A Combinational Logic Controller Based on Proteus
Da-wei XIAO and Tian-yu HE
City institute, Dalian university of technology, Dalian, 116600, China

Keywords: Proteus, Keil, Combinational logic controller, Computer organization.

Abstract. Proteus and Keil are used to implement a combinational logic controller. Firstly, the instruction system, instruction flow and control signal of the controller are designed. Secondly, the logical expressions of each control signal are listed and simplified. Then, the circuit schematic diagram of the combinational logic controller is drawn by Proteus software. Finally, using Keil, the program is loaded and run in the controller, which is verified by simulation. This design uses logic devices, improves the execution speed of instructions, and enhances the flexibility of design. In the experiment teaching of Computer Organization course, it can improve students' thinking ability and practical ability well.

Introduction

The controller is the command and control center of the computer. According to different design methods, it is divided into combinational logic controller and microprogram controller [1]. The combinational logic controller consists of instruction decoding circuit, sequential circuit and combinational logic circuit. The instruction decoder determines the currently executed instructions, combines the beat generated by the sequential circuit, and jointly acts as the input of the combinational logic circuit, and outputs the corresponding control signal. Combinational logic controllers are composed of complex combinational logic gates and flip-flops, which have fast execution speed. Therefore, they have been widely used in new computer structures such as RISC in recent years [2].

Machine Instruction Design

Instruction Code and Functions

This controller can realize direct and indirect addressing, so four instructions of NOP, JMP1, JMP2 and HLT are designed. The instruction code and its functions are shown in Table 1.

| Instruction | Instruction Function | Opcode | Instruction Code | Instruction Code Length |
|-------------|----------------------|--------|------------------|------------------------|
| NOP         | No operation         | 000    | 000 00000        | 1B                     |
| JMP1        | Jump instruction for direct addressing, Executes "addr→PC" | 001 | 001 00000 addr | 2B                     |
| JMP2        | Jump instruction for indirect addressing, Executes "[addr]→PC" | 010 | 010 00000 addr | 2B                     |
| HLT         | Machine halts       | 111    | 111 00000        | 1B                     |

As can be seen from Table 1, NOP instructions do not execute any operations, JMP1 executes "addr→PC" operation, JMP2 executes "[addr]→PC" operation, HLT instruction directly halts machines through hardware. The highest three bits of instructions are opcodes. That is, the opcodes of NOP, JMP1, JMP2 and HLT are 000, 001, 010 and 111, respectively. Since the execution of NOP and HLT instructions does not involve operands, the instruction codes of this two instructions occupy 8 bits, five "0" are added after the highest three bits opcodes, so the instruction codes of this two instructions are "00000000" and "11100000" respectively. JMP1 and JMP2 instructions involve
jumped address, so the length of instruction code is designed to be 16 bits, the high 8-bits are "00100000" and "01000000" respectively, and the low 8-bits are "addr".

**Instruction Flow**

During the execution of these four machine instructions, the largest number of States is JMP2 instructions with three states. The controller adopts a single-period structure, by this standard, the state machine of the controller consists of three states: S1, S2 and S3. In each state, the operation of "Source Component→Bus→Target Component" is executed. So, in each state, the two beats are designed, which are recorded as T1 and T2. The operation is "Source Part→Bus" in T1 and the operation is "Bus→Target Component" in T2 [3].

Thus, the state flow chart of the four machine instructions execution process is shown in Figure 1.

**Micro-command Design**

As shown in Figure 1, the following micro-commands are required to execute four instructions: NOP, JMP1, JMP2 and HLT (Bus has no input and output control signals):

PC→AR: This operation requires two micro-commands, one is PC output, recorded as PCOUT and the other is AR loading, recorded as LDAR.

ROM→BUS: This operation requires a micro-command, which is ROM output, and the ROM output control signal is usually low level effective, so the ROM output is recorded as /OE.

BUS→IR: This operation requires a micro-command, which is IR loading, recorded as LDIR.

PC+1→PC: This operation requires a micro-command, which is PC plus 1, recorded as PC_INC.

BUS→PC: This operation requires a micro-command, which is PC loading. The signal is also low level effective, so it recorded as /LDPC.

As can be seen from Figure 1, the micro-command signal is the result of "and" or "OR" operation between the current machine instruction signal and the specific beat signal. In all the following expression, "*" is AND operation, "+" is OR operation).

Table 2 shows the states associated with each micro-command signal.

| Micro-command | NOP/HLT | JMP1 | JMP2 |
|---------------|---------|------|------|
| LDAR          | S1      | S1, S2 | S1, S2, S3 |
| /OE           | S1      | S1, S2 | S1, S2, S3 |
| LDIR          | S1      | S1    | S1    |
| PC_INC        | S1      | S1, S2 | S1, S2, S3 |
| /LDPC         | S2      | S2, S3 |

Figure 1. Instruction Flow Figure.
The logical expressions of these micro-command refer with Eq. 1~3.

\[
\text{LDAR} = /\text{OE} = \text{PC\_INC} = S_1 + (\text{JMP1} + \text{JMP2}) \times S_2 + \text{JMP2} \times S_3. \quad (1)
\]

\[
\text{LDIR} = S_1. \quad (2)
\]

\[
/L\text{DPC} = (\text{JMP1} + \text{JMP2}) \times S_2 + \text{JMP2} \times S_3. \quad (3)
\]

**Combinational Logic Controller Circuit Design**

The circuit of the controller consists of data path, instruction decoding circuit, sequential circuit and combinational logic circuit.

**Data Path and Instruction Decoding Circuit**

This part of the circuit consists of program counter PC, address register AR, program memory ROM, instruction register IR and instruction decoder.

Both AR and IR are composed of latch 74LS273. AR is used to store the address of the access program memory. IR is used to store information such as instruction codes or addresses read from program memory.

ROM is composed of 2764 chips for storing machine language programs. There are two green tubes on the left to display instruction codes or addresses read from memory, and two red tubes on the right to display addresses currently accessing memory.

PC is composed of 74LS163 which is a 4-bit binary synchronous counter. ENP, ENT, MR, CLK, LOAD, D3-D0 are the input and Q3~Q0 are the output.

Since LDIR, PC\_INC and LDAR are CLK signals and CLK signals are edge signals, so the three signals must be combined with the beat signal T1 or T2 of the specified period to produce new edge trigger signals, which are recorded as IR\_CLK, PC\_CLK and AR\_CLK. The logical expressions of these CLK signals refer with Eq. 4~6.

\[
\text{IR\_CLK} = \text{LDIR} \times T_2. \quad (4)
\]

\[
\text{PC\_CLK} = \text{PC\_INC} \times T_2. \quad (5)
\]

\[
\text{AR\_CLK} = \text{LDAR} \times T_1. \quad (6)
\]

Instruction decoder is composed of 74LS138 which receives opcodes (IR7~IR5) of instructions and generates valid signals corresponding to four instructions after decoding.

Data Path and Instruction Decoding Circuit are shown in Figure 2.

**Sequential Circuit**

The sequential circuit consists of reset switch, clock signal source and beat sequence.

The reset switch is realized by a dial switch. When dialed to a high level state, the ON signal can be validated by 74LS74 trigger. S, R, CLK and D are the inputs of 74LS74 and Q and /Q are the outputs of 74LS74.

There are two kinds of clock signal sources: automatic signal source AUTO\_CLK and manual signal source MANUAL\_CLK. The automatic signal source can output a continuous clock signal, while the manual signal source only outputs a clock signal at each switch press. In order to observe the results of each step of the application, the manual signal source is used to realize the one-step operation of the application.

The CLK signal of the controller is controlled by MANUAL\_CLK, ON and HLT. HLT stands for stop instruction. Only when IR7~IR5 = 111, the HLT signal is a low level, so the HLT signal is controlled by IR7~IR5 through NAND gate.
In the beat sequence, T1 and T2 beats are generated by a 74LS74 flip-flop, and the state signals S1～S3 are generated by 74LS74 flip-flop, 74LS139 decoder and 74LS240 reverse output driver. The sequence circuit diagram is shown in Figure 3.

**Combinational Logic Circuit**

In this part of the circuit, NOP, JMP1, JMP2 and HLT command signals are generated by 74LS138 decoder, which are all low level effective signals. Then, the simulation circuit diagram is drawn according to the micro-command logic expression (refer with: Eq. 1～6). The combined logic circuit is shown in Figure 4.
Simulation and Verification

Application Design

The machine instruction code is 8 bits, each piece 2764 is also 8 bits, so the ROM is composed of one piece 2764. Keil software is used to store the following code in ROM:

```
ORG 0000H
DB 20H ; JMP1 03H
DB 03H
DB 00H ; NOP
DB 40H ; JMP2 07H
DB 07H
DB 0E0H ; HLT
DB 00H ; NOP
DB 05H ; Addr
END
```

Start Simulation

Put the CLK switch to the MANUAL_CLK position. First set RESET=0, then RESET=1, therefore the CLK starts to output and enters the S1 state and T1 beat.

In T1 beat of S1 state, effective micro-commands are /OE and LDIR. Therefore, ROM outputs 20H stored in 00H address unit. At this time, red digital tube displays 00H and green digital tube displays 20H, which is the first instruction of the application. At the moment, IR_CLK is still invalid, so even if the green digital tube in the data path displays 20H, it means that the opcode part of the "JMP1 03H" instruction has been taken out, in fact, the op-code has not been written to IR.

Run Application

After starting the simulation, every time the MANUAL_CLK switch closes and disconnects again, it outputs a beat, which is circulated in sequence of S1T1, S1T2, S2T1, S2T2, S3T1 and S3T2. Each cycle, an instruction is executed. Table 3 records the order of instruction execution. When the HLT instruction is executed, hardware shutdown occurs. At this time, a rising edge must be given to the RESET signal to resume the operation of the application.

Figure 4. Combinational Logic Circuit.
Table 3. Instruction running sequence record table.

| Sequence Number | Red Digital Tube (Display Address) | Green Digital Tube (Display Instruction Code) | Currently Executed Instructions |
|------------------|-----------------------------------|---------------------------------------------|---------------------------------|
| 1                | 00H                               | 20H                                         | JMP 03H                         |
| 2                | 01H                               | 03H                                         |                                 |
| 3                | 03H                               | 40H                                         | JMP 07H                         |
| 4                | 04H                               | 07H                                         |                                 |
| 5                | 07H                               | 05H                                         |                                 |
| 6                | 05H                               | E0H                                         | HLT                             |

Summary

This paper completes the design of a combinational logic controller, which realizes the functions of direct and indirect addressing. The controller is designed by Proteus, which makes the controller easy to modify, enhances the design flexibility and improves the execution speed of instructions. In the future, more executable instructions will be added to strengthen the function of the controller.

References

[1] Liqiu Jiang, Dawei Xiao, Computer Composition and Design, Dalian University Of Technology, Da Lian, 2009.

[2] Yan Gao, Kun Fu, Shuzhen Luo, etc., Combinational Logic Controller in the Model Machine Based on FPGA, Laboratory Research and Exploration. 34 (2015) 86-91.

[3] Xiaozheng Lai, Computer System Experiment Course Based on Proteus, China Machine Press, Bei Jing, 2017.