组合逻辑的控制单元的实现

计算机科学与技术 3 班

1140310323 蒲毅

一、实验目的：按照题要求用硬布线（组合逻辑）控制法设计一个简单模型机的控制单元 CU（微操作信号产生电路），决定外部的端口（名称、有效电平）和内部各元件的连接，画出系统框图和逻辑图，设计仿真数据，用 VHDL 编程和仿真。逻辑设计：主要元件设计：

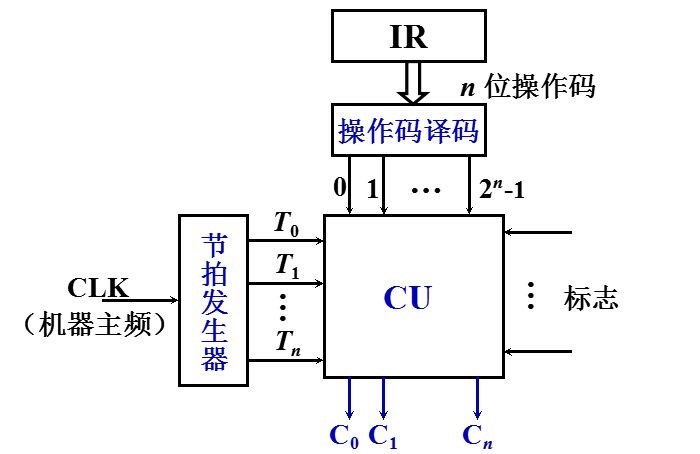
1．指令译码器

功能要求：4-10 译码器。

2．控制单元

功能要求：指令系统有 10 条不同类型的指令。包括：清除累加器指令CLA，累加器取反指令COM，算术右移一位指令SHR，循环左移一位指令CSL，停机指令STP，加法指令ADD X，存数指令STAX，取数指令LDAX，无条件转移指令JMPX，有条件转移（负则转）指令BAN X等。根据每条指令的功能和时序，分析其执行过程中需要在各个阶段产生的全部微操作，导出产生这些微操作控制信号的逻辑表达式（用积之和式表示）。并且能够正确产生 10 条不同指令在执行中（每个机器周期、每个节拍）发出的全部微操作。

系统设计框图：



I

9

…

I

0

…

Y

9

Y

8

Y

7

Y

6

Y

5

Y

4

Y

3

Y

2

Y1

Y0

4

-

10

译码器

S

A

3

A

2

A

1

A0

t0

q1

9

……

q0

m0

t1

控制单元

m1

t2

d

9

…

d0

m2

T

0

T

1

T

2

M0

M1

M2

Q1

9

…

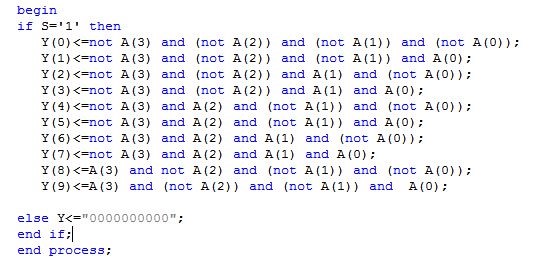
Q0

1.4-10译码器：原理图：

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Y0 | Y1 | Y2 | Y3 | Y4 Y5 Y6 Y7  4-10 译码器 | Y8 | Y9 |
|  | S |  | A3 | A2 A1 | A0 |  |

工作原理：

S 是工作信号，当 S=1 时，4-10 译码器工作；当 S=0 时，4-10 译码器不工作。A（0-3）是输入信号，Y（0-9）是输出信号。



2、控制单元（10 条指令）：

设计过程：将每条指令的实现分成取指令、分析指令、执行指令三个步骤，每一步由一个机器周期实现，一条指令的实现需要三个机器周期，即 M1（取指周期）、M2（分析周期）、 M3（执行周期）；每个机器周期由三个节拍组成，即 T0、T1、T2，如下图：机器周期

节拍

T

0

T

1

T

2

M

0

M

1

M

2

原理图：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| t0 |  | q19 ……q0 |  | m0 |
| t1 |  | 控制单元（10 指令） |  | m1 |
| t2 | d9 | … | d0 | m2 |

工作原理：

1.指令系统：非访存指令：

1. 清除累加器指令 CLA：
2. 累加器取反指令 COM：
3. 算术右移一位指令SHR：
4. 循环左移一位指令CSL：
5. 停机指令STP：访存指令：
6. 加法指令ADD X：
7. 存数指令STA X：
8. 取数指令LDA X：
9. 无条件转移指令JMP X：
10. 有条件转移（负则转）指令BAN X：

2.微指令的周期及节拍安排（微操作命令的操作时间表）：

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 工作周期标记 | 节拍 | 状态条件 | 微操作命令信号 | CL  A | C  O  M | S  H  R | CSL | ST  P | ADD | STA | LDA | JMP | BAN |
| FE  (  取指周期  )  M0 | T0 |  | PC->MAR | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 1->R | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| T1 |  | M(MAR)->MDR | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| (PC+1)->PC | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| T2 |  | MDR->IR | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| OP(IR)->ID | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| I | I->IND |  |  |  |  |  | 1 | 1 | 1 | 1 | 1 |
| -I | I->EX | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| IND  (  间址 | T1 |  | Ad(IR)->MAR |  |  |  |  |  | 1 | 1 | 1 | 1 | 1 |
| 1->R |  |  |  |  |  | 1 | 1 | 1 | 1 | 1 |
| T2 |  | M(MAR)->MDR |  |  |  |  |  | 1 | 1 | 1 | 1 | 1 |
|  |  | MDR->Ad(IR) |  |  |  |  |  | 1 | 1 | 1 | 1 | 1 |
| 周期  )  M1 | T3 | -  IND | 1->EX |  |  |  |  |  | 1 | 1 | 1 | 1 | 1 |
| EX  (  执行周期  )  M2 | T1 |  | Ad(IR)->MAR |  |  |  |  |  | 1 | 1 | 1 |  |  |
| 1->R |  |  |  |  |  | 1 |  | 1 |  |  |
| 1->W |  |  |  |  |  |  | 1 |  |  |  |
| T2 |  | M(MAR)->MDR |  |  |  |  |  | 1 |  | 1 |  |  |
| AC->MDR |  |  |  |  |  |  | 1 |  |  |  |
| T3 |  | (AC)+(MDR)  ->AC |  |  |  |  |  | 1 |  |  |  |  |
| MDR->M(MAR) |  |  |  |  |  |  | 1 |  |  |  |
| MDR->AC |  |  |  |  |  |  |  | 1 |  |  |
| 0->AC | 1 |  |  |  |  |  |  |  |  |  |
| -AC->AC |  | 1 |  |  |  |  |  |  |  |  |
| L(AC)->R(AC)  ,ACo不变 |  |  | 1 |  |  |  |  |  |  |  |
| P-1(AC) |  |  |  | 1 |  |  |  |  |  |  |
| Ad(IR)->PC |  |  |  |  |  |  |  |  | 1 |  |
| Ao | Ad(IR)->PC |  |  |  |  |  |  |  |  |  | 1 |
|  | 0->G |  |  |  |  | 1 |  |  |  |  |  |

每个微操作对应的逻辑表达式

0.PC->MAR=T0\*M0

1. 1->R=T0\*(M0+M1(S5+S6+S7+S8+S9)+M2(S5+S7))
2. M(MAR)->MDR=T1\*(M0+M1(S5+S6+S7+S8+S9)+M2(S5+S7))

3.(PC)+1->PC=M0\*T1

4.MDR->IR=T2\*(M0+M1(S5+S6+S7+S8+S9))

5.OP(IR)->ID=M0\*T2

1. I->IND=I\*M0\*T2\*(S5+S6+S7+S8+S9)
2. I->EX=T2\*(-I\*M0+M1(S5+S6+S7+S8+S9)\*(-IND))

8.Ad(IR)->MAR=T0\*(M1(S5+S6+S7+S8+S9)+M2(S5+S6+S7))

9.1->W=M2\*T0\*S6

10.AC->MDR=M2\*T1\*S5

11.(AC)+MDR->AC=M2\*T2\*S5

12.MDR->M(MAR)=M2\*T2\*S6

13.MDR->AC=M2\*T2\*S7 14.0->AC=M2\*T2\*S0

15.-AC->AC=M2\*T2\*S1

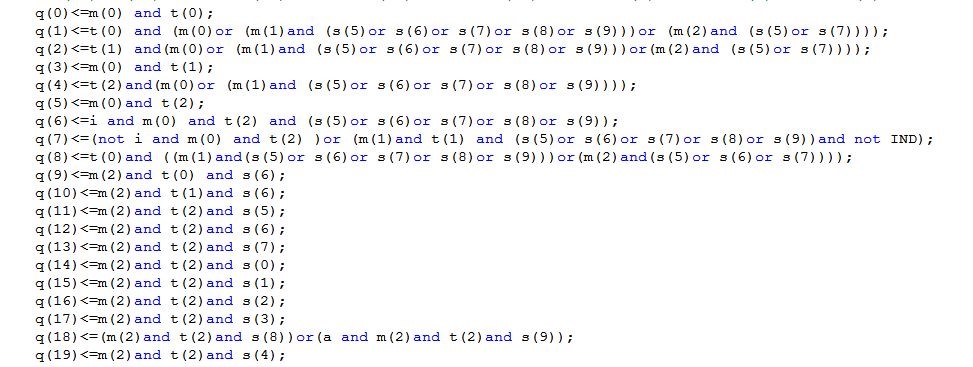
16.L(AC)->R(AC),ACo不变=M2\*T2\*S2

17.P-1(AC)=M2\*T2\*S3

18.Ad(IR)->PC=M2\*T2\*(S8+Ao\*S9)

19.0->G=M2\*T2\*S4

注：S0-9表示表格内十条指令（从左到右）；-X表示X取反。指令系统对应代码：



VHDL代码： 4-10译码器： library IEEE; use IEEE.STD\_LOGIC\_1164.ALL; entity decode is

Port ( S : in STD\_LOGIC;

A : in STD\_LOGIC\_VECTOR (3 downto 0);

Y : out STD\_LOGIC\_VECTOR (9 downto 0)); end decode; architecture Behavioral of decode is begin

process(S) begin if S='1' then

Y(0)<=not A(3) and (not A(2)) and (not A(1)) and (not A(0));

Y(1)<=not A(3) and (not A(2)) and (not A(1)) and A(0);

Y(2)<=not A(3) and (not A(2)) and A(1) and (not A(0)); Y(3)<=not A(3) and (not A(2)) and A(1) and A(0);

Y(4)<=not A(3) and A(2) and (not A(1)) and (not A(0));

Y(5)<=not A(3) and A(2) and (not A(1)) and A(0);

Y(6)<=not A(3) and A(2) and A(1) and (not A(0));

Y(7)<=not A(3) and A(2) and A(1) and A(0);

Y(8)<=A(3) and not A(2) and (not A(1)) and (not A(0));

Y(9)<=A(3) and (not A(2)) and (not A(1)) and A(0);

else Y<="0000000000"; end if; end process; end Behavioral; CU\_central单元：

library IEEE; use IEEE.STD\_LOGIC\_1164.ALL;

entity CU\_central is

Port ( m:in std\_logic\_vector(2 downto 0); i:in std\_logic;

a:in std\_logic;

IND:in std\_logic; t:in std\_logic\_vector(2 downto 0); d:in std\_logic\_vector(9 downto 0); q:out std\_logic\_vector(19 downto 0)); end CU\_central;

architecture behave of CU\_central is signal s:std\_logic\_vector(9 downto 0);

begin s(0)<=(not d(9)) and (not d(8))and (not d(7)) and (not d(6)) and (not d(5))

and (not d(4)) and (not d(3)) and (not d(2)) and (not d(1)) and d(0); s(1)<=notd(9)andnotd(8)and(notd(7))and(notd(6))and(notd(5))and( not

d(4)) and (not d(3)) and (not d(2)) and( d(1)) and (not d(0)); s(2)<=not d(9) and not d(8)and not d(7) and not d(6) and not d(5) and not d(4)

and not d(3) and d(2) and not d(1) and not d(0); s(3)<=not d(9) and not d(8)and not d(7) and not d(6) and not d(5) and not d(4)

and d(3) and not d(2) and not d(1) and not d(0); s(4)<=not d(9) and not d(8)and not d(7) and not d(6) and not d(5) and d(4) and

not d(3) and not d(2) and not d(1) and not d(0); s(5)<=not d(9) and not d(8)and not d(7) and not d(6) and d(5) and not d(4) and

not d(3) and not d(2) and not d(1) and not d(0); s(6)<=not d(9) and not d(8)and not d(7) and d(6) and not d(5) and not d(4) and

not d(3) and not d(2) and not d(1) and not d(0); s(7)<=not d(9) and not d(8)and d(7) and not d(6) and not d(5) and not d(4) and not d(3) and not d(2) and not d(1) and not d(0);

s(8)<= not d(9) and d(8)and not d(7) and not d(6) and not d(5) and not d(4)

and not d(3) and not d(2) and not d(1) and not d(0); s(9)<= d(9) and not d(8)and not d(7) and not d(6) and not d(5) and not d(4)

and not d(3) and not d(2) and not d(1) and not d(0);

-- s(12)<=not d(13) and d(12) and not d(11) and not d(10) and not d(9) and not d(8)and not d(7) and not d(6) and not d(5) and not d(4) and not d(3) and not d(2) and not d(1) and not d(0);

-- s(13)<=d(13) and not d(12) and not d(11) and not d(10) and not d(9) and not d(8)and not d(7) and not d(6) and not d(5) and not d(4) and not d(3) and not d(2) and not d(1) and not d(0); q(0)<=m(0) and t(0);

q(1)<=t(0)and(m(0)or(m(1)and(s(5)ors(6)ors(7)ors(8)ors(9)))or(m(2)and

(s(5)or s(7))));

q(2)<=t(1) and(m(0)or (m(1)and (s(5)or s(6)or s(7)or s(8)or s(9)))or(m(2)and

(s(5)or s(7))));

q(3)<=m(0) and t(1); q(4)<=t(2)and(m(0)or (m(1)and (s(5)or s(6)or s(7)or s(8)or s(9)))); q(5)<=m(0)and t(2); q(6)<=i and m(0) and t(2) and (s(5)or s(6)or s(7)or s(8)or s(9)); q(7)<=(notiandm(0)andt(2))or(m(1)andt(1)and(s(5)ors(6)ors(7)ors(8)or

s(9))and not IND); q(8)<=t(0)and ((m(1)and(s(5)or s(6)or s(7)or s(8)or s(9)))or(m(2)and(s(5)or

s(6)or s(7)))); q(9)<=m(2)and t(0) and s(6); q(10)<=m(2)and t(1)and s(6); q(11)<=m(2)and t(2)and s(5); q(12)<=m(2)and t(2)and s(6); q(13)<=m(2)and t(2)and s(7); q(14)<=m(2)and t(2)and s(0); q(15)<=m(2)and t(2)and s(1); q(16)<=m(2)and t(2)and s(2); q(17)<=m(2)and t(2)and s(3); q(18)<=(m(2)and t(2)and s(8))or(a and m(2)and t(2)and s(9));

q(19)<=m(2)and t(2)and s(4);

end behave;

总的控制单元：

library IEEE; use IEEE.STD\_LOGIC\_1164.ALL; entity HCU is

Port (

S : in STD\_LOGIC;

A : in STD\_LOGIC\_VECTOR (3 downto 0);

M : in STD\_LOGIC\_VECTOR (2 downto 0);

T : in STD\_LOGIC\_VECTOR (2 downto 0); I:in std\_logic; aa:in std\_logic;

IND:in std\_logic;

Q : out STD\_LOGIC\_VECTOR (19 downto 0)); end HCU;

architecture mygod of HCU is signal e:std\_logic\_vector(9 downto 0);

component decode port( S:in std\_logic;

A:in std\_logic\_vector(3 downto 0);

Y:out std\_logic\_vector(9 downto 0)); end component; component CU\_central port( m:in std\_logic\_vector(2 downto 0); i:in std\_logic; a:in std\_logic;

IND:in std\_logic; t:in std\_logic\_vector(2 downto 0); d:in std\_logic\_vector(9 downto 0); q:out std\_logic\_vector(19 downto 0));

end component; begin

G1:decode port map

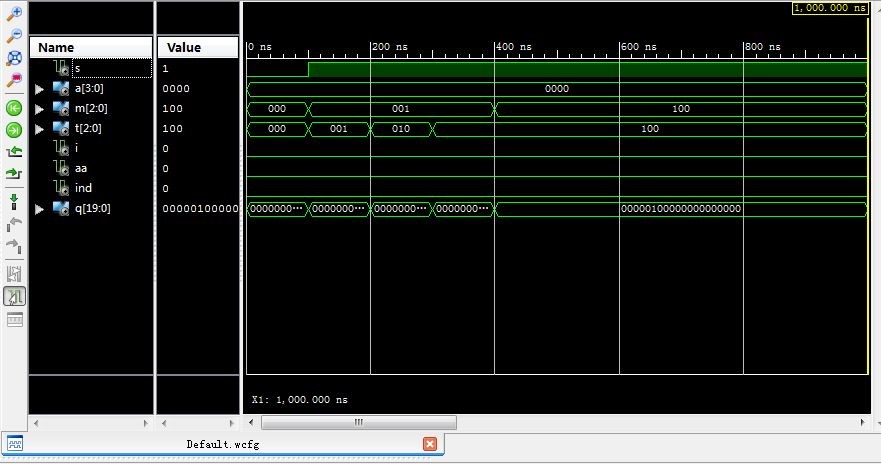
(S=>S,A=>A,Y=>e);

G2:CU\_central port map

(d=>e,t=>T,m=>M,q=>Q,IND=>IND,i=>i,a=>aa); end mygod;

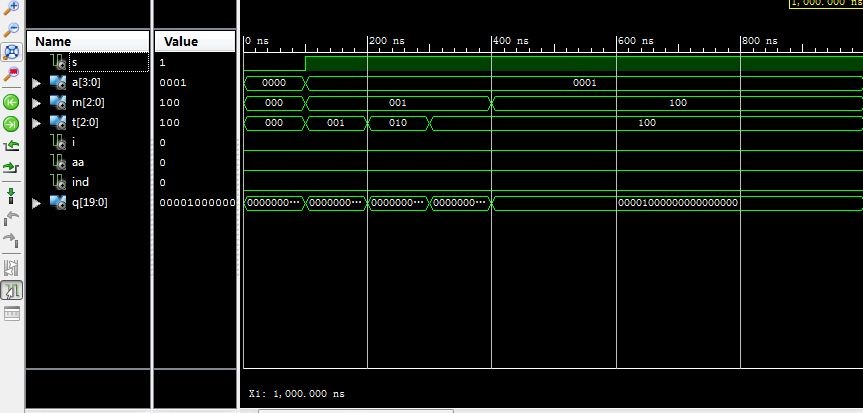
三、仿真设计仿真结果和说明：指令系统：

1.清除累加器指令CLA：



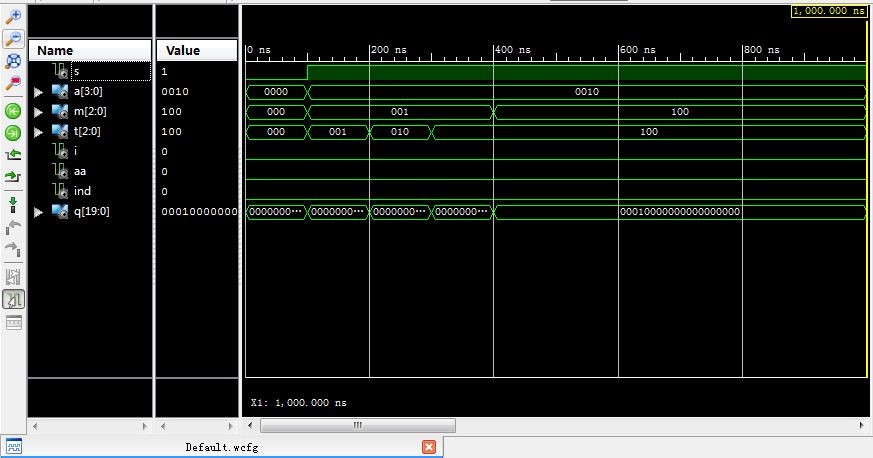
由微操作命令的操作时间表中可知，该指令最终结果q14=1，其余为0，与仿真结果相符。

2.累加器取反指令COM：



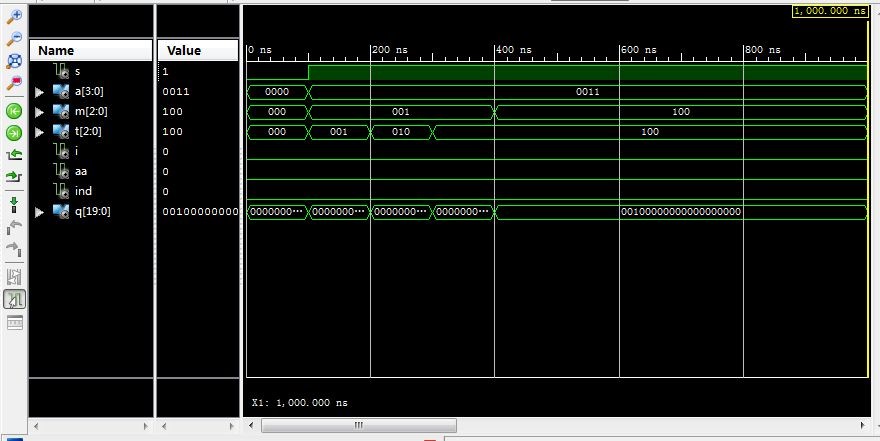
由微操作命令的操作时间表中可知，该指令最终结果q15=1，其余为0，与仿真结果相符。

3.算术右移一位指令SHR：



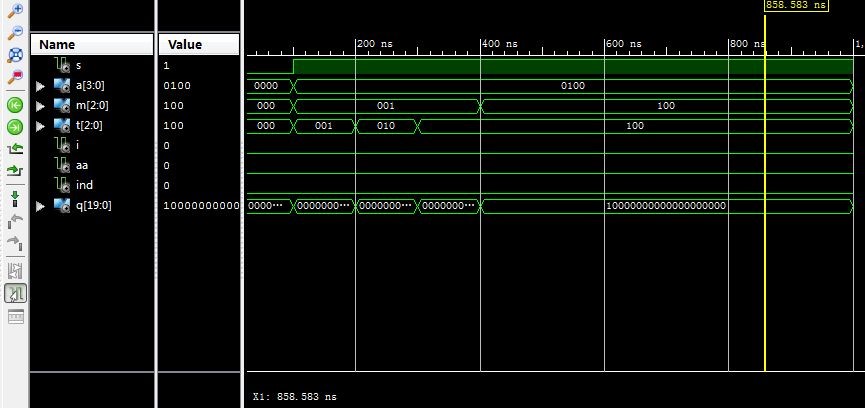
由微操作命令的操作时间表中可知，该指令最终结果q16=1，其余为0，与仿真结果相符。

4.循环左移一位指令CSL：



由微操作命令的操作时间表中可知，该指令最终结果q16=1，其余为0，与仿真结果相符。

5.停机指令STP：



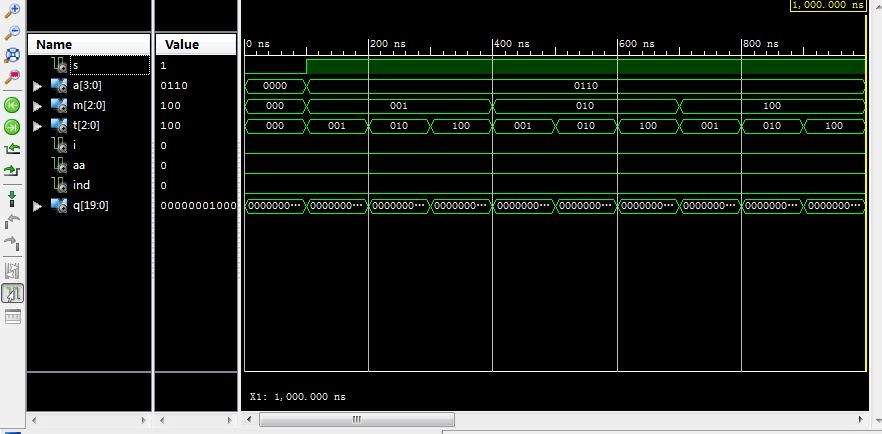
由微操作命令的操作时间表中可知，该指令最终结果q19=1，其余为0，与仿真结果相符。

6.加法指令ADD X：



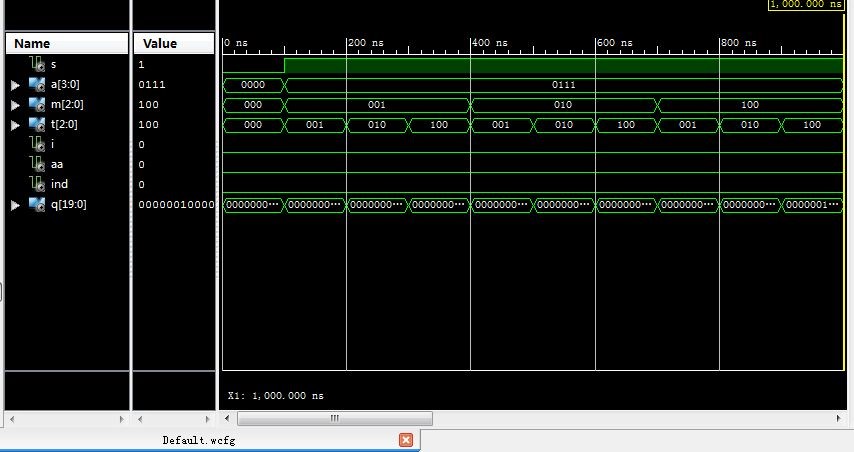
由微操作命令的操作时间表中可知，该指令最终结果q11=1，其余为0，与仿真结果相符。

7.存数指令STA X：



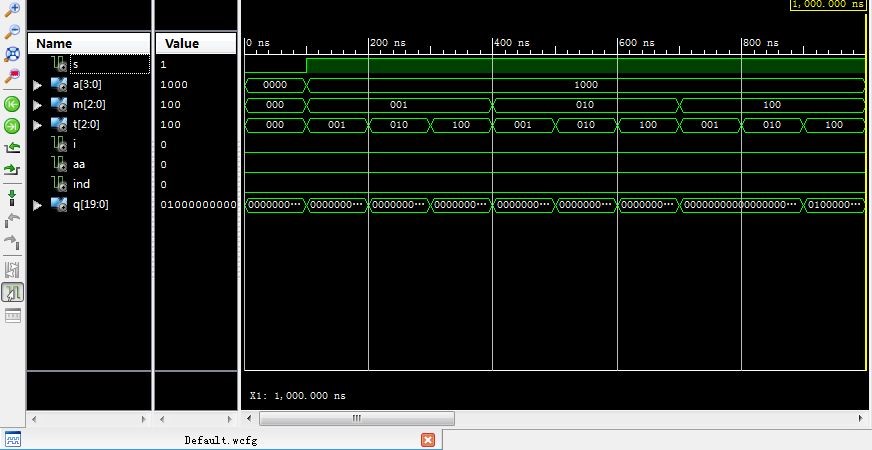
由微操作命令的操作时间表中可知，该指令最终结果q12=1，其余为0，与仿真结果相符。

8.取数指令LDA X：



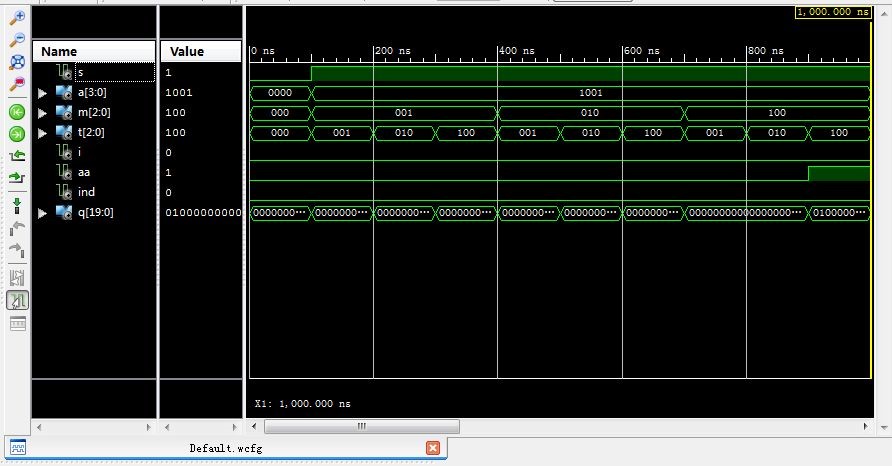
由微操作命令的操作时间表中可知，该指令最终结果q13=1，其余为0，与仿真结果相符。

9.无条件转移指令JMP X：



由微操作命令的操作时间表中可知，该指令最终结果q18=1，其余为0，与仿真结果相符。

10.有条件转移（负则转）指令BAN X：



由微操作命令的操作时间表中可知，该指令最终结果q18=1，其余为0，与仿真结果相符。