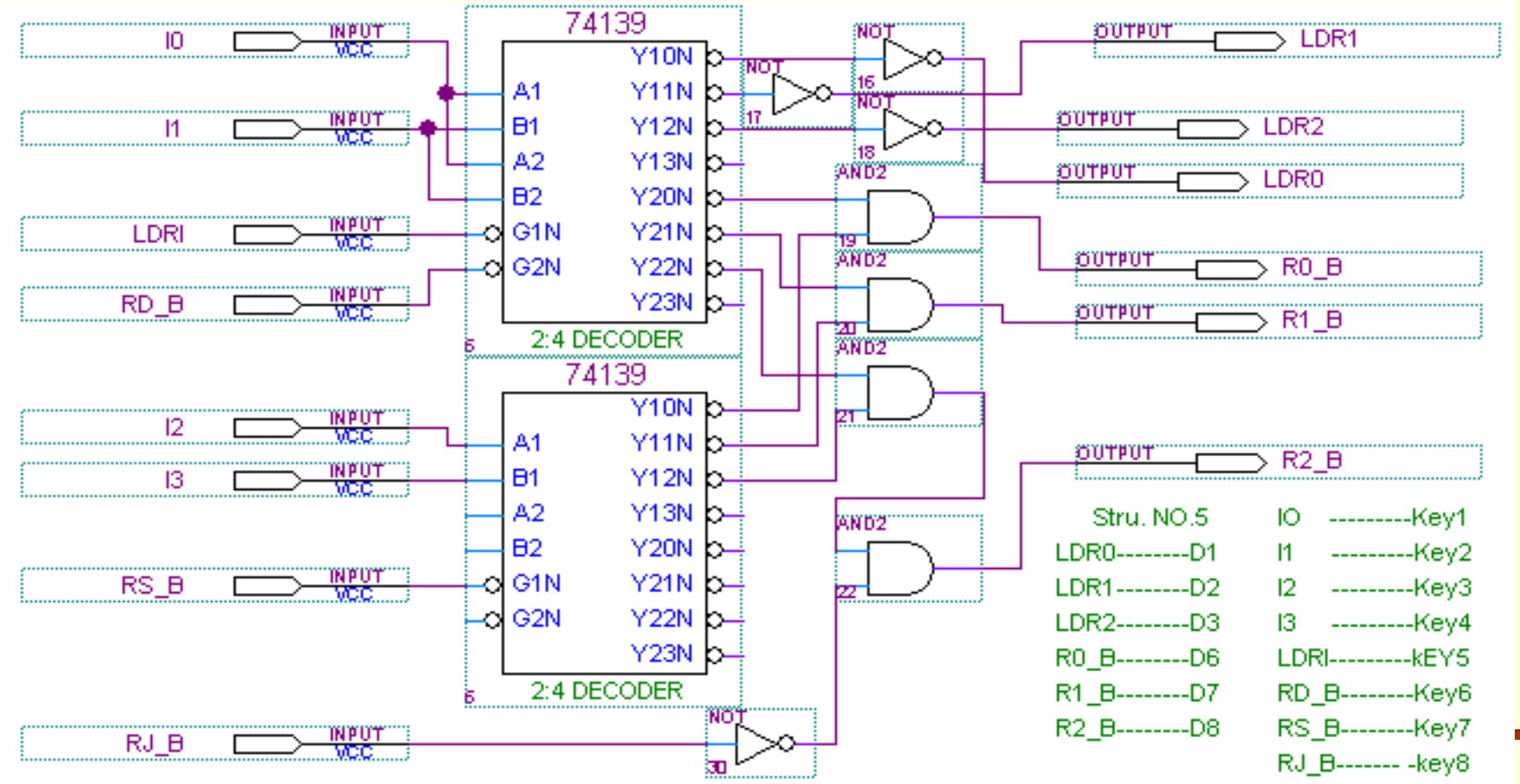

图4-32 数据寄存器控制逻辑

4.2 CPU中的基本部件

4.2.10 时序产生器

1. 连续节拍发生电路设计

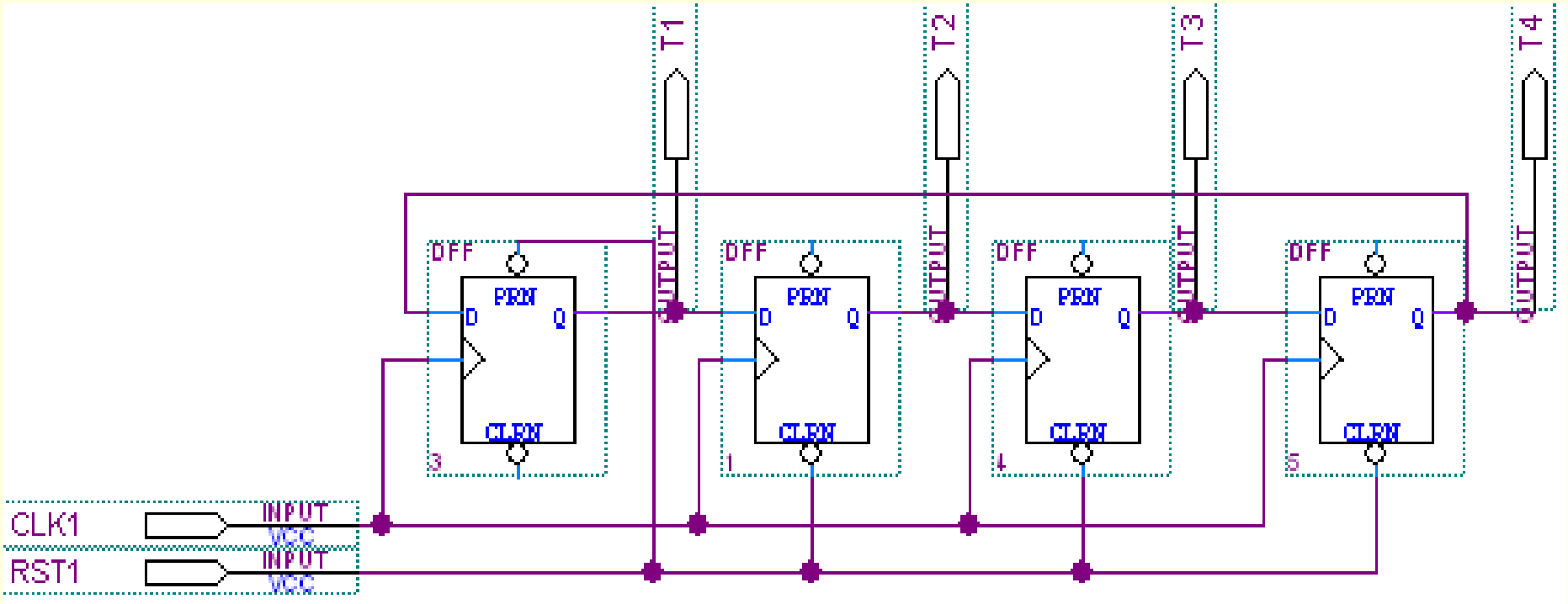

图4-33 节拍脉冲发生器的工作原理

4.2 CPU中的基本部件

4.2.10 时序产生器

1. 连续节拍发生电路设计

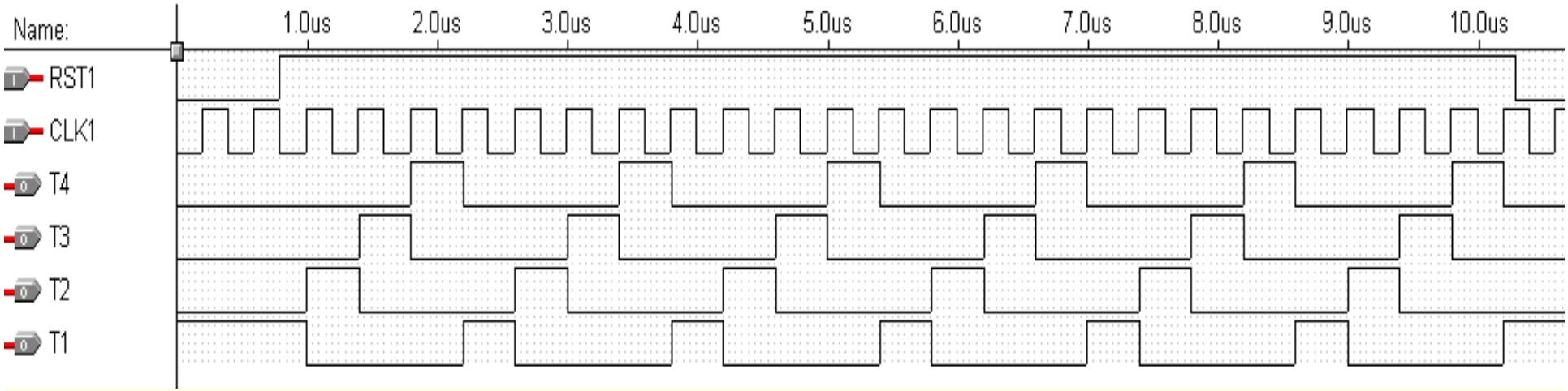

图4-34 节拍脉冲发生器工作波形

4.2 CPU中的基本部件

4.2.10 时序产生器

2. 单步节拍发生电路

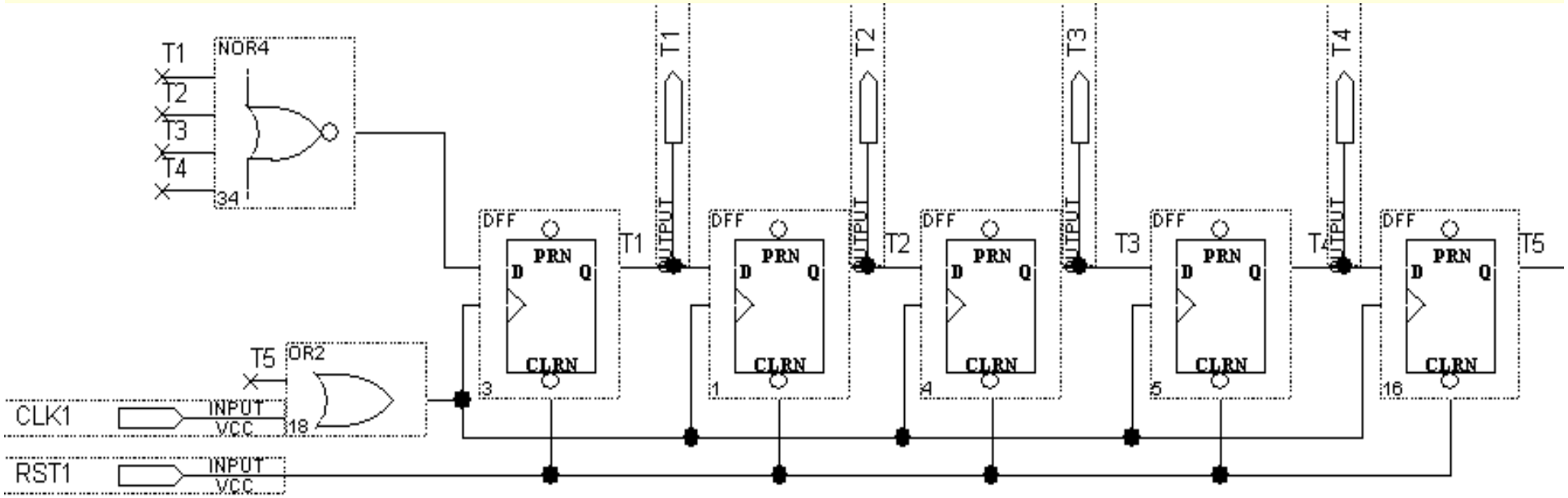

图4-35单步运行电路工作原理

4.2 CPU中的基本部件

4.2.10 时序产生器

2. 单步节拍发生电路

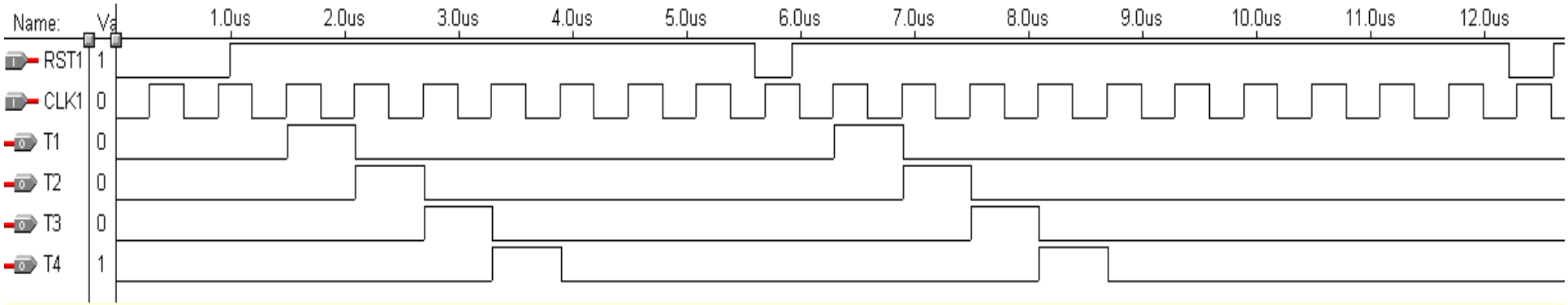

图4-36单步运行电路工作波形

4.2 CPU中的基本部件

4.2.10 时序产生器

3. 单步/连续节拍发生电路

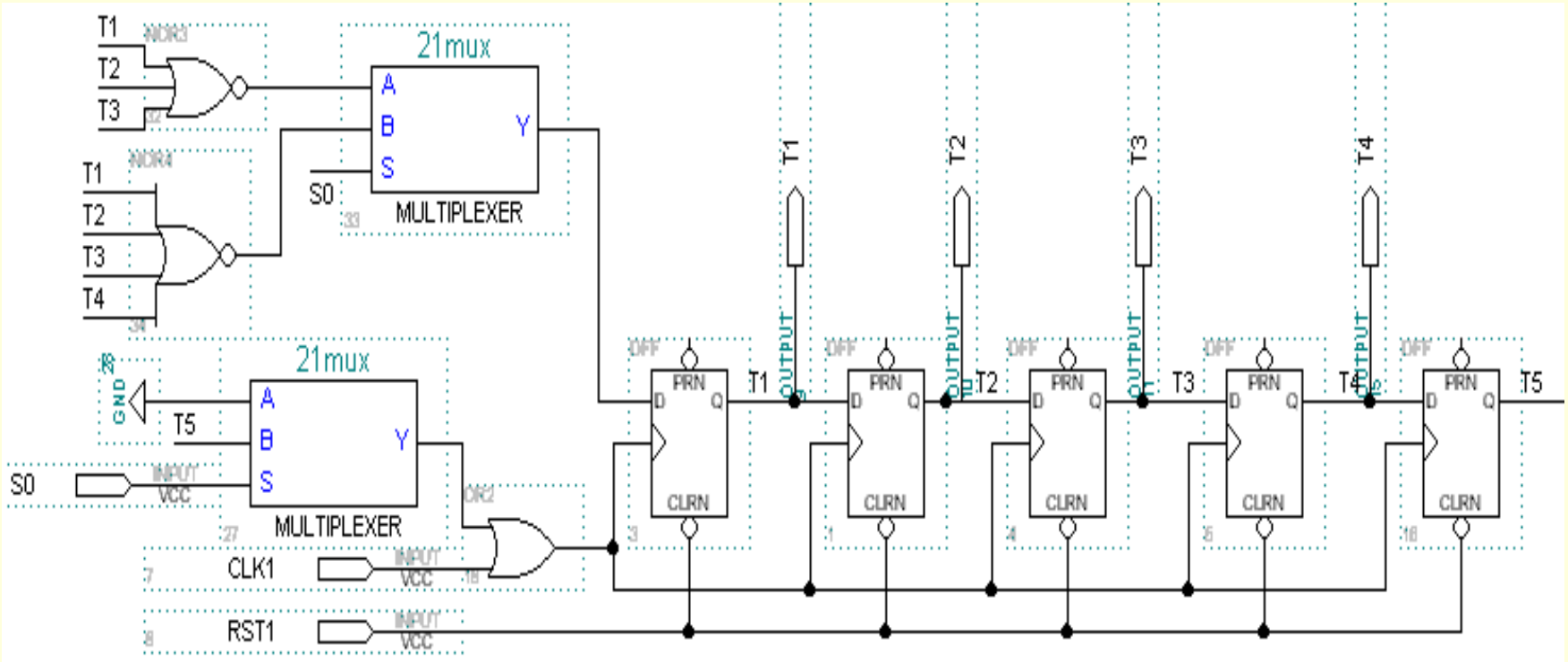

图4-37单步/连续运行电路工作原理

4.2 CPU中的基本部件

4.2.10 时序产生器

3. 单步/连续节拍发生电路

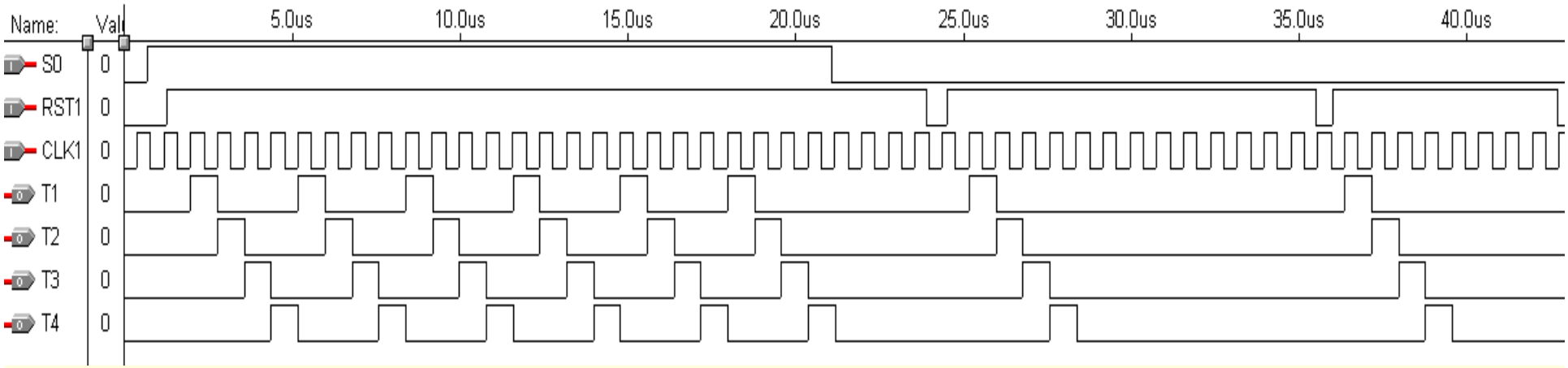

图4-38 单步运行电路工作波形

4.3 数据通路设计

4.3.1 模型机的数据通路

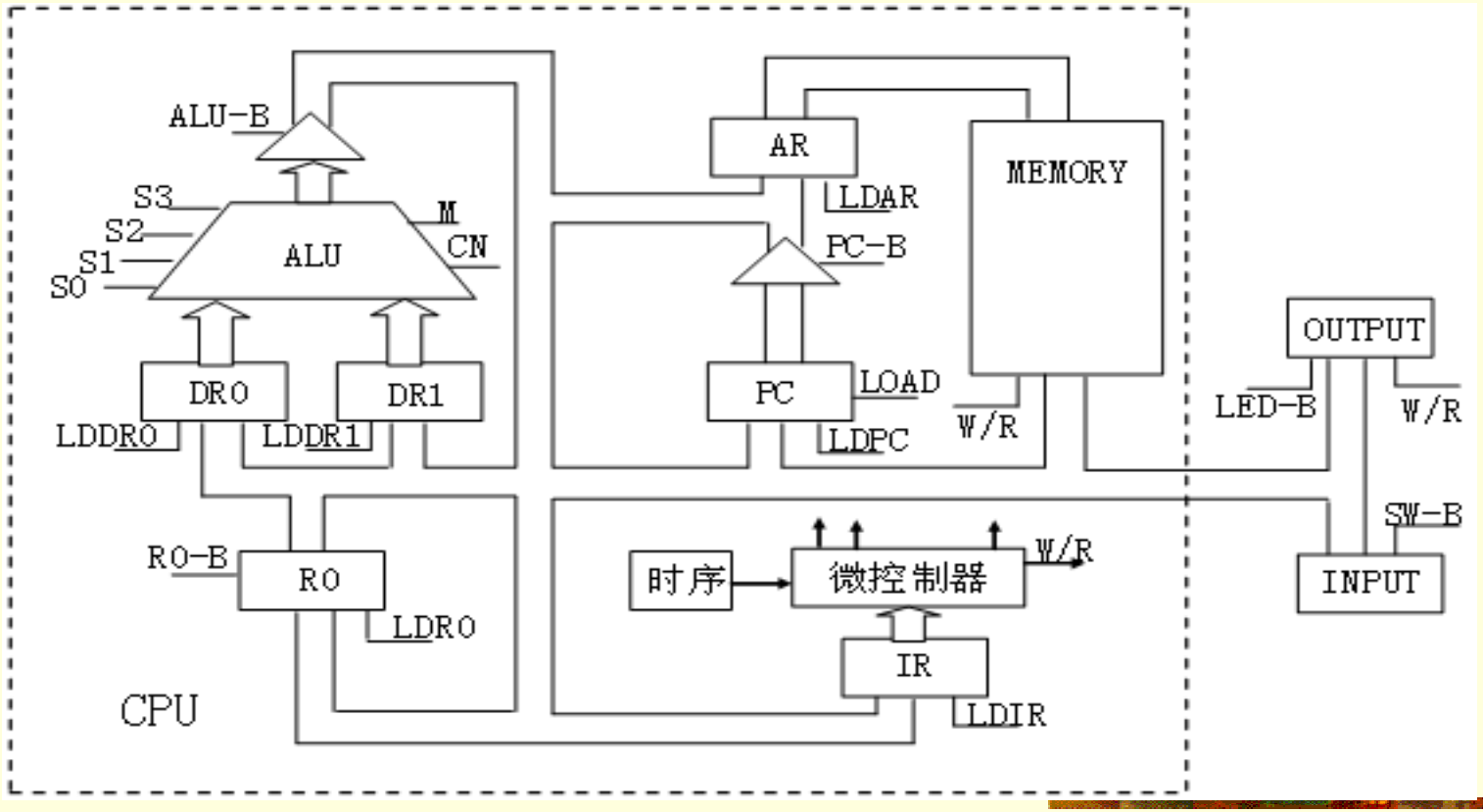

图4-39 单总线结构的模型机CPU的内部数据通路

4.3 数据通路设计

4.3.2 模型机的电路结构

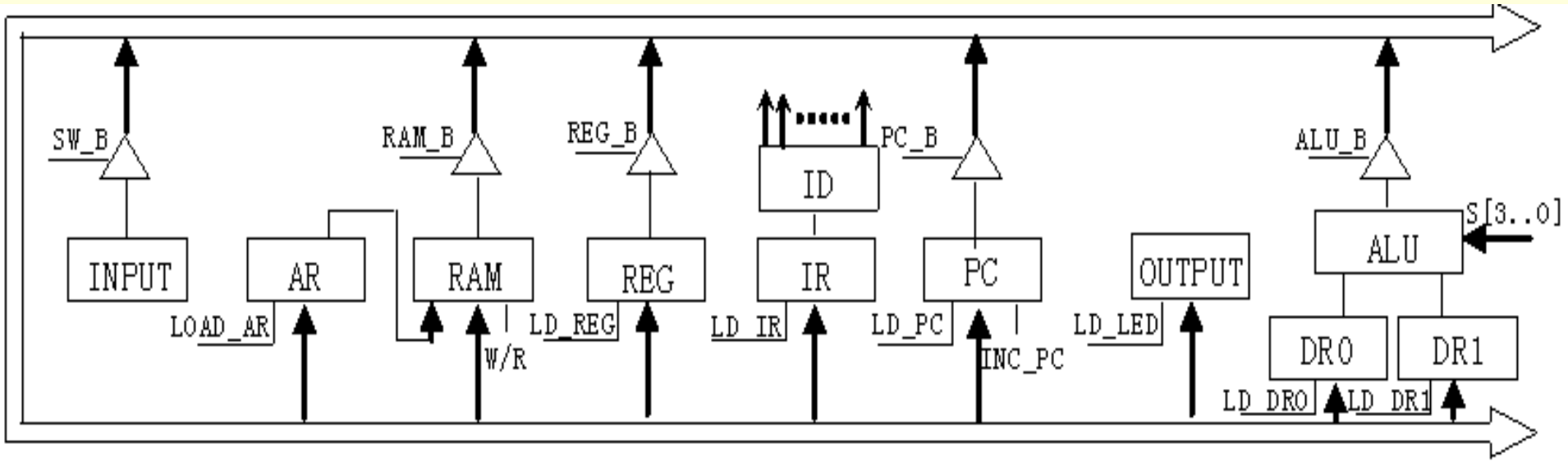

图4-40简单模型CPU的数据通路

4.4在系统存储器数据读写编辑器应用

（1）打开在系统存储单元编辑窗

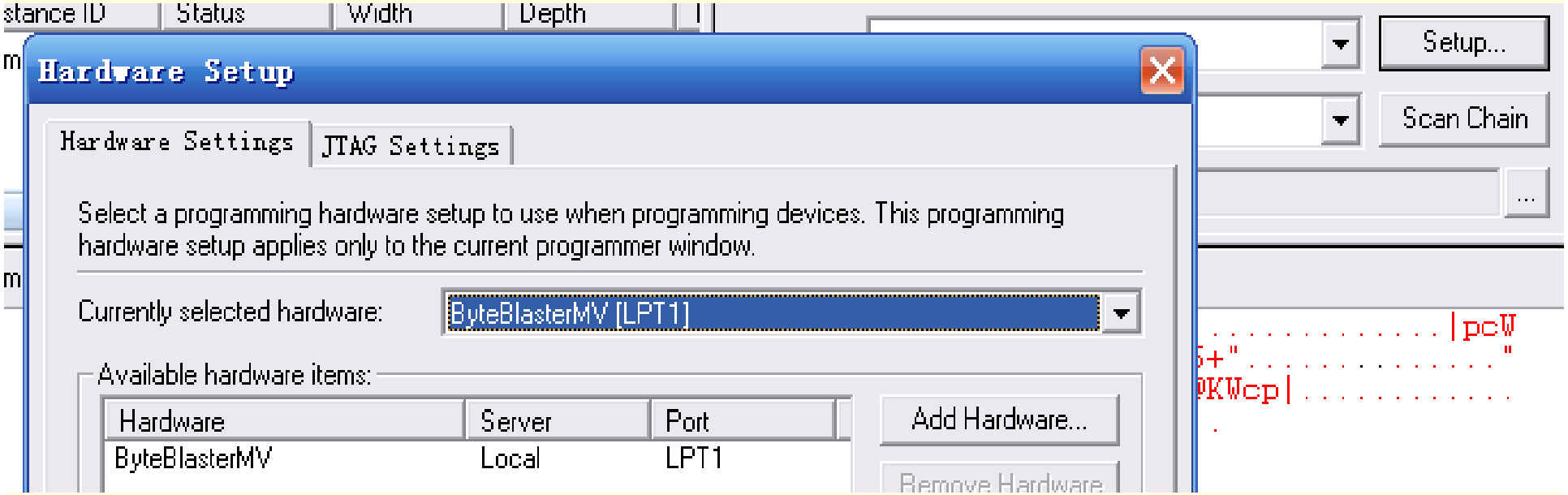

图4-41 In-System Memory Content Editor编辑窗

4.4在系统存储器数据读写编辑器应用

（2）读取ROM中的波形数据

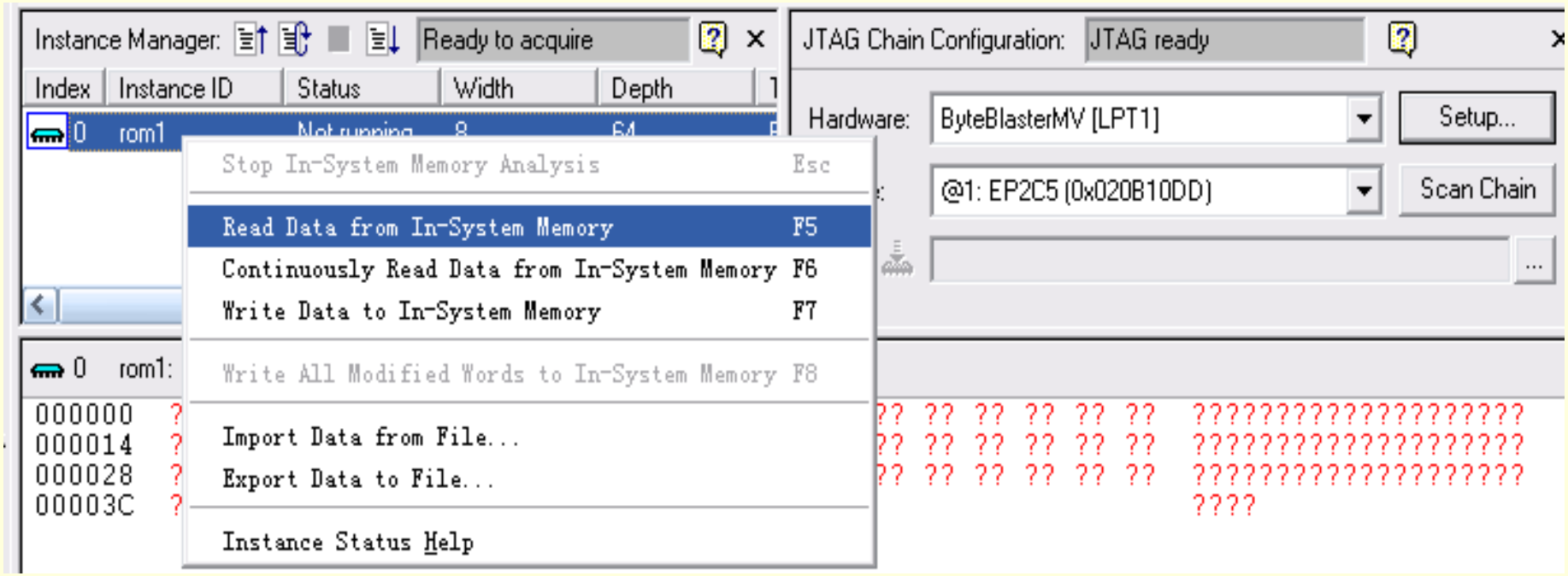

图4-42 与实验系统上的FPGA通信正常情况下的编辑窗界面

4.4在系统存储器数据读写编辑器应用

（2）读取ROM中的波形数据

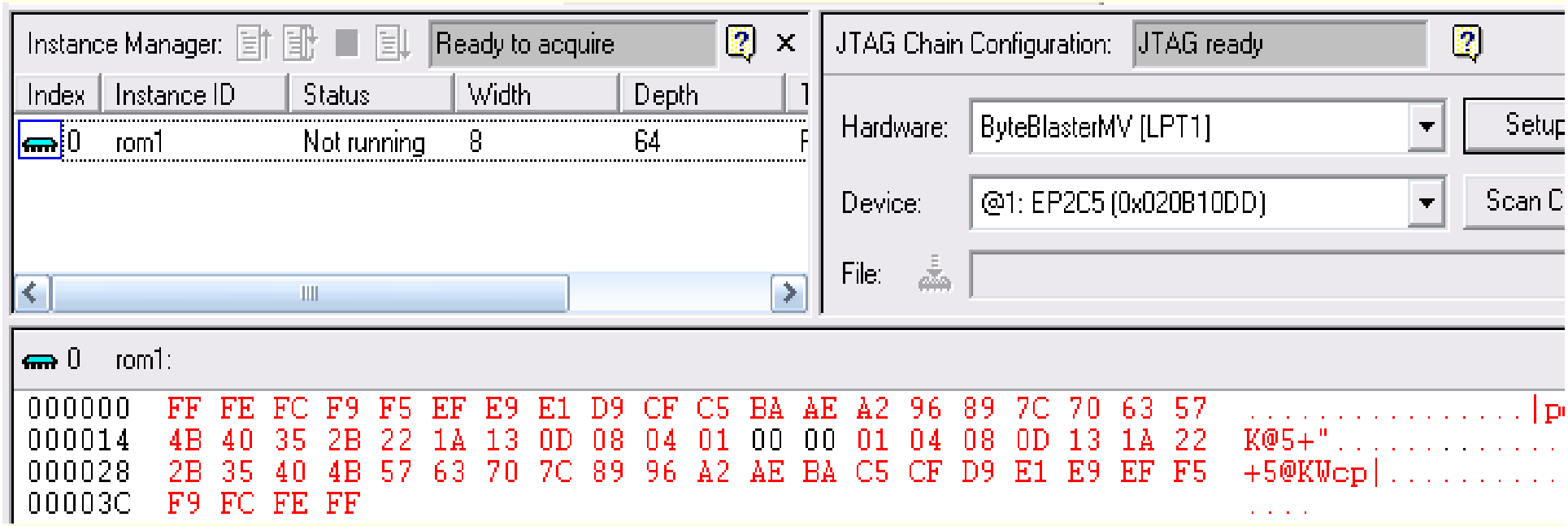

图4-43 从FPGA中的ROM读取波形数据

4.4在系统存储器数据读写编辑器应用

（3）写数据

0 rom1:

```
000000  11 11 11 11 F5 EF E9 E1 D9 CF C5 BA AE A2 96 89 7C 70 63 57
000014  4B 40 35 2B 22 1A 13 0D 08 04 01 00 00 01 04 08 0D 13 1A 22
000028  2B 35 40 4B 57 63 70 7C 89 96 A2 AE BA C5 CF D9 E1 E9 EF F5
00003C  F9 FC FE FF
```

图4-44 编辑波形数据

4.4在系统存储器数据读写编辑器应用

（3）写数据

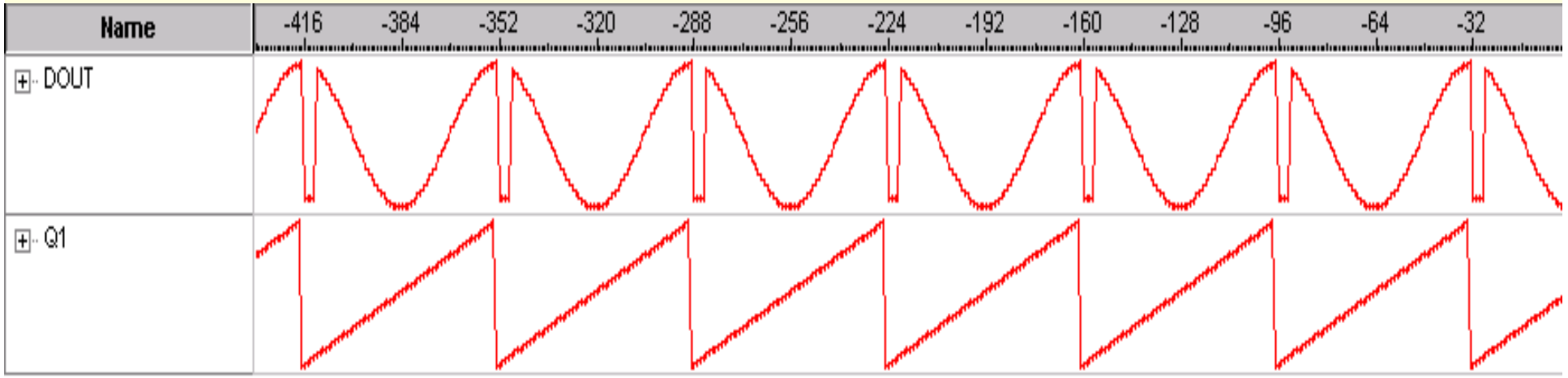

图4-45下载编辑数据后的SignalTap II采样波形

（4）输入输出数据文件

4.5 嵌入式锁相环调用

4.5.1 建立嵌入式锁相环元件

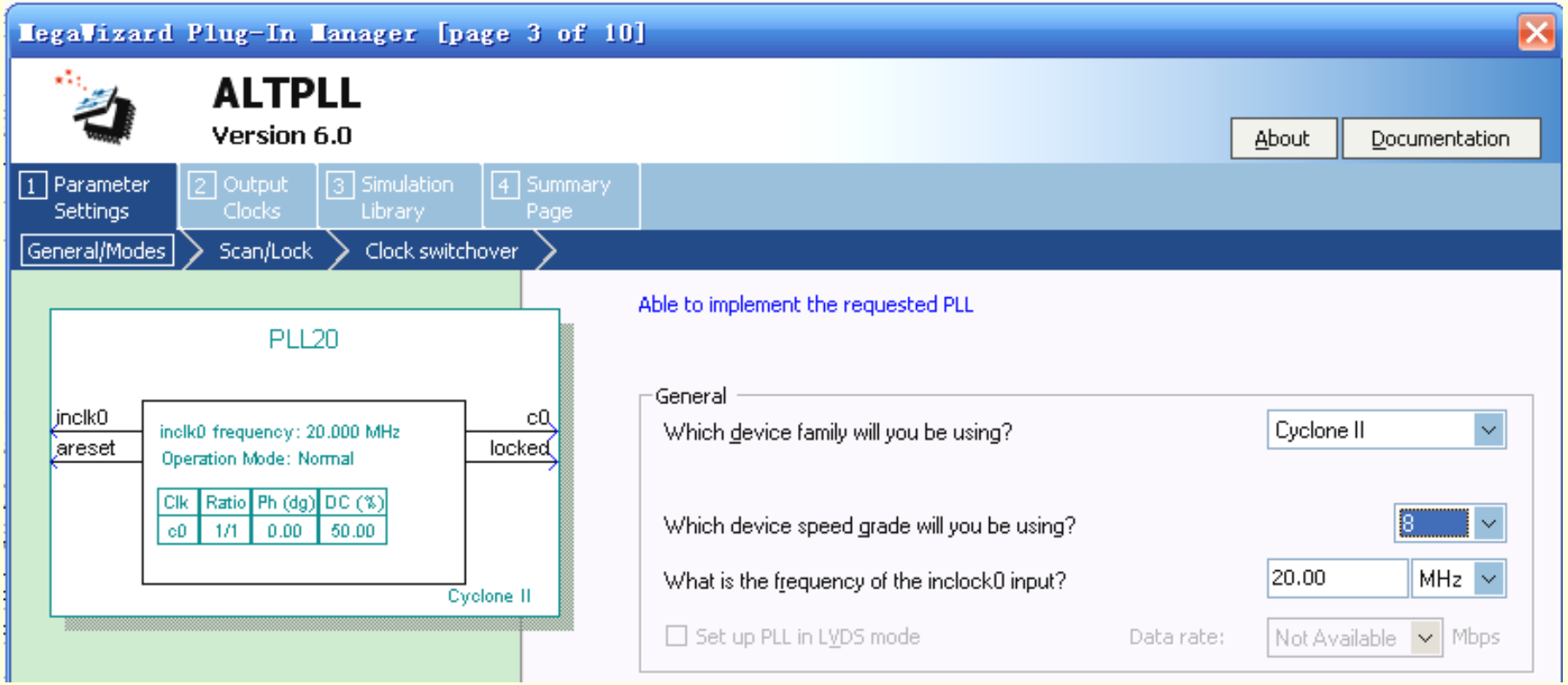

图4-46 选择参考时钟为20MHz

4.5 嵌入式锁相环调用

4.5.1 建立嵌入式锁相环元件

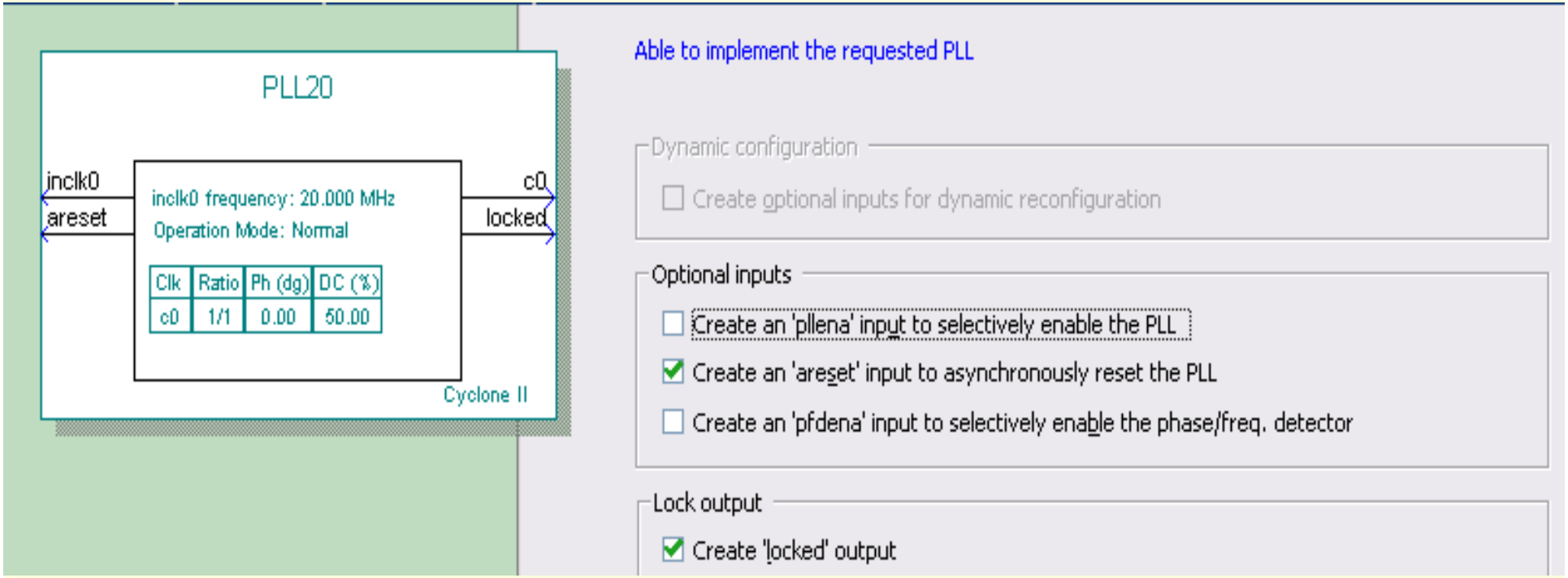

图4-47 选择控制信号

4.5 嵌入式锁相环调用

4.5.1 建立嵌入式锁相环元件

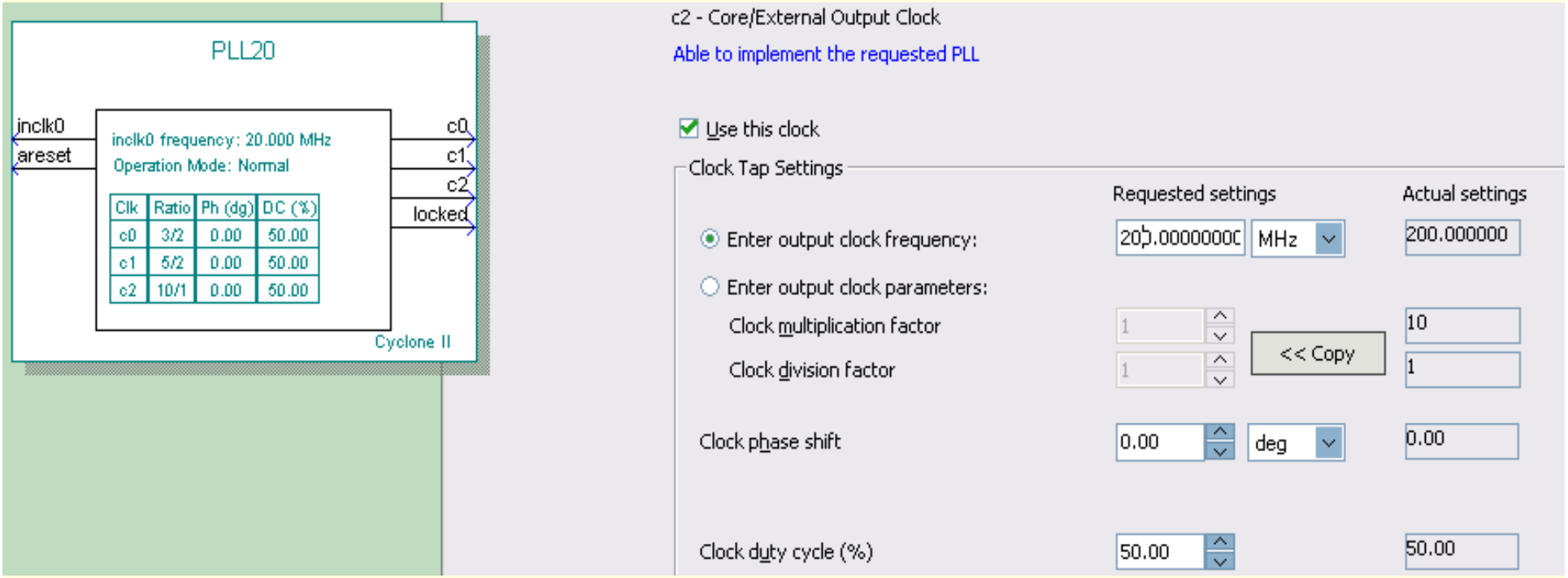

图4-48 选择e0的输出频率为210MHz

4.5 嵌入式锁相环调用

4.5.2 测试锁相环

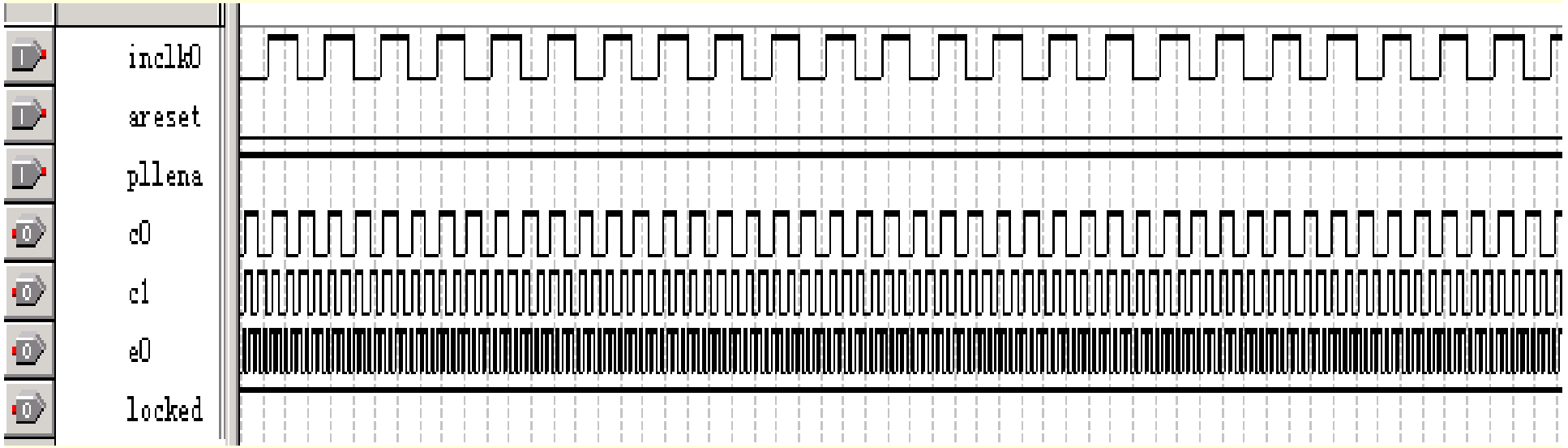

图4-49 PLL元件的仿真波形

4.5 嵌入式锁相环调用

4.5.2 测试锁相环

单频率输出的VHDL文本调用PLL的示例：

```
ENTITY DDS_VHDL IS
    PORT (    CLKK : IN  STD_LOGIC;  --此时钟进入锁相环
              FWORD : IN  STD_LOGIC_VECTOR(7 DOWNTO 0);
...;
ARCHITECTURE one OF DDS_VHDL IS
  COMPONENT PLLU                    --调入PLL声明
     PORT (     inclk0 : IN STD_LOGIC  := '0';
                          c0 : OUT STD_LOGIC        );
   END COMPONENT;
    COMPONENT REG32B
...;
u6  :    SIN_ROM  PORT  MAP(  address=>D32B(31  DOWNTO  22),  q=>POUT,
inclock=>CLK );
 u7 :   PLL20  PORT MAP( inclk0=> CLKK,c0=>CLK);  --例化
END;
```

习 题

4-1. 简述微程序控制器和组合逻辑控制器的异同点。

4-2．简要说明图**6-1**中，**CPU**各组成部件的作用。控制器由哪些部件组成，运算器由哪些部件组成？

4-3．在微硬序控制器中，微程序计数器**uPC**可以用**uAR**来代替，试问是否可以用具有计数功能的存储器地址寄存器**AR**来代替程序计数器**PC?**为什么**?**

4-4．试说明机器指令和微指令之间的关系。

4-5．机器指令包含哪两个基木要素？微指令义也含哪两个基本要素？程序靠什么实现顺序执行？靠什么实现转移？微程序中顺序执行和转移依靠什么方法？

习 题

4-6．完成下列数据传输功能，说明数据传输的具体操作步骤：

编号	功　能	助记符
1	从INPUT端口输入数据写入R1	IN R1，PORT
2	从INPUT端口输入数据写入R2	IN R2，PORT
3	从INPUT端口输入数据写入RAM某单元	IN RAM，PORT
4	将RAM某单元内容读入R1	LD R1，RAM
5	将RAM某单元内容读入R2	LD R2，RAM
6	将R1内容写入RAM某单元	ST RAM，R1
7	将R2内容写入RAM某单元	ST RAM，R2
8	将R1内容传到R2	MOV R2，R1
9	将R2内容传到R1	MOV R1，R2
10	将R1内容输出到LED端口显示	OUT LED，R1
11	将R2内容输出到LED端口显示	OUT LED，R2
12	将RAM某单元内容输出到LED端口显示	OUT LED，RAM
13	从INPUT端口输入数据送LED端口显示	OUR LED，PORT

习 题

4-7. 如果不使用MegaWizard Plug-In Manager工具，如何在自己的设计中调用LPM模块？以计数器lpm_counter为例，写出调用该模块的程序，其中参数自定。

4-8. LPM_ROM、LPM_RAM、LPM_FIFO等模块与FPGA中嵌入的EAB，ESB，M4K有怎样的联系关系？

4-9. 参考QuartusII的Help（Contents），详细说明LPM元件altcam、altsyncram、lpm_fifo、lpm_shiftreg的使用方法，以及其中各参量的含义和设置方法。

实验与设计

实验4-1. 算术逻辑运算单元ALU设计实验

参考实验示例和实验课件：

/ CMPUT_EXPMT/CH4_Expt/ DEMO_41_alu/ 和 实验4_1.ppt 。

实验与设计

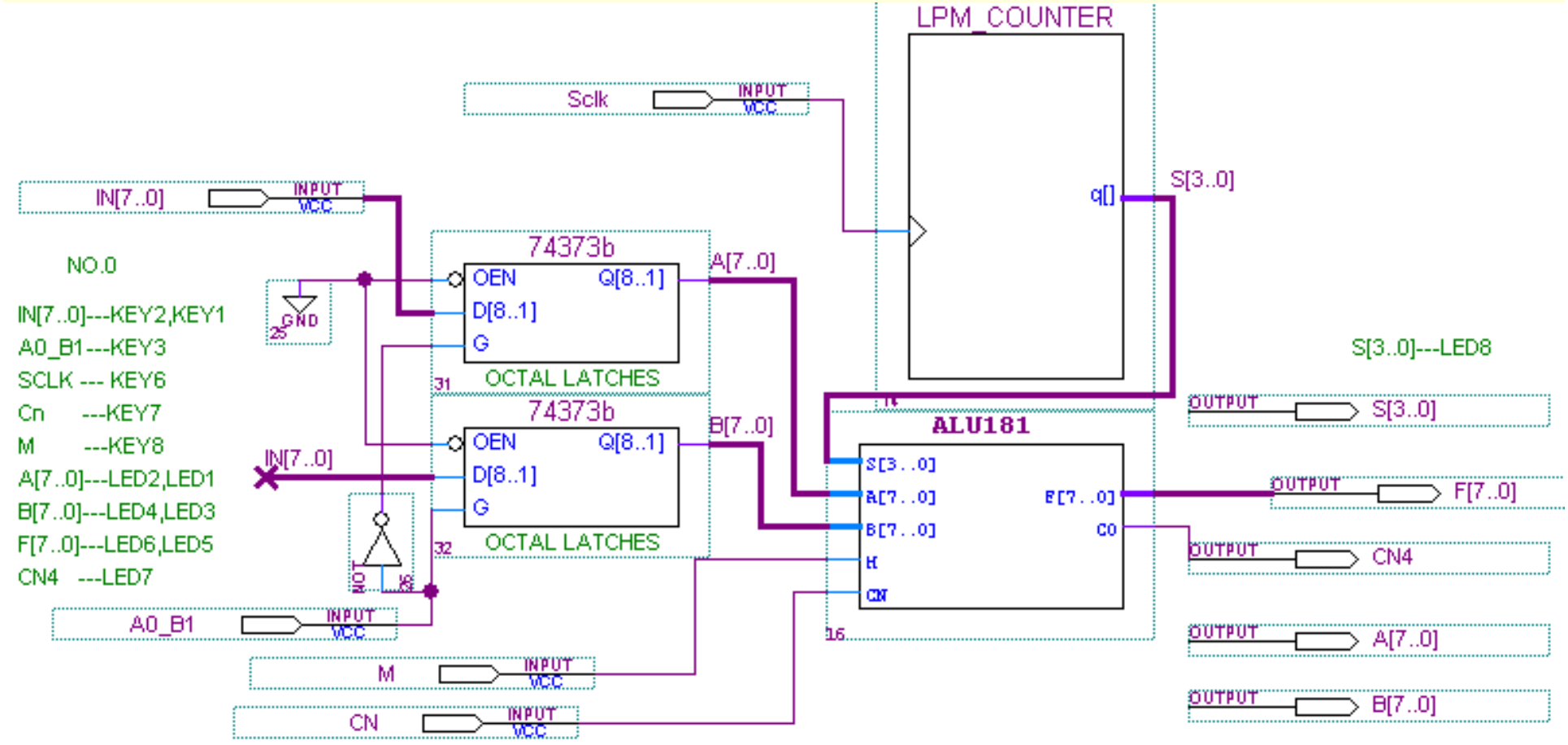

图4-50 算术逻辑单元ALU实验原理图

实验与设计

实验4-1. 算术逻辑运算单元AL设计实验

表4-3 A[7..0]，B[7..0]设置值检查

F[7..0]	SW_B	寄存器内容		S3 S2 S1 S0	M	BUS
		A[7..0]	B[7..0]			
		01010101	1010101010			
		01010101	1010101010			

实验与设计

表4-4

S3 S2 S1 S0	A[7..0]	B[7..0]	算术运算 M=0		逻辑运算（M=1）
			Cn=0（无进位）	cn=1（有进位）	
0000	AA	55	F=（ ）	F=（ ）	F=（ ）
0001	AA	55	F=（ ）	F=（ ）	F=（ ）
0010	AA	55	F=（ ）	F=（ ）	F=（ ）
0011	AA	55	F=（ ）	F=（ ）	F=（ ）
0100	FF	01	F=（ ）	F=（ ）	F=（ ）
.....省略					
0101	FF	01	F=（ ）	F=（ ）	F=（ ）
1101	55	01	F=（ ）	F=（ ）	F=（ ）
1110	55	01	F=（ ）	F=（ ）	F=（ ）
1111	55	01	F=（ ）	F=（ ）	F=（ ）

实验与设计

实验4-1. 算术逻辑运算单元AL设计实验

表4-5　8种常用的算术与逻辑运算

操作	S3S2S1S0	M	Cn	DR1	DR2	运算关系及结果显示	Cn4
逻辑乘				66	FF	$DR_1 \wedge DR_2 \rightarrow DR_2$(　)	
传送						$DR_1 \rightarrow DR_2$　(　)	
按位加						$DR_1 \oplus DR_2 \rightarrow DR_2$(　)	
取反						$\overline{DR_1} \rightarrow DR_2$　(　)	
加1						$DR_2 + 1 \rightarrow DR_2$　(　)	
求负						$\overline{DR_2} + 1 \rightarrow DR_2$　(　)	
加法						$DR_1 + DR_2 \rightarrow DR_2$(　)	
减法						$DR_1 - DR_2 \rightarrow DR_2$(　)	

实验与设计

实验4-2. 带进位算术逻辑运算单元ALU设计实验

参考实验示例和实验课件：

/CMPUT_EXPMT/CH4_Expt/ DEMO_42_aluc/ 和 实验 **4_2.ppt** 。

实验与设计

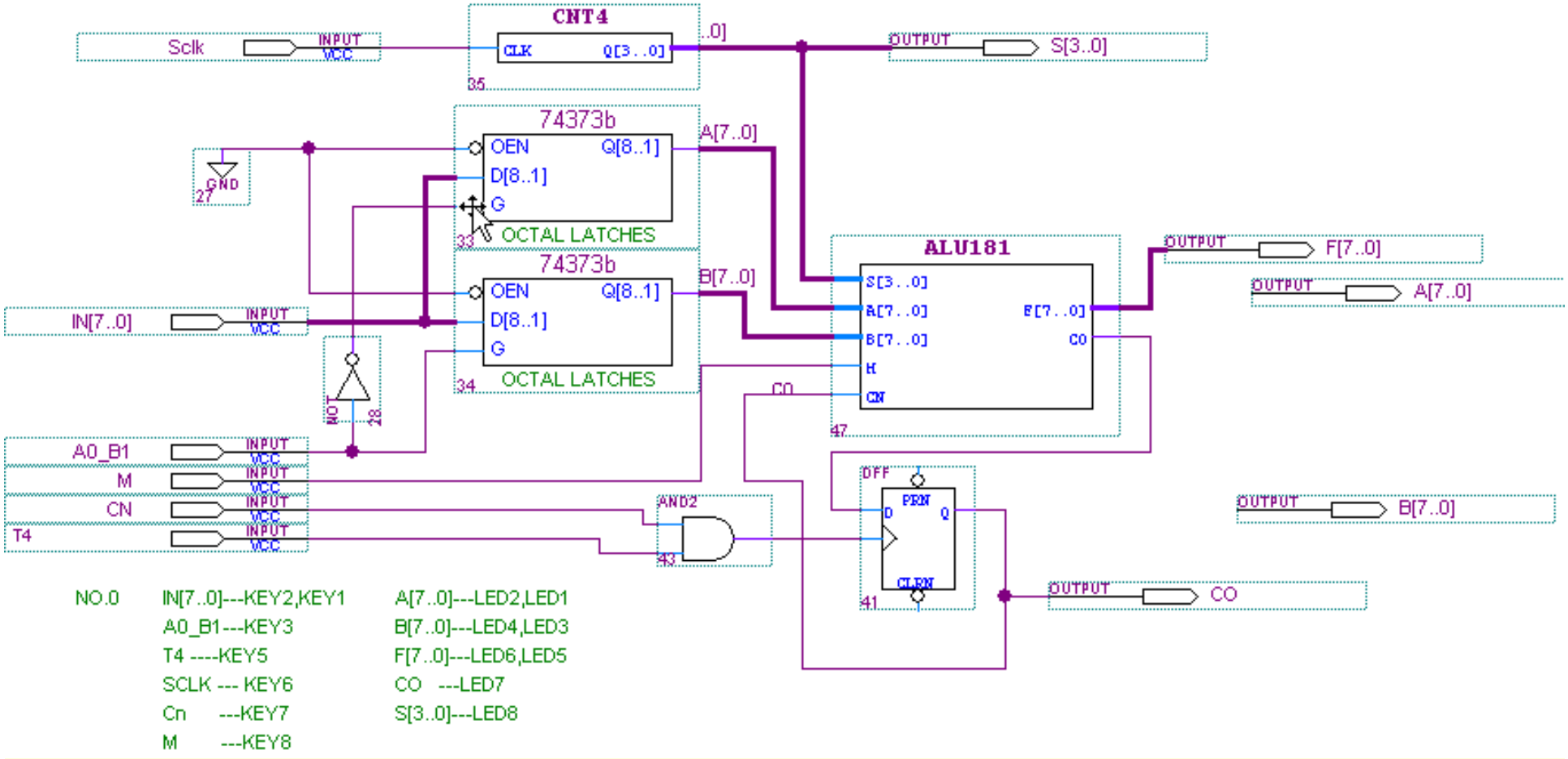

图4-51 带进位控制的ALU

实验与设计

实验4-2. 带进位算术逻辑运算单元ALU设计实验

表4-6

S3 S2 S1 S0	A[7..0]	B[7..0]	算术运算 M=0		逻辑运算（M=1）
			cn=0（无进位）	cn=1（有进位）	
0101	FF	01	F=（ ）	F=（ ）	F=（ ）
0110	FF	01	F=（ ）	F=（ ）	F=（ ）
0111	FF	01	F=（ ）	F=（ ）	F=（ ）
1000	FF	FF	F=（ ）	F=（ ）	F=（ ）
1001	FF	FF	F=（ ）	F=（ ）	F=（ ）
1010	FF	FF	F=（ ）	F=（ ）	F=（ ）

实验与设计

实验4-2. 带进位算术逻辑运算单元ALU设计实验

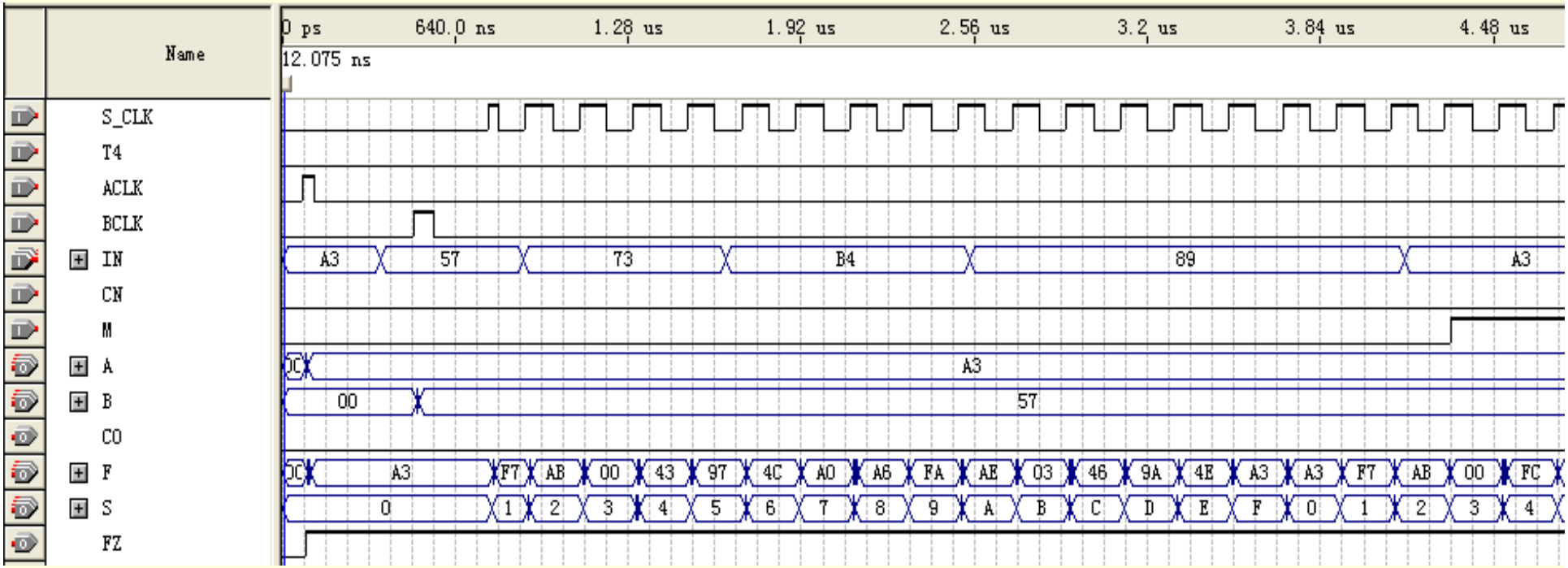

图4-52 带进位控制的ALU的仿真波形

实验与设计

实验4-3. 移位运算器设计实验

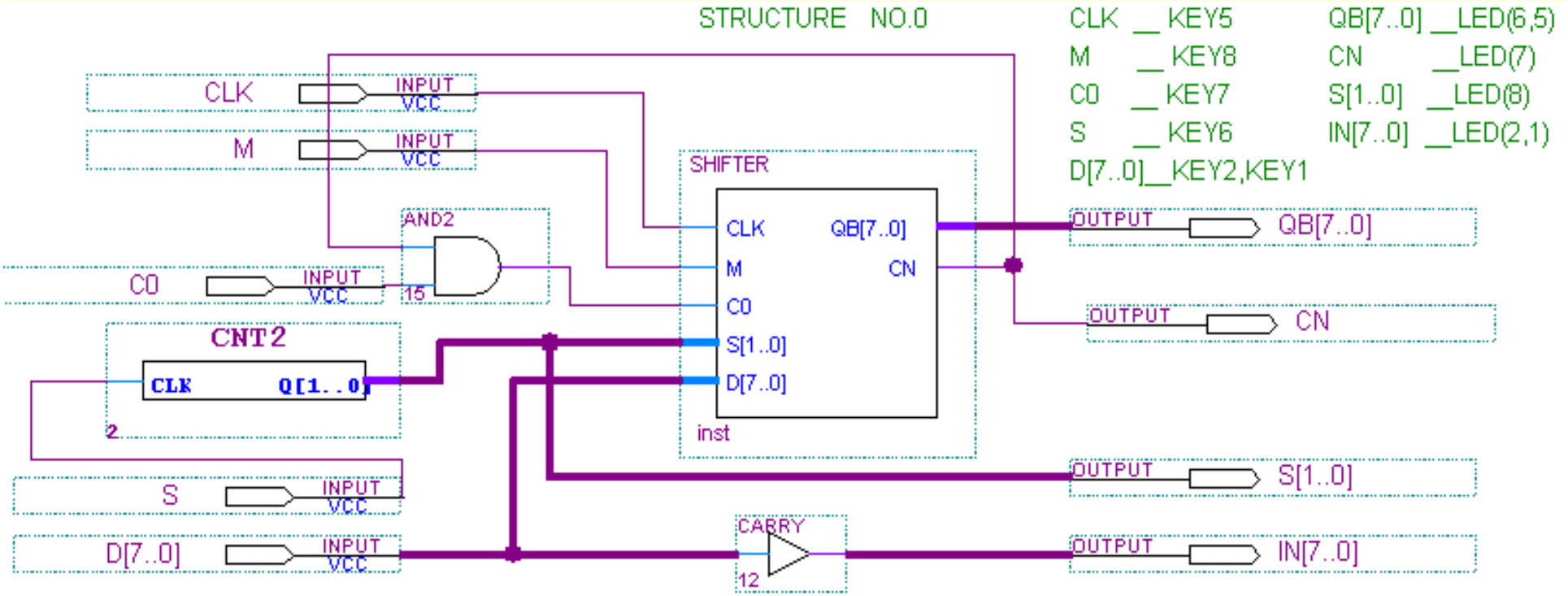

图4-53 移位运算实验原理图

实验与设计

实验4-3. 移位运算器设计实验

表4-7　移位发生器的功能

G	S1	S0	M	功　能
0	0	0	任意	保持
0	1	0	0	循环右移
0	1	0	1	带进位循环右移
0	0	1	0	循环左移
0	0	1	1	带进位循环左移
任意	1	1	任意	加载待移位数

实验与设计

实验4-3. 移位运算器设计实验

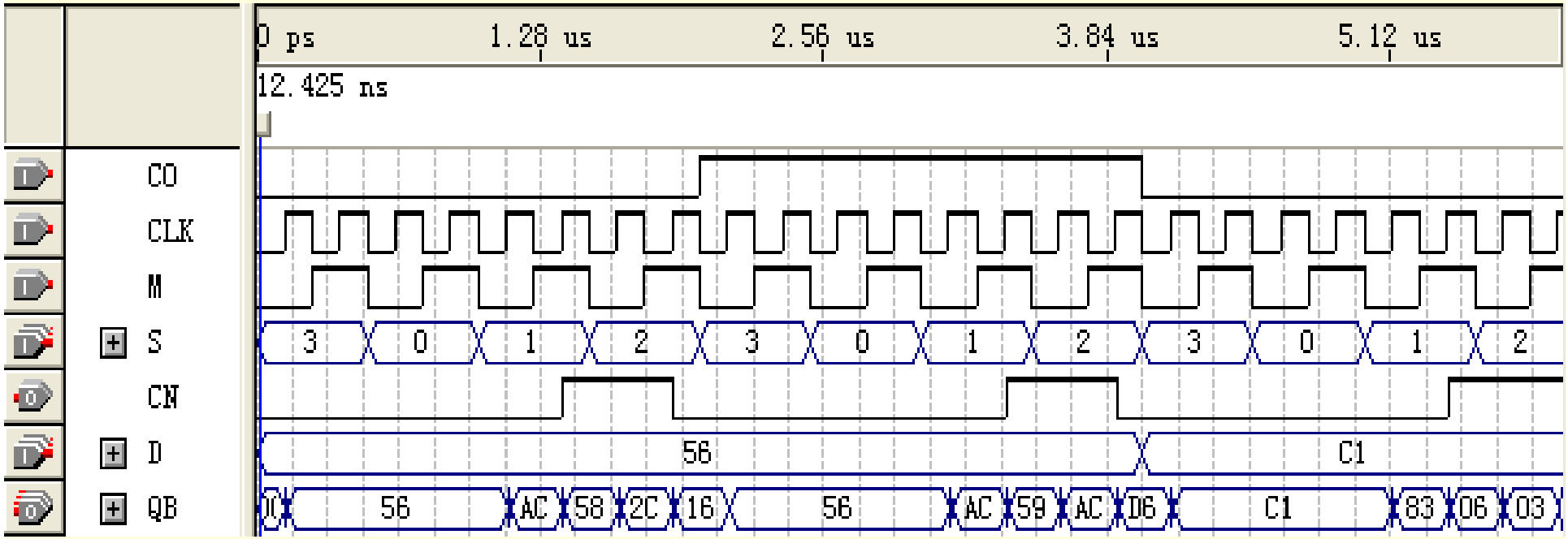

图4-54 shift移位运算器仿真波形

实验与设计

实验4-4. LPM_ROM实验

参考实验示例和实验课件：

/CMPUT_EXPMT/CH4_Expt/ DEMO_44_ROM/ 和 实验4_4.ppt 。

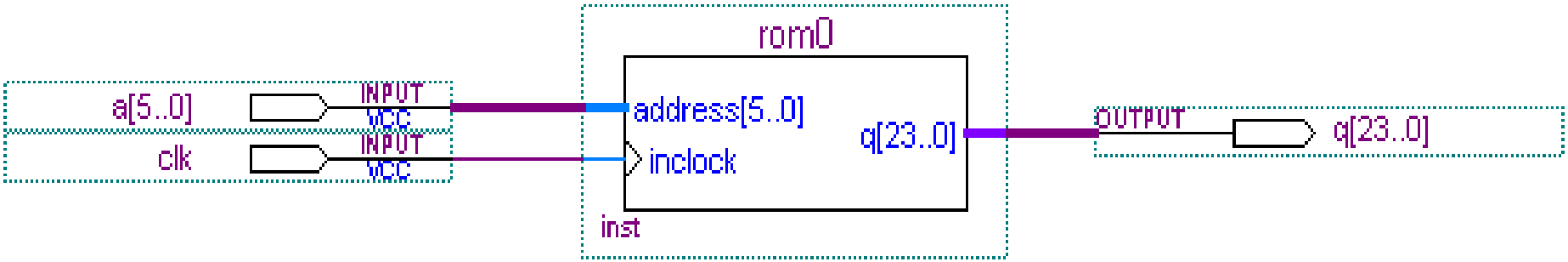

图4-55 LPM_ROM的结构

实验与设计

实验4-4. LPM_ROM实验

Addr	+0	+1	+2	+3	+4	+5	+6	+7
00	018108	00ED82	00C050	00E004	00B005	01A206	959A01	00E00F
08	00ED8A	00ED8C	00A008	008001	062009	062009	070A08	038201
10	001001	00ED83	00ED87	00ED99	00ED9C	31821D	31821F	318221
18	318223	00E01A	00A01B	070A01	00D181	21881E	019801	298820
20	019801	118822	019801	198824	019801	018110	000002	000003
28	000004	000005	000006	000007	000008	000009	00000A	00000B
30	00000C	00000D	00000E	00000F	000010	000011	000012	000013
38	000014	000015	000016	000017	000018	000019	00001A	00001C

图4-56 ROM初始化文件ROM_A.mif的内容

实验与设计

实验4-4. LPM_ROM实验

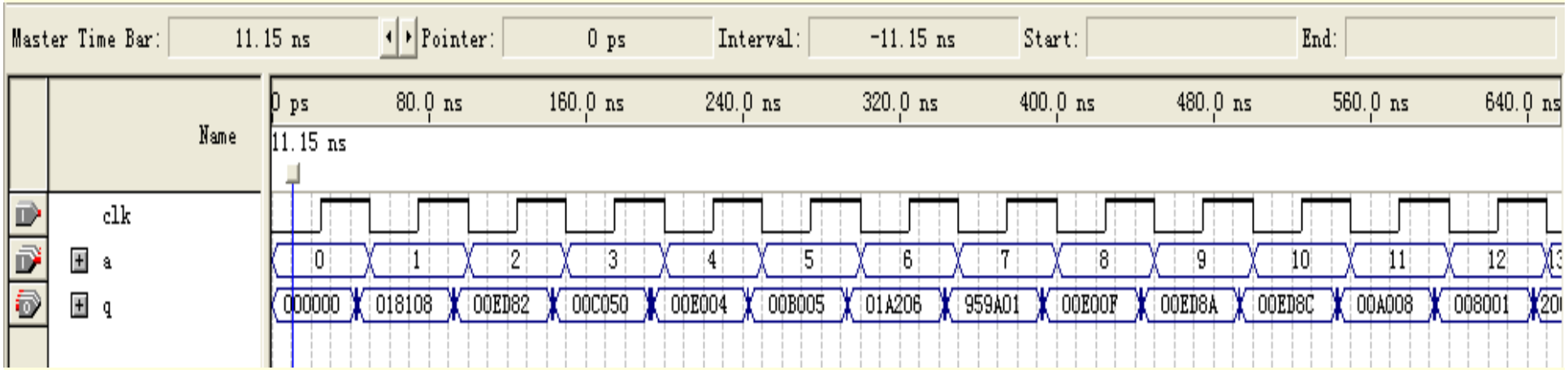

图4-57 LPM_ROM的仿真波形

实验与设计

实验4-5. LPM_RAM实验

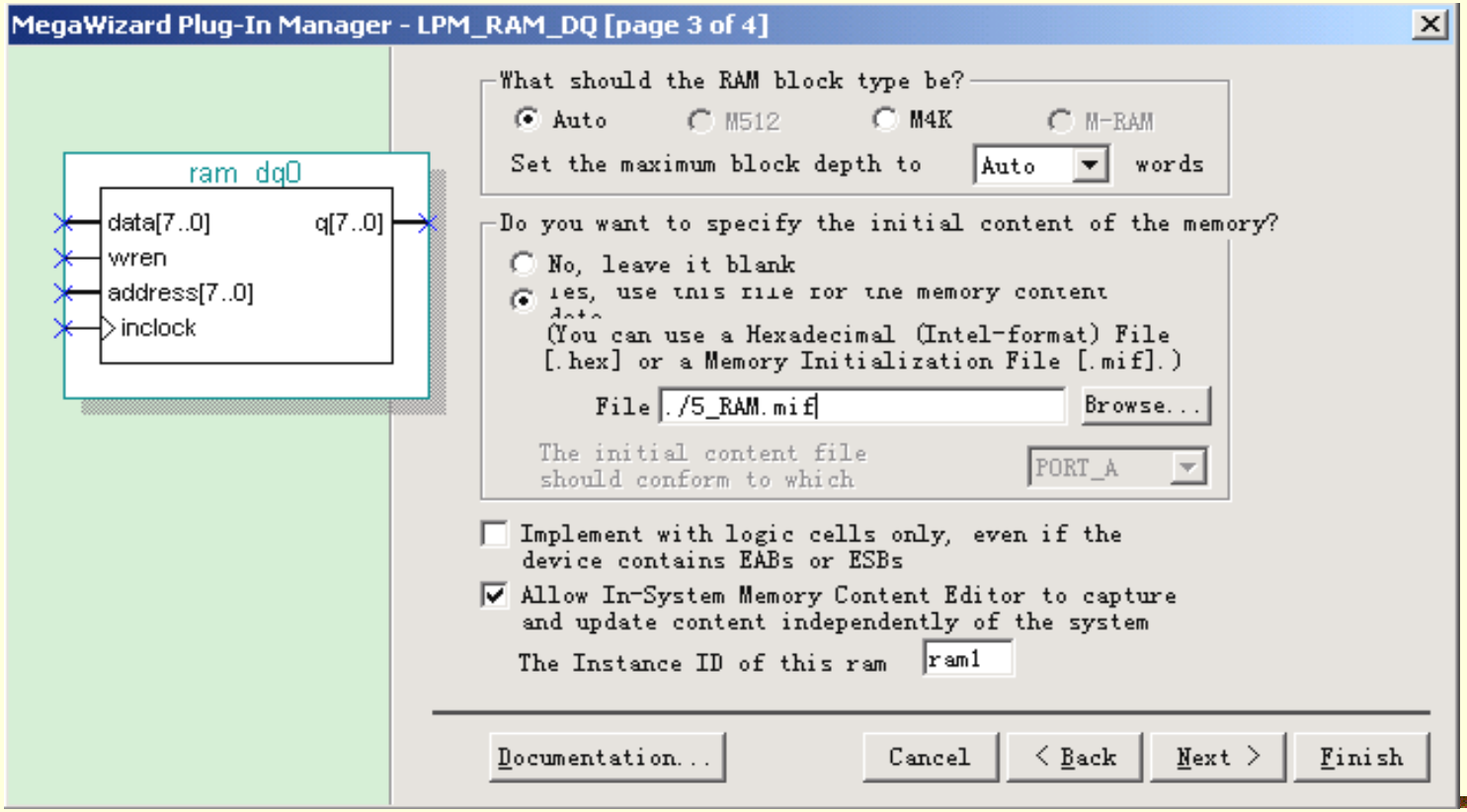

图4-58 lpm_ram_dq加入初始化文件和选择在系统读写RAM功能

实验与设计

实验4-5. LPM_RAM实验

Instance Manager: Ready to acquire

Index	Instance ID	Status	Width	Depth	Type	Mode
0	RAM4	Not running	8	256	RAM/ROM	Read/

JTAG Chain Configuration: JTAG ready

Hardware: ByteBlasterMV [LPT1] Setup...

Device: @1: EP1C6 (0x020820DD) Scan Chain

File: E:\CMPUT_EXPMT\CH6\EX4_2_RAM\RAM_DP1.sof

0 RAM4:

000000	11	12	13	14	15	16	17	18	19	21	22	23	24	25	26	27	28	29	30	21	21	33	34	35	.....!"#$%&'()0!!345
000018	36	37	38	39	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	6789.....
000030	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
000048	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
000060	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
000078	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
000090	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
0000A8	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
0000C0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
0000D8	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
0000F0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00									.....

图4-59 使用在系统读写RAM的工具对lpm_ram中的数据进行读写操作

实验与设计

实验4-6. LPM_FIFO实验

参考实验示例和实验课件：/CMPUT_EXPMT/CH4_Expt/DEMO_46_FIFO/ 和 实验4_6.ppt 。

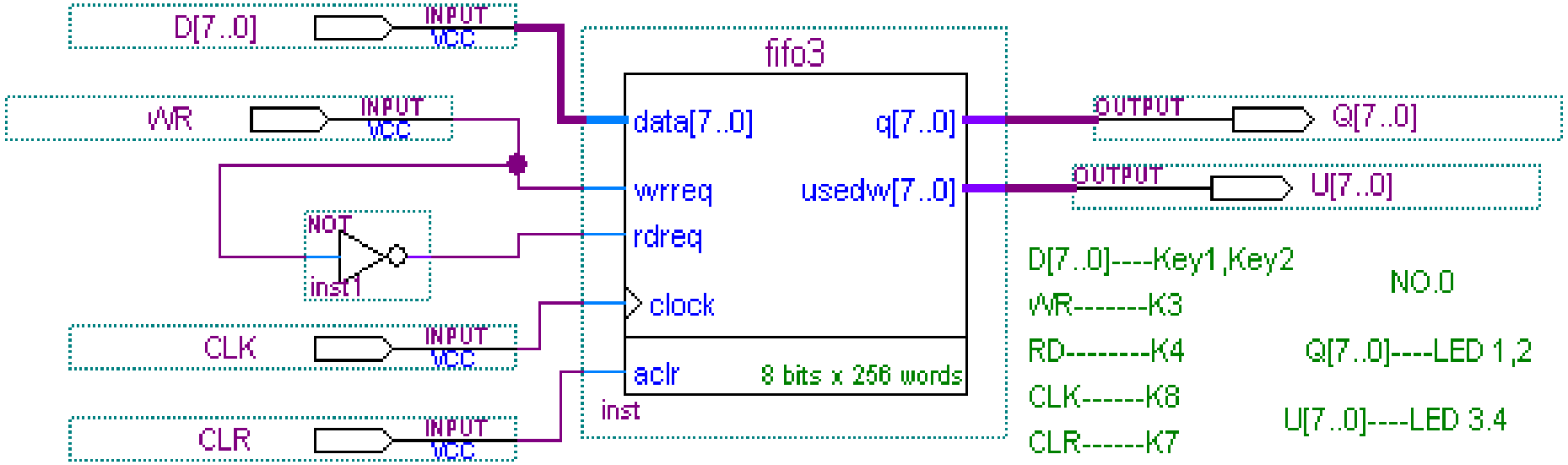

图4-60 lpm_fifo的实验结构图

实验与设计

实验4-6. LPM_FIFO实验

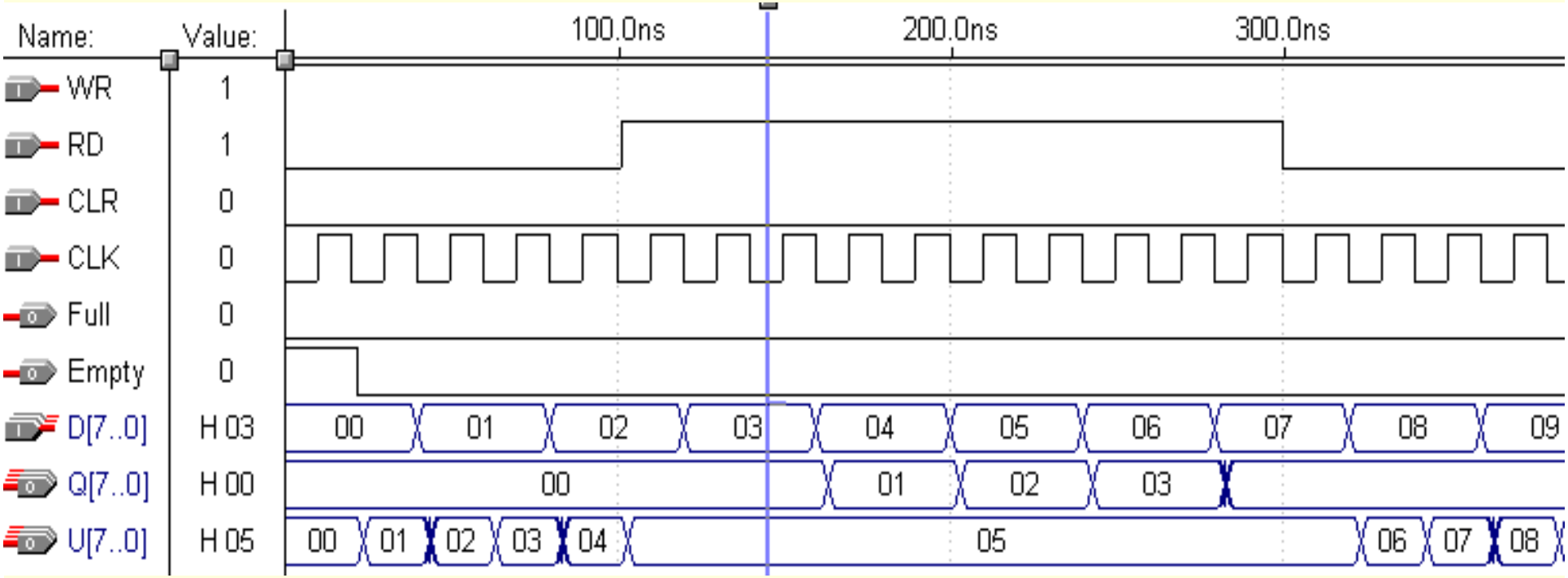

图4-61 lpm_fifo的仿真波形图

实验4-7. FPGA与外部16位RAM接口实验

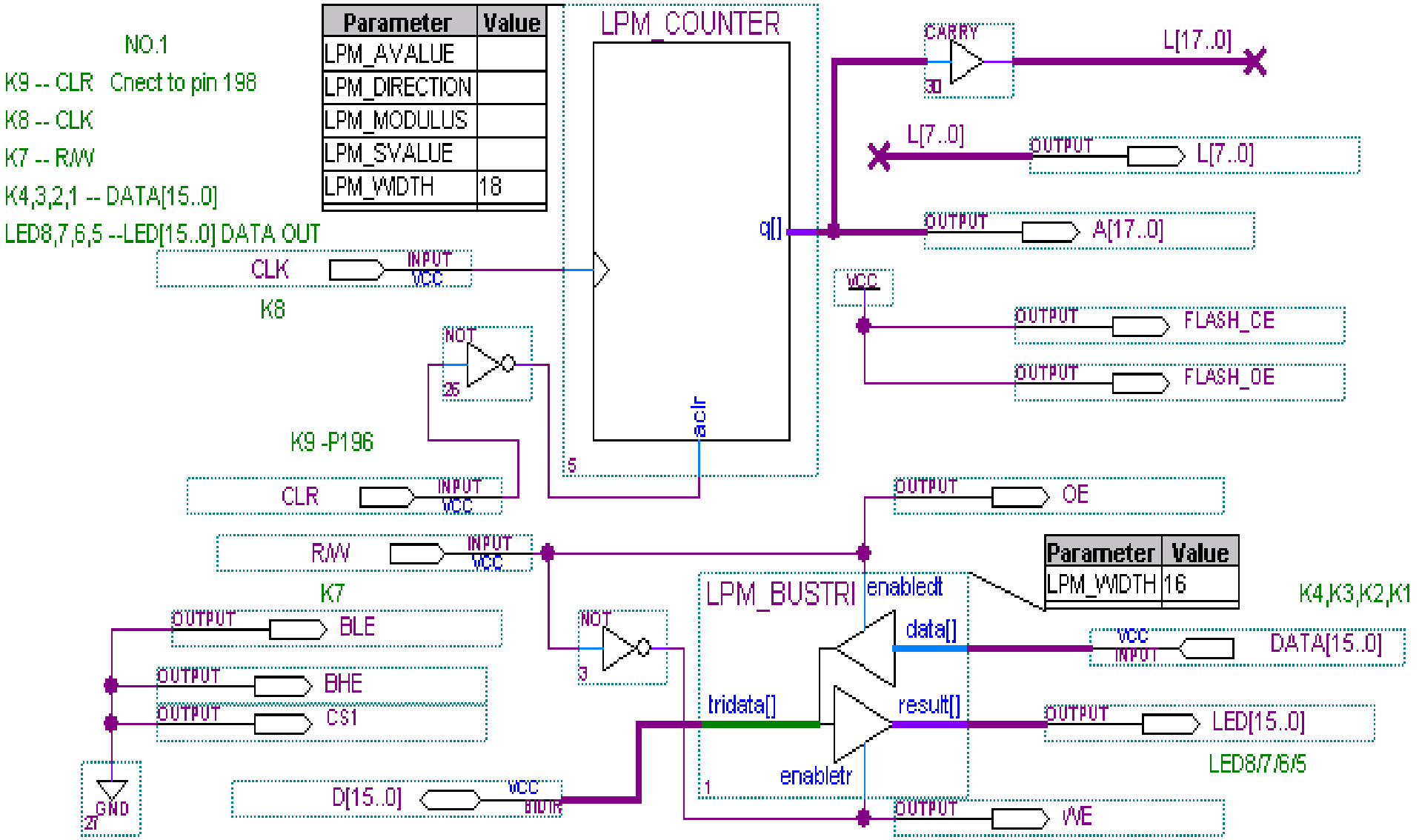

图4-62 FPGA与外部16位数据，18位地址线宽SRAM接口电路结构

实验4-7. FPGA与外部16位RAM接口实验

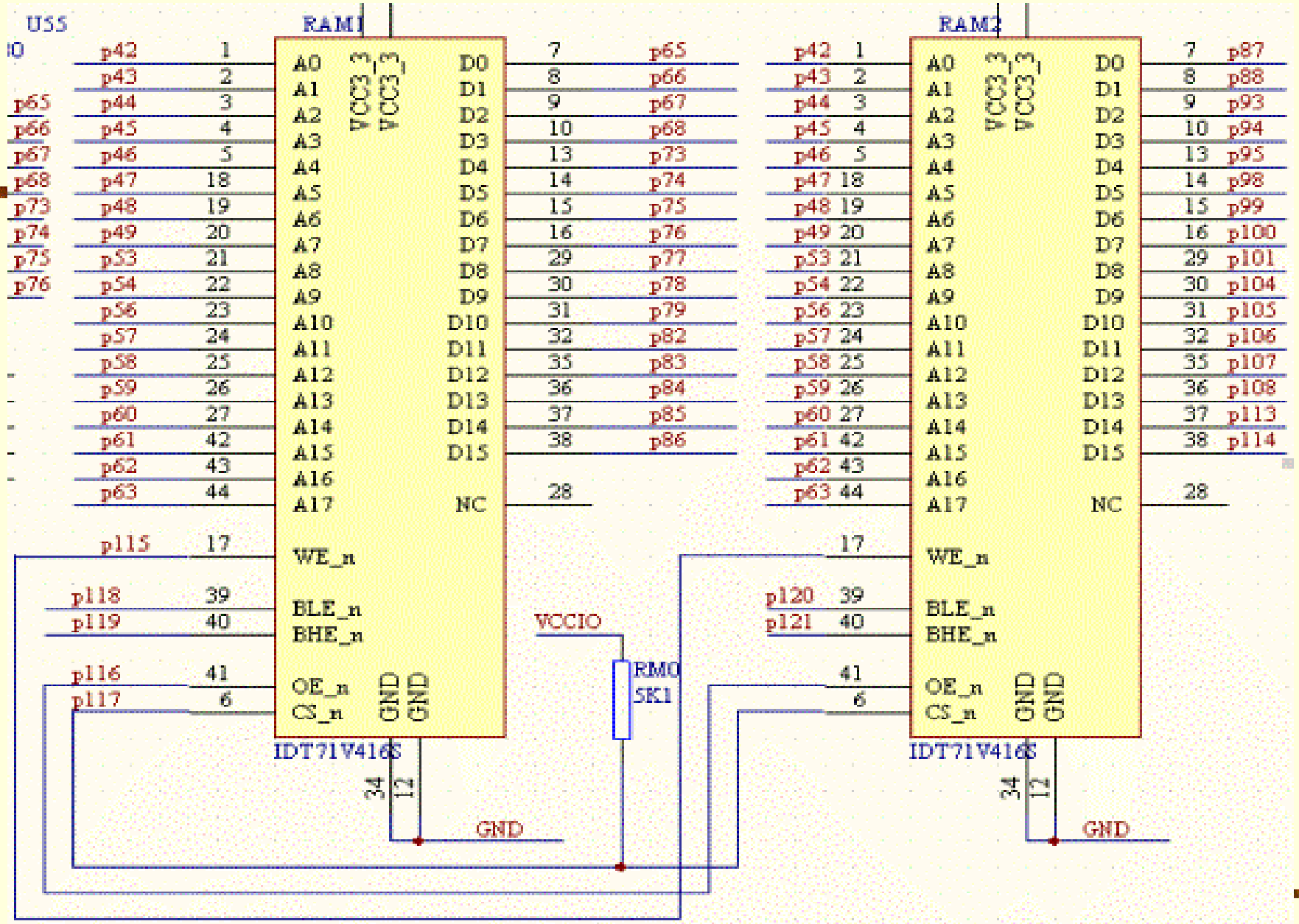

图4-63 16位SRAM IDT71V416电路原理图

实验与设计

实验4-8. 微控制器实验1　节拍脉冲发生器时序电路实验

(1)实验目的

掌握节拍脉冲发生器的设计方法，理解节拍脉冲发生器的工作原理。

(2)实验原理

(3)实验任务1：连续节拍发生电路设计

(4)实验任务2：单步节拍发生电路设计

(5)实验任务3：单步/连续节拍发生电路设计

(6) 思考题

实验与设计

实验4-9. 微控制器实验2：程序计数器PC与地址寄存器AR实验

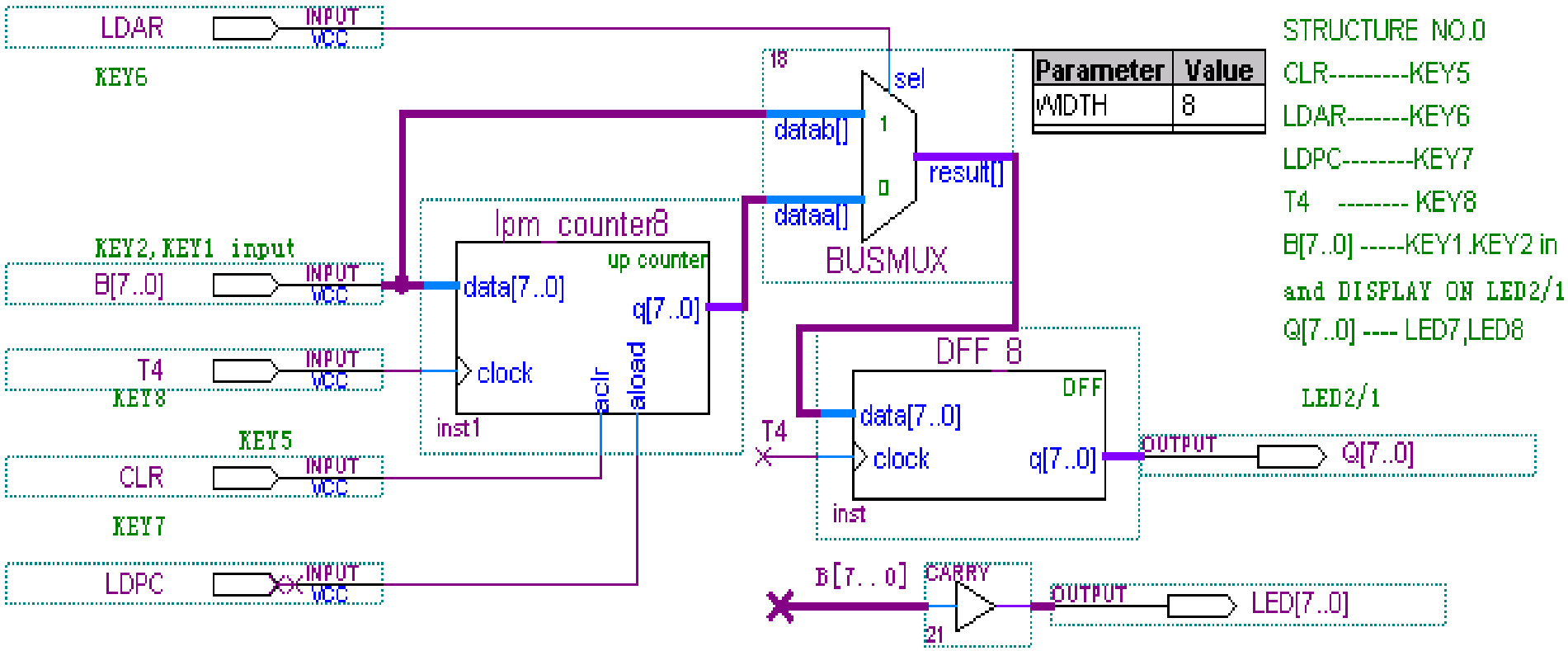

图4-64 程序计数器原理图

实验与设计

实验4-9. 微控制器实验2：程序计数器PC与地址寄存器AR实验

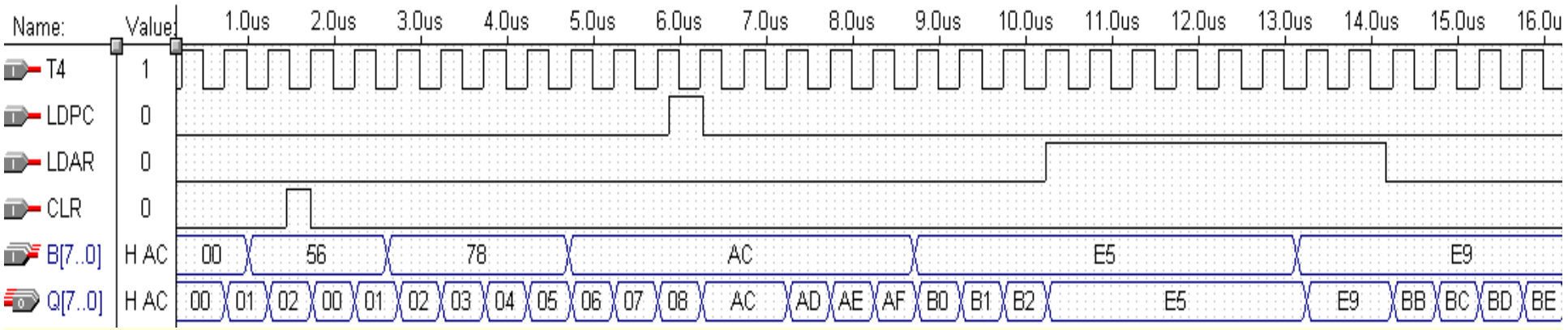

图4-65 程序计数器工作波形

实验与设计

实验4-9. 微控制器实验2：程序计数器PC与地址寄存器AR实验

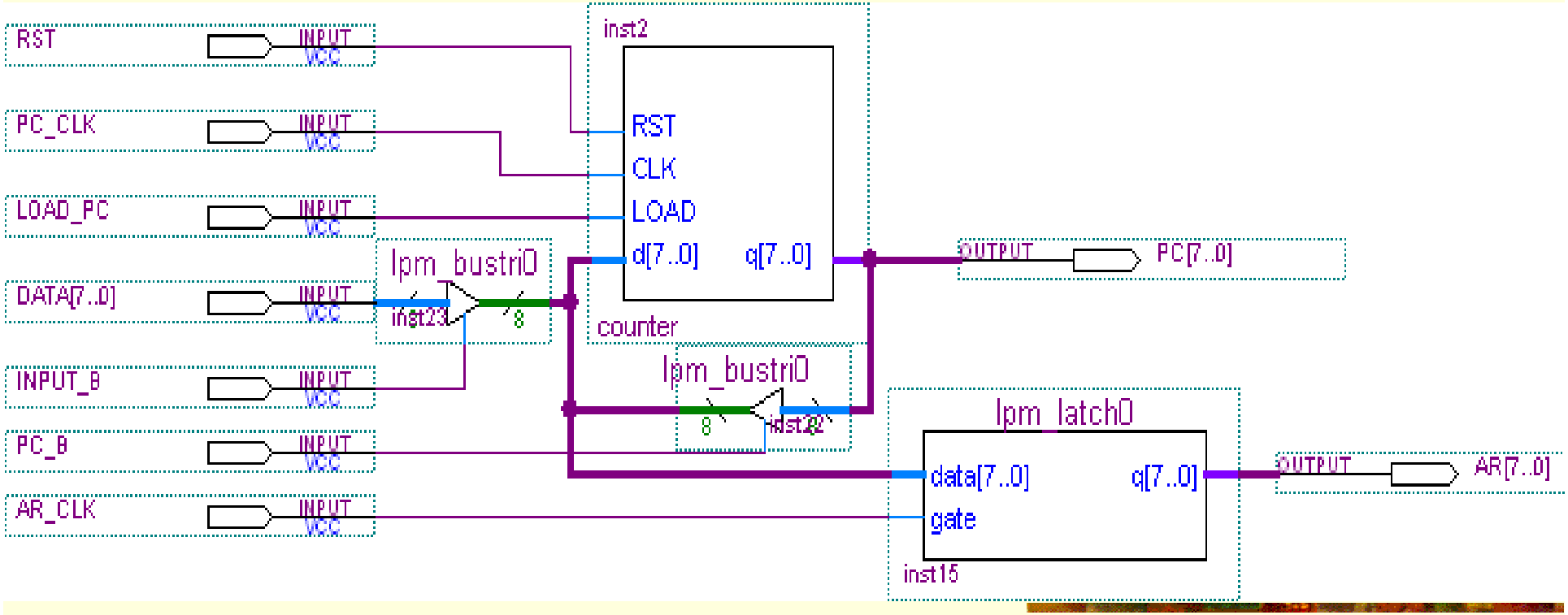

图4-66 程序计数器与地址寄存器

实验与设计

实验4-10. 微控制器实验3　　微控制器组成实验

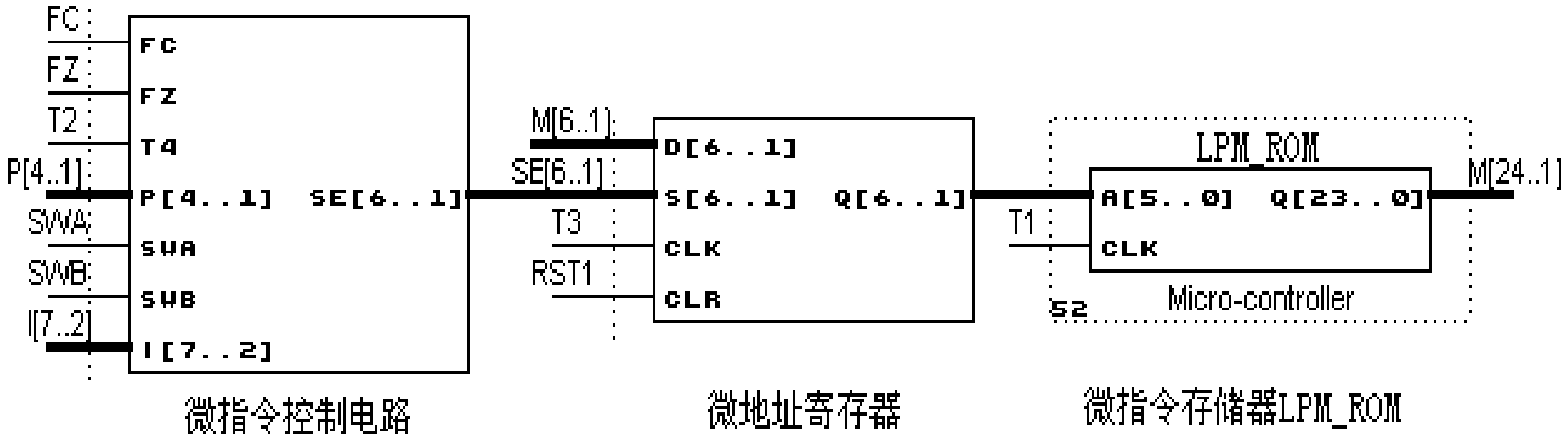

图4-67 微程序控制电路

实验4-10. 微控制器实验3　　微控制器组成实验

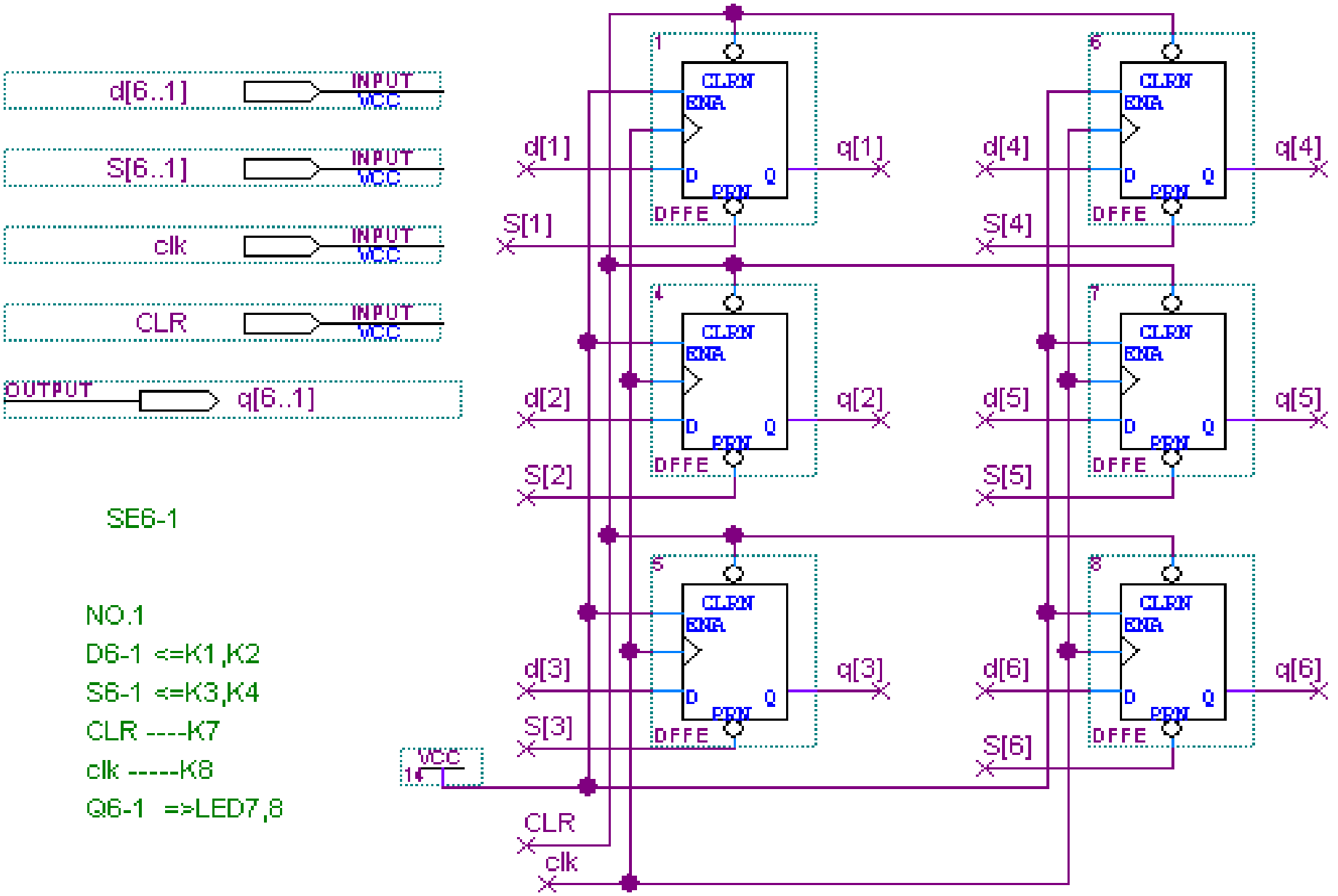

图4-68 微地址寄存器控制电路

实验与设计

实验4-10. 微控制器实验3　　微控制器组成实验

Addr	+0	+1	+2	+3	+4	+5	+6	+7
00	018108	00ED82	00C050	00A004	00E0A0	00E006	00A007	00E0A0
08	00ED8A	00ED8C	00A008	008001	062009	00A00E	01B60F	95EA25
10	00ED83	00ED85	00ED8D	00EDA6	001001	030401	018016	3D9A01
18	019201	01A22A	03B22C	01A232	01A233	01A236	318237	318239
20	009001	028401	05DB81	0180E4	018001	95AAA0	00A027	01BC28
28	95EA29	95AAA0	01B42B	959B41	01A42D	65AB6E	0D9A01	01AA30
30	0D8171	959B41	019A01	01B435	05DB81	B99B41	0D9A01	298838
38	019801	19883A	019801	070A08	062009	000000	000000	000000

图4-69 存储器初始化文件的内容

实验与设计

实验4-11. 正弦信号发生器设计

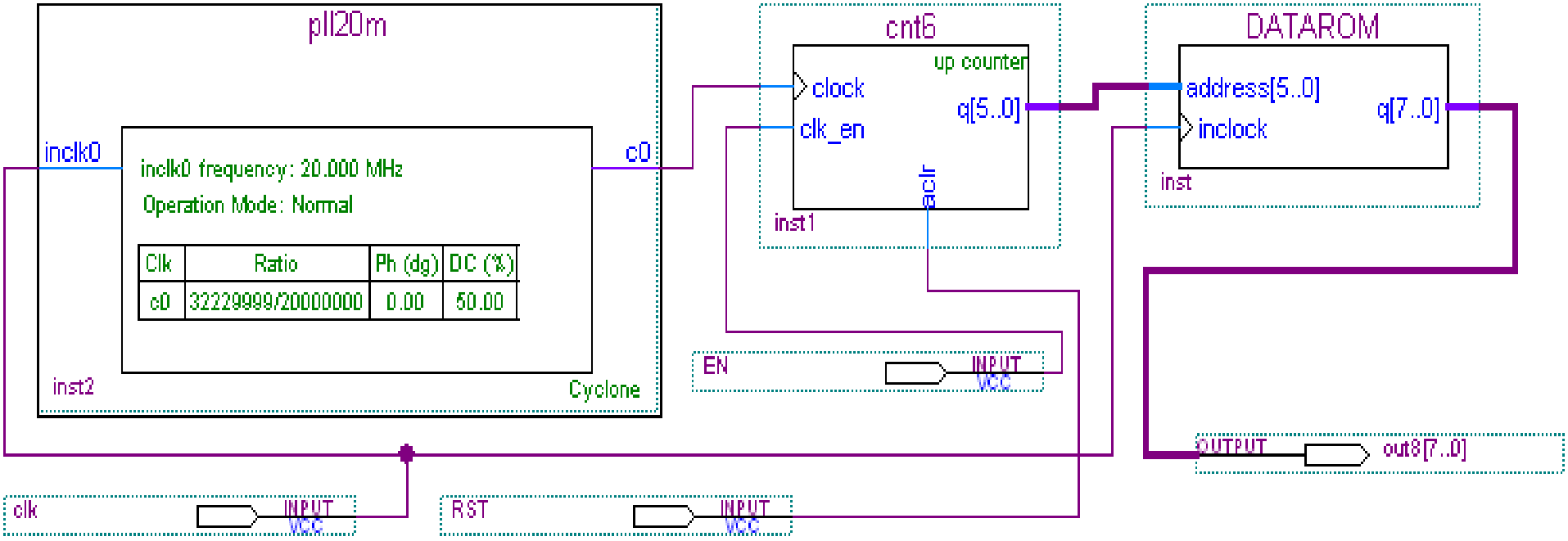

图4-52 调用了PLL元件信号发生器原理图