

第3章 系统仿真

刻鹄类鹜 《后汉书·马援传》

仿真技术是在系统建模基础上,使用仿真工具对系统进行仿真验证。仿真目的是验证系统建模的正确性与合理性。仿真方法有离散系统仿真、连续系统仿真以及离散-连续系统的混合仿真。本章将要介绍的仿真工具有离散系统仿真工具 Modelsim 和连续及离散-连续混成系统的仿真工具 Matlab/Simulink,并给出一些例子来说明如何进行仿真。

第3.1节 离散系统仿真

输入输出有限状态自动机的一种仿真方法是使用基于硬件描述语言 Verilog 的仿真工具 Modelsim。本节将简单介绍硬件描述语言 Verilog 的语法以及 Modelsim 仿真工具,并给出一些例子来说明如何使用 Verilog 语言依据自动机模型进行编写仿真程序和仿真。

3.1.1 硬件描述语言 Verilog

硬件描述语言(Hardware Description Language, HDL)是电子系统硬件行为描述、结构描述、数据流描述的语言。以文本形式来描述数字系统硬件的结构和行为的语言,用它可以表示逻辑电路图、逻辑表达式,还可以表示数字逻辑系统所完成的逻辑功能以及时序功能。硬件描述语言有 VHDL、Verilog、System Verilog 等。

Verilog 语言拥有几种重要的描述风格,包括:结构模型、用于逻辑综合的组合电路和时序电路的行为模型、FSM 数据通道模型以及周期精确的描述。本节只介绍 Verilog 语言,比较详细的材料可以参考文献[18]。

Verilog 行为描述语言作为一种结构化和过程性的语言,其语法结构非常适合于算法级和 RTL 级模型设计:

- 1、可描述顺序执行或并行执行的程序结构;
- 2、用延迟表达式或事件表达式来明确地控制过程的启动时间;
- 3、通过命名的事件来触发其它过程里的激活行为或停止行为;
- 4、提供了条件、循环程序结构;

- 5、提供了可带参数且非零延续时间的任务程序结构；
- 6、提供了可定义新操作符的函数结构；
- 7、提供了用于建立表达式的算术运算符、逻辑运算符和位运算符；
- 8、作为一种结构化语言也非常适合于门级和开关级的模型设计。

Verilog 语言以模块作为基本程序单位，关键字：**module**。一个模块的基本语法如下：

```
module module_name (port_list);
Declarations:
reg, wire, parameter,
input, output,
function, task, ...
Statements:
Initial statement
Always statement
Module instantiation
Gate instantiation
UDP instantiation
Continuous assignment
endmodule
```

其中，UDP(User Defined Primitives)是指用户可以自行定义的基本单元。

例如，使用 Verilog 语言写的代码：

```
module turnstile_FSM (C,P,clk,reset,y); //FSM for turnstile
{module 是 Verilog 关键字，表示一个模块}
{turnstile_FSM 是模块名}
{C, P, clk, reset, y 是端口名，其中 clk 是时钟关键字，reset 是重置关键字，C, P 和 y 是本模
块自身定义的端口}
//FSM for turnstile 是解释，程序不执行这段}
input C,P,clk,reset;
{输入端口： C,P,clk,reset }
output y;
{输出端口： y}
reg state;
{reg: 寄存器，state: 放在寄存器中的状态，1 位寄存器}
parameter S0=1'b0,S1=1'b1;
{参数： 自动机中两个状态。 若有 3-4 个状态则 reg 需要设置成 2 位寄存器。}
// Define the turnstile block
always @(posedge reset or posedge clk)
{always 语句，符号@，关键字 posedge 是上沿时触发，}
Begin
{状态定义开始}
```

```

if(reset)
    state<=S0; //initial state
else
    case(state)// case statement depending on state at register
        S0: if (C) state <=S1;  else state <=S0;
    // state S0:  if C is true then state at register waves to S1 else keep the state S0
        S1: if (P) state <=S0;  else state <=S1;
    // state S1: if P is true then state waves to S0 else continues to stay at S1
    endcase
end
// Define output
assign y = state;
{assign 是一个关键字，定义输出变量函数，把 state 值赋给输出变量 y}
endmodule

```

硬件描述语言 Verilog 可用来撰写数字电路，比如如图 3-1 所示的门级电路图，其 Verilog 代码如下所示。

```

module smpl_circuit(A,B,C, x,y);
    input A,B,C;
    output x,y;
    wire e;
    and g1(e,A,B);
    not g2(y,C);
    or  g3(x,e,y);
endmodule

```

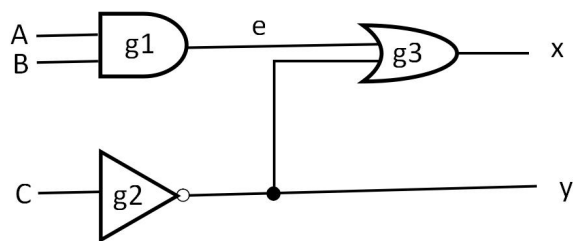


图 3-1 Verilog 门级电路图

Verilog 模型可以是实际电路不同级别的抽象。这些抽象的级别和它们对应的模型类型共有五种：

系统级(system): 用高级语言结构实现设计模块外部性能模型

算法级(algorithmic): 用高级语言结构实现设计算法的模型

RTL 级(Register Transfer Level): 描述数据在寄存器之间流动和如何处理、控制这些数据流动的模型

门级(gate-level): 描述逻辑门以及逻辑门之间连接的模型

开关级(switch-level): 描述器件中三极管和储存节点以及它们之间连接的模型。

RTL 设计定义为：用语言的方式去描述硬件电路行为的过程。有 3 种基本的描述方式：

数据流描述：采用 `assign` 连续赋值语句。

行为描述：使用 `always` 语句(可以多次执行，只要触发条件触发了)或 `initial` 语句块(只执行一次)的过程赋值语句。

结构化描述：实例化已有的功能模块或原语，即元件例化和 IP 核。

电路在物理上是并行工作：一旦接通电源，所有电路都同时工作。

在物理电路设计时，依据电路种类使用不同的语句进行设计。**组合电路：**采用 `assign` 语句或者 `always` 语句，**时序电路：**采用 `always` 语句。

Verilog 模块中：`assign` 语句、实例元件、`always` 模块描述的逻辑功能是并发的，可以同时执行。然而，在 `always` 模块内，逻辑是按照指定的顺序执行的，因此 `always` 模块内的语句也称为顺序执行。

3.1.2 仿真工具 ModelSim

Mentor 公司的 ModelSim 是业界常用 HDL 语言仿真软件，它能提供友好的仿真环境，是业界唯一的单内核支持 VHDL 和 Verilog 混合仿真的仿真器。它采用直接优化的编译技术、Tcl/Tk (Tool Command Language) 技术和单一内核仿真技术，编译仿真速度快，编译的代码与平台无关，便于保护 IP 核，个性化的图形界面和用户接口，为用户加快调错提供强有力的手段，是 FPGA/ASIC 设计的首选仿真软件。

本段简单介绍 Modelsim SE 10.1a 工具，详细请直接阅读 Modelsim 工具介绍。

Modelsim 工具从建立工作库开始的，直到仿真结束，需要经历六个步骤。

步骤一、建立工作库 (Library)

Project 一般是要在一个名为 `work` 的 Library 下面工作的，如果已经有 `work` 这个 Library 则可跳过这步直接去建立 Project。

步骤二、建立工程 (Project)

点击 `File—New—Project`，建立新的工程，会弹出对话框，输入 Project 名即可。

步骤三、编写代码

建立完 Project 后在弹出的对话框选择 `Create New File` 新建文件，选择文件类型为 Verilog，并在文件中编辑代码(文件名)以及测试程序代码(文件名`_tb`)。

步骤四、编译

编写完主程序代码和测试代码后，进行编译的时候可以看到文件的状态是蓝

色问号（?），右击一个文件，点击 **Compile—Compile All** 进行编译。

如果代码没有错误，会编译成功，问号会变成绿色对号（√），如果代码有错误就会变成红色的叉（×），此时需要根据错误信息修改代码，直到编译成功。

编译是否成功和错误信息可以看下面的 **Transcript** 栏，成功会提示 **successful**，错误会提示有 **error**。

步骤五、仿真

编译成功之后可以进行仿真。在工作库中选择相应的项目对测试文件进行仿真，选择变量以及通过 **add to wave** 按钮将变量的波形展示出来。

步骤六：退出仿真

点击 **Simulate** 选择 **End Simulation**。

3.1.3 仿真例子

例 3.1 自动旋转式栅门(Turnstile) 依据例 2.4 的建模，进行基于 ModelSim 仿真。

解： 首先要建立状态： **State S0: turnstile 锁住**， **State S1: turnstile 解锁**。

其次确定输入变量和输出变量：输入变量 **C** 和 **P**，**C** 表示刷卡成功，**P** 表示通过成功；输出变量 **y**，表示 **Turnstile** 锁住或解锁成功，其值为 **0** 和 **1**，**0** 表示锁住而 **1** 表示解锁成功。寄存器 **state** 是 **1** 位，因为只有两个状态 **S0** 和 **S1**。

现在使用 **Verilog** 语言撰写仿真模块 **module**，仿真模块反映状态间的逻辑转换关系，文件名为 **turnstile_FSM**。

```
Module turnstile_FSM (C,P,clk,reset,y);
//FSM for turnstile
    input C,P,clk,reset;
    output y;
    reg state;
    parameter S0=1'b0,S1=1'b1;
// Define the turnstile block
    always @(posedge reset or posedge clk)
    begin
        if(reset)
            state<=S0; //initial state
        else
            case(state)
                S0: if (C) state <=S1;   else state <=S0;
                S1: if (P) state <=S0;   else state <=S1;
            endcase
    end
```

```

    end
// Define output
    assign y = state;
endmodule

```

Modelsim 工具仿真还需要撰写测试文件，反映事件发生的先后时间关系，即时序关系，通过时延建立这时序关系。下面是自动旋转式栅门的测试文件，文件名 turnstile_FSM_tb。

```

`timescale 1ns/100ps
//时延单位是 1ns，时间精度为 100ps，1ns=1000ps。
module turnstile_FSM_tb;
    reg C,P,clk,reset;
    wire y;
    always #10 clk=~clk;
    initial
    begin
        clk=0;
        reset=0;
        C=0;
        P=0;
        #50 reset=1;//时延 50ns
        #20 reset=0;// 时延 20ns
        #10 C=1; // 时延 10ns，输入变量 C=1，刷卡开始时刻为 80ns。
        #20 C=0; // 时延 20ns，输入变量 C=0，即刷卡结束时刻为 100ns
        #20 P=0;// 时延 20ns，输入变量 P=0，
        #20 P=1;// 时延 20ns，输入变量 P=1，即通过开始，开始时刻在 140ns，
        #20 P=0;// 时延 20ns，输入变量 P=0，即通过结束，结束时刻在 160ns。
        #50 reset=1; //上述过程再开始一遍，开始时刻在 210ns。
        #20 reset=0;
        #10 C=1;
        #20 C=0;
        #20 P=0;
        #20 P=1;
        #20 P=0;
    end
    turnstile_FSM
    turnstile_FSM(.C(C),.P(P),.clk(clk),.reset(reset),.y(y));
    {波形图中要展示的波形变量，如 C, P, clk, reset, y}
endmodule

```

第一行是输入变量刷卡 C 的波形图，凸起图表示刷卡成功 2 次，第一次是在

80ns，第 2 次是在 240ns，

第二行是输入变量通过 P 的波形图，表示通过成功 2 次，第一次是在 140ns，第二次是在 300ns；

第三行是输入变量时钟 clk 的波形图，20ns 是一个时钟周期，即高电平保持 10ns，低电平保持 10ns；

第四行是输入变量 reset 波形图，先是低电平，在 50ns 时高电平开始，持续了 1 个时间周期即 20ns，变成低电平，过了 140ns，在 210ns 时 reset 开始高电平，又过了 1 个时钟周期，开始一直保持低电平；

第五行是输出变量 y 波形图，其实为状态转换波形，在 50ns 之前状态机没有工作，因而 y 值还没有获得，显示为中间红线。在 50ns 时状态为 S0，在 C 成功输入 10ns 后，即在 90ns 时输出值为 1 时，状态转换为 S1，保持 3 个时钟周期，在输入变量 P 成功输入后 10ns，即在 150ns 输出变量 y 为低电平，状态转换为 S0。这恰好体现了先刷卡后通过的时序关系。

在 210ns 时，上述过程又重复进行一遍。

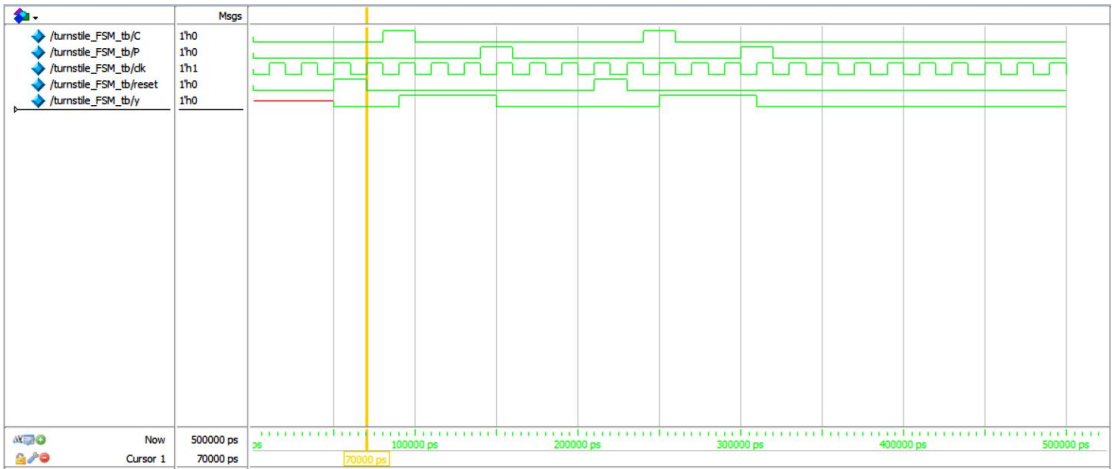


图 3-2 Turnstile 仿真图

例 3.2 餐巾纸售货机

一款餐巾纸售货机，只接受 5 角和 1 元硬币，1 包餐巾纸价格为 1.5 元，没有找零功能。

解：使用 J 表示 5 角，Y 表示 1 元（10 角）

首先定义状态集：S0: 表示 0 角，是起始状态，S5:表示 5 角，S10:表示 10 角，

S15:表示 15 角, S20:表示 20 角。在此基础上建立 Moore 型自动机模型和 Mealy 型自动机模型, 分别如图 3-3 和 3-4 所示。

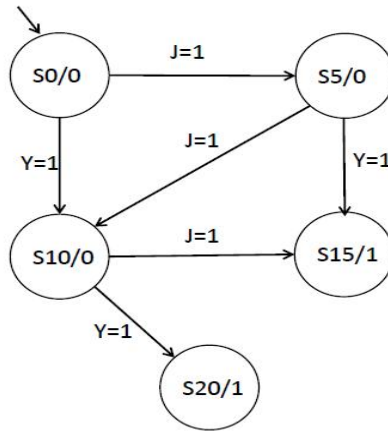


图 3-3 例 3.2 Moore 型自动机模型

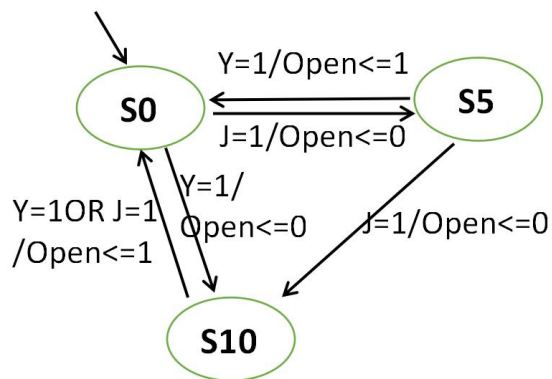


图 3-4 例 3.3 Mealy 型自动机模型

其次, 基于两个模型建立餐巾纸售货机仿真代码和测试代码。

(A) 餐巾纸售货机的 Moore 型自动机

对应的 Verilog 仿真文件代码

```

module vender_moore(J,Y,clk,reset,open);
    //Moore FSM for a vender
    input J,Y,clk,reset;
    output open;
    reg [2:0] state;
    parameter S0=3'b000,S5=3'b001,S10=3'b010,S15=3'b011,S20=3'b100;
  
```



```

//Define the vender block
always @(J or Y or state or reset)
    if(reset) state<=S0;
    else
        case(state)
            S0:if(J) state<=S5;
                else if(Y) state<=S10;
                    else state<=S0;
            S5:if(J) state<=S10;
                else if(Y) state<=S15;
                    else state<=S5;
            S10:if(J) state<=S15;
                else if(Y) state<=S20;
                    else state<=S10;
            S15:state<=S15;
            S20:state<=S20;
        endcase
//Define output during S3
assign open=(state==S15 || state==S20);
endmodule

```

测试代码:

```

`timescale 1ns/100ps
module vender_moore_tb;
reg J,Y,clk,reset;

wire open;
always #10 clk=~clk;
initial
    begin
        clk=0;
        reset=0;
        J=0;
        Y=0;
        #20 reset=1;
        #20 reset=0;
        #20 J=1;
        #20 J=0;
        #20 Y=1;
        #20 Y=0;
        #20 reset=1;
        #20 reset=0;
        #20 Y=1;
        #20 Y=0;
        #20 Y=1;
        #20 Y=0;
    end
endmodule

```

```

        #20 reset=1;
        #20 reset=0;
        #20 J=1;
        #20 J=0;
        #20 J=1;
        #20 J=0;
        #20 J=1;
        #20 J=0;
        #20 J=1;
        #20 J=0;
    end
    vender_moore
    vender_moore(J(J),Y(Y),clk(clk),reset(reset),open(open));
endmodule

```

仿真结果如图 3-5 所示:

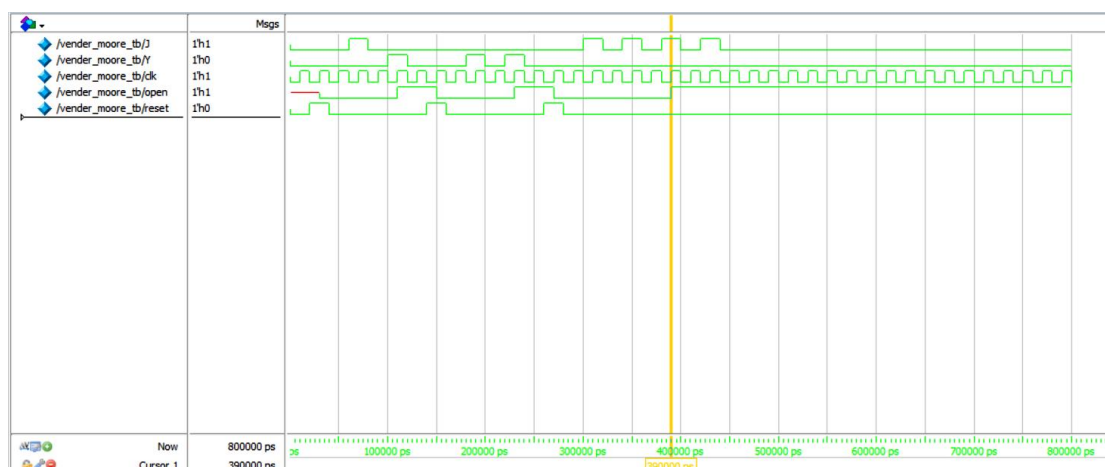


图 3-5 例 3.2 Moore 型自动机仿真图

(B) 餐巾纸售货机 Mealy 型自动机

对应的 Verilog 仿真文件代码:

```

module vender_mealy(J,Y,clk,reset,open);
    //Mealy FSM for a vender
    input J,Y,clk,reset;
    output open;
    reg [1:0] state,nstate;
    reg open;
    parameter S0=2'b00,S5=2'b01,S10=2'b10;
    //Next state and output combinational logic
    always @ (J or Y or state or reset)
        if(reset)
            begin nstate<=S0;open<=0;end
        else case(state)

```

```

S0:begin open <=0;
      if(J) nstate<=S5;
      else if(Y) nstate<=S10;
      else nstate<=S0;
    end
S5:begin
      if(J) begin open<=0; nstate<=S10; end
      else if(Y) begin open<=1; nstate<=S0; end
      else nstate<=S5;
    end
S10:begin
      if(J) begin open<=1; nstate<=S0;end
      else if (Y) begin open<=1; nstate<=S0;end
      else nstate<=S10;
    end
  endcase
always @(posedge clk)
  state<=nstate;
endmodule

```

测试代码:

```

`timescale 1ns/100ps
module vender_mealy_tb;
reg J,Y,clk,reset;

wire open;
always #10 clk=~clk;
initial
begin
  clk = 0;
  reset=0;
  J=0;
  Y=0;
  #4 reset=1;
  #10 reset=0; //时钟同步导致取不到 reset 信号
  #4 J=1;
  #20 J=0;
  #20 Y=1;
  #20 Y=0;
  #20 J=1;
  #20 J=0;
  #5 Y=1;
  #20 Y=0;
  #5 J=1;
  #20 J=0;

```

```
end
vender_mealy
vender_mealy(.J(J),.Y(Y),.clk(clk),.reset(reset),.open(open));
endmodule
```

仿真结果如图 3-6

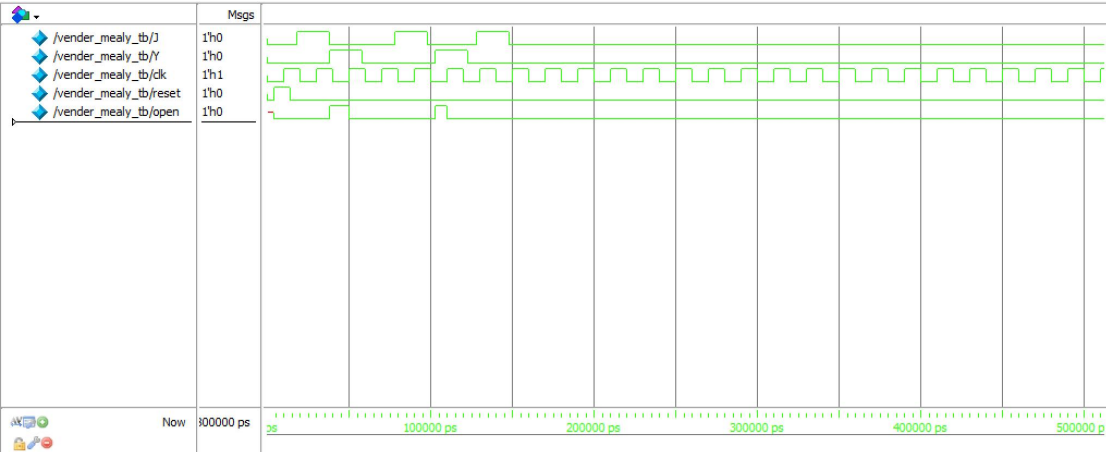


图 3-6 例 3.2 Mealy 型自动机仿真图

例 3.3 汽车自主防撞系统

汽车自主防撞系统是在汽车行驶过程中自动感知前面是否有障碍物或行人，若存在则汽车自动采取停车或避开措施，避免发生碰撞事故。

自主防撞系统与汽车行驶状态融合，构成了汽车自主防撞系统的状态：

- 1、自动检测状态（Det）：自动检测汽车前方是否有障碍物或行人，为初始状态；
- 2、预警状态（Al）：当汽车发现前面有障碍物或行人时，汽车进入预警状态。当障碍物或行人消失后，预警状态转移到自动检测状态；
- 3、减速状态（Dec）：在预警状态下，检测到障碍物或行人在运动中，则汽车进入减速状态。在减速状态下，若障碍物或行人消失则进入自动检测状态；
- 4、制动状态（Br）：在预警状态下，若障碍物或行人处于停止情况，则汽车进入制动状态，准备停车。在制动状态下，若障碍物或行人消失，则汽车进入自动检测状态。

解：建立 Mealy 型自动机模型

状态集 S: {Det, Al, Dec, Br}。

数据输入变量集 Input: {Ob, Obsp}，数据输入变量取值为 0 和 1，Ob=1 表示

系统检测到了障碍物或行人， $Ob=0$ 表示没有检测到障碍物和行人， $Ob_{sp}=1$ 表示障碍物或行人在运动中， $Ob_{sp}=0$ 表示在障碍物或行人在静止中。

控制输出变量集 **Output**: {Detection, Alarm, Deceleration, Brake}, 控制输出变量值集: {0, 1}, 若控制输出变量取值 1 表示控制变量采取相应状态动作: **Brake** 状态下汽车采取制动, **Deceleration** 状态下汽车采取减速, **Alarm** 状态下采取提醒, **Detection** 状态下采取自动检测; 若控制输出变量取值 0 表示控制变量不采取相应状态动作。

转移函数和输出函数如图 3-7 所示:。

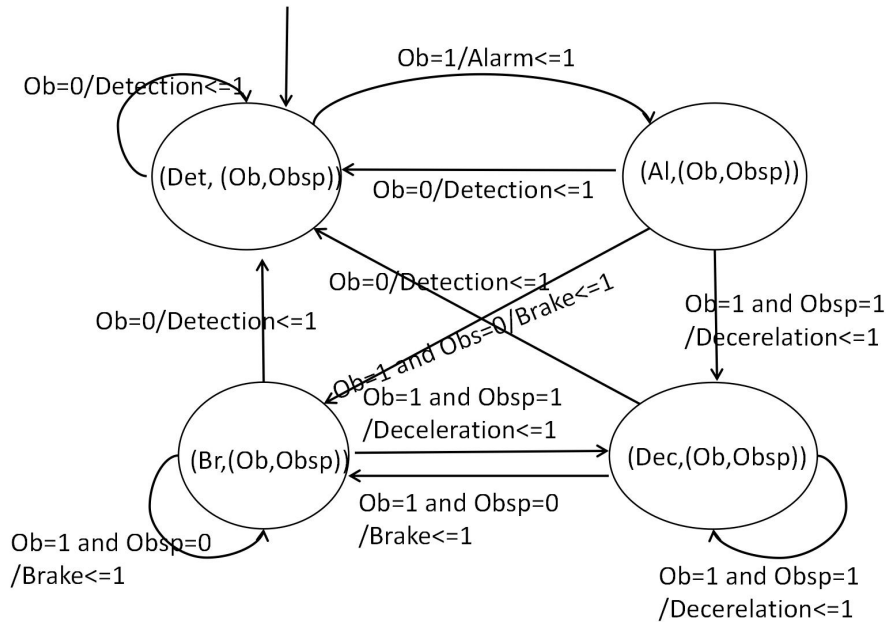


图 3-7 例 3.3 Mealy 型自动机建模

依据 Mealy 型自动机撰写汽车自主防撞系统的 Verilog 仿真代码:

```

module avoidCrush_Mealy(clock,Ob,Ob_sp,reset,
                        Detection,Alarm,Deceleration,Brake);
input clock,Ob,Ob_sp,reset;
output reg Detection, Alarm, Deceleration, Brake;
reg[1:0] ControlState;
parameter //ControlState
Det=2'b00, Al=2'b01, Br=2'b10, Dec=2'b11;
always @(posedge clock)
    if(reset)
        begin
            ControlState<=Det;
        end
end
  
```

```

        Detection<=1;
    end
else
    case(ControlState)
    Det: if(Ob) begin ControlState<=Al; Alarm<=1; Detection<=0; end
        else begin ControlState<=Det; Detection<=1; end
    Al: if(Ob) begin
        if (Obsp) begin ControlState<=Dec; Deceleration<=1; Alarm<=0; end
        else begin ControlState<=Br; Brake<=1; Alarm<=0; end
    end
        else begin ControlState<=Det; Detection<=1; Alarm<=0; end
    Dec: if(Ob) begin if (Obsp) begin ControlState<=Dec; Deceleration<=1; end
        else begin ControlState<=Br; Brake<=1; Deceleration<=0; end
    end
        else begin ControlState<=Det; Detection<=1; Deceleration<=0; end
    Br: if(Ob) begin if (Obsp) begin ControlState<=Dec; Deceleration<=1; Brake<=0; end
        else begin ControlState<=Br; Brake<=1; end
    end
        else begin ControlState<=Det; Detection<=1; Brake<=0; end
    endcase
endmodule

```

对应的测试代码:

```

`timescale 1ns/100ps
module avoidCrush_Mealy_tb;
reg clock,Ob,Obsp,reset;;
wire Detection, Alarm, Deceleration, Brake;
wire[1:0] ControlState;
always #10 clock=~clock;
always @(posedge clock)
begin
end
initial
begin
    reset=0;
    clock=0;
    Ob=0;
    Obsp=0;
    #10 reset=1;
    #20 reset=0;
    #20 Ob=1;
    #20 Obsp=1;
    #50 Obsp=0;
    #20 Ob=0;
    #40 Ob=1;
end

```

```
#20 Obsp1;
#40 Ob=0;
#40 Obsp=0;
#500 reset=1;
end
avoidCrush_Mealy avoidCrush_Mealy(.clock(clock),.Ob(Ob),.Obsp(Obsp),.reset(reset),
.Detection(Detection),.Alarm(Alarm),.Deceleration(Deceleration),.Brake(Brake));
endmodule
```

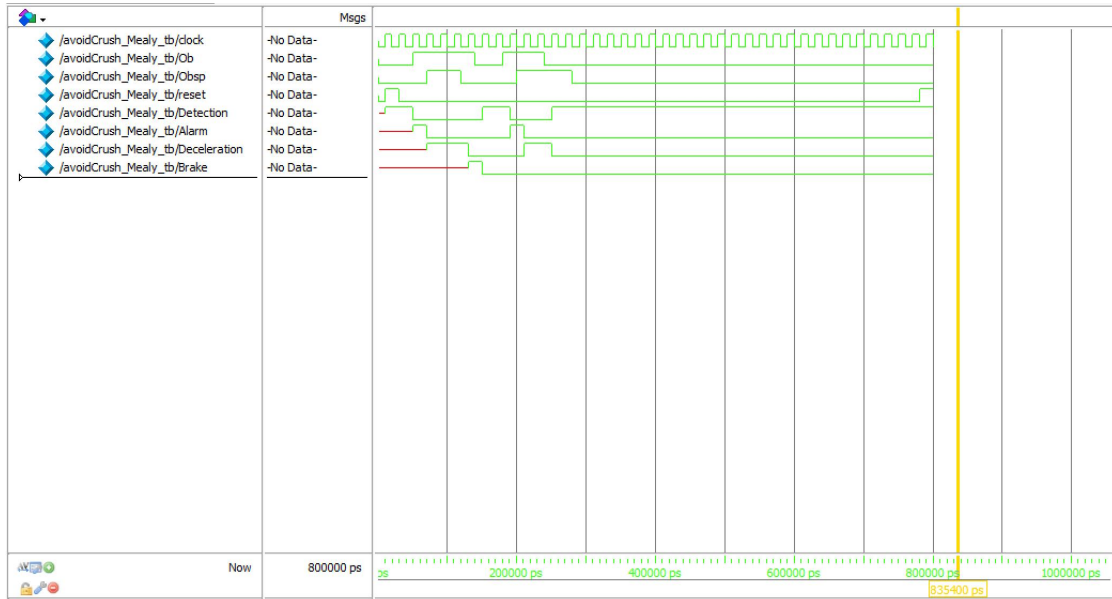


图 3-8 例 3.3 Mealy 型自动机仿真结果

第 3.2 节 离散-连续系统仿真

第 2.4 节介绍了混成系统，一个连续（实值）状态和离散（有限值）状态的混合系统，它反映了状态随时间的演化，并给出了空调系统以及双水缸控制系统的例子。

本节介绍混成系统的仿真方法，主要使用建模仿真工具 **Matlab/Simulink** 进行仿真。

Simulink 集成于 **Matlab**，可对联系以及离散-连续融合系统进行建模仿真。**Simulink** 使用块（block）组成图形界面，可调用 **Matlab** 工作区接口，使用工作区参数，进行系统数据输入和输出。

一个 **Simulink** 模型包含三个模块：信号输入源模块、系统模拟模块和输出显示模块。信号输入源模块负责系统数据输入，主要包含三种输入源类型：常数、函数信号发生器（如正弦函数波、跳跃函数波）以及用户自定义函数；系统模拟

模块是核心模块，主要负责数据的处理，用户需组合定义模拟模块；输出显示模块负责显示系统模拟模块的仿真结果，可通过图形、示波器或文件的方式进行显示。



3.2.1 工具介绍

本节是在 Matlab2019a 平台上进行仿真的，因此，一些命令和界面是依赖于这个平台的。

在 Matlab 命令窗口 (Command Window) 输入命令“simulink”，启动 Simulink 工具。

Simulink 工具是基于模块化的，因此，Simulink 提供了许多模块供建模中使用，Simulink 的说明书详细介绍了这些符号的具体涵义和使用方法。

(1) 模块：在弹出的“Simulink Library Browser”窗口中，可根据需要，找到所需块。启动 Simulink 有两种模式：在 Command Window 框架下，fx>>输入 simulink，或者在 Matlab 界面栏中直接点击 Simulink。

Simulink 是基于模块化建模的，可直接将模块从 Simulink 库中拖放到模型中，模型最左端是模型起点，可从 Sources 中选择相应的模块，如 Sine wave 模块 ，作为源块，最右端是输出模块，从 Sinks 库中选 Scope 模块 。

(2) 连接：模块之间需要建立连接，实现数据的传输。连接块时，需要选择源块的输出端口，然后连接到目标块的输入端口，使用连接线——建立源块和目标块的连接。

(3) 仿真：点击输出端 Scope，系统会启动 Scope 界面，在 Scope 界面上栏中点击播放按钮，界面展示仿真结果，如图 3-9 展示了正弦 Sine 函数图像，其函数值在[-1, 1]内，振幅频率=1。可以通过调整 Sine 函数的参数，调整函数图像，实现更多场景的仿真，如车速。

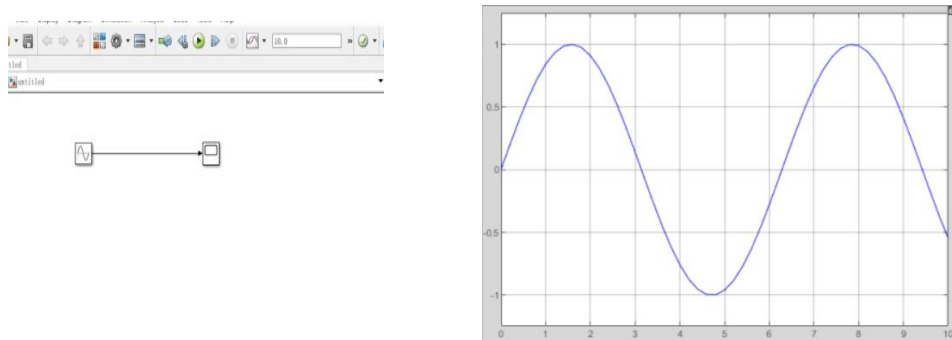


图 3-9 Sine 函数仿真结果（振幅频率=1）

3.2.2 参数设置

模块需要设置其参数，完成计算值的确定，实现仿真结果的改变。通过双击区块图标，会出现 **Block Parameter** 界面，依据这个界面即可查看或修改块参数。

对于正弦 **Sine** 函数，先调整频率 **Frequency** 从 1 调整成 2，函数图像的振幅频率增大 1 倍，再打开输出端，得到如下的仿真图 3-10(a)：

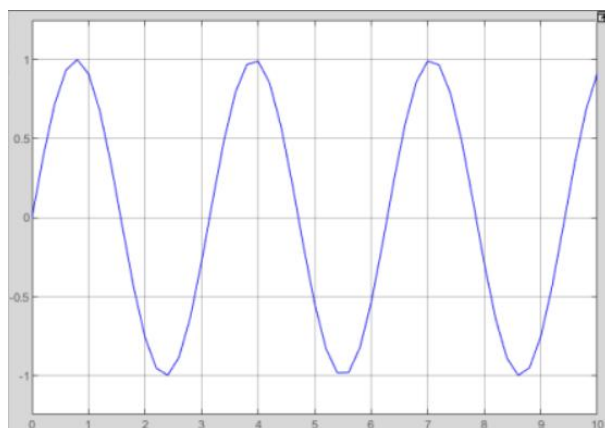


图 3-10(a) Sine 函数仿真结果(振幅频率=2)

若把 **Frequency** 从 1 缩小到 0.5 则会得到频率变小的图形，函数图像变比较缓，如图 3-10(b)。

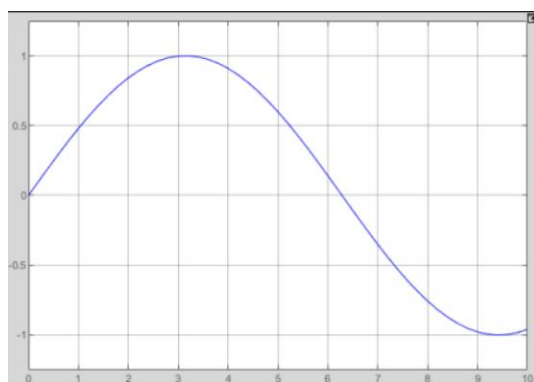


图 3-10(b) Sine 函数仿真结果(振幅频率=0.5)

调整参数 **Amplitude** 来调整 **Sine** 函数的振幅。如若 **Amplitude** 参数调整为 5 则得到函数值区间为 $[-5, 5]$ 。若调整 **Bias** 参数，如将 **Bias** 参数从 1 调整为 10，则函数图像值在 $[9, 11]$ 中。但若把 **Bias** 参数从 1 调整为 100 则函数图像值在 $[99, 101]$ 中，这不符合车速的实际情况。

输入模块实现函数值为非负数，用 **Sine** 函数仿真车速，需要增加模型中模块，增加输入模块，与 **Sine** 函数相加，实现函数值非负。从 **Sources** 中选中 **Constant**

模块（这个模块值是常数但可以调整），从 **Math operation** 中选中加模块 **add**，把 **Sine** 模块和常数模块连接到加法模块的输入端口，把加法模块输出端口和输出模块的输入端口相连接。从仿真结果来看，函数图像上移了一个单位，即函数值区间为 $[0,2]$ 。若把上限值上升到 120，则需要增加一个模块：从 **Commonly Used Blocks** 库中选择 **Gain** 模块添加到模型中，放在加法模块和输出模块之间，把 **Gain** 模块中的参数 1 调整为 50，仿真结果显示函数值调整为 $[0, 100]$ ，若调整为 60 则函数值在 $[0, 120]$ 中，如图 3-10(c)所示。

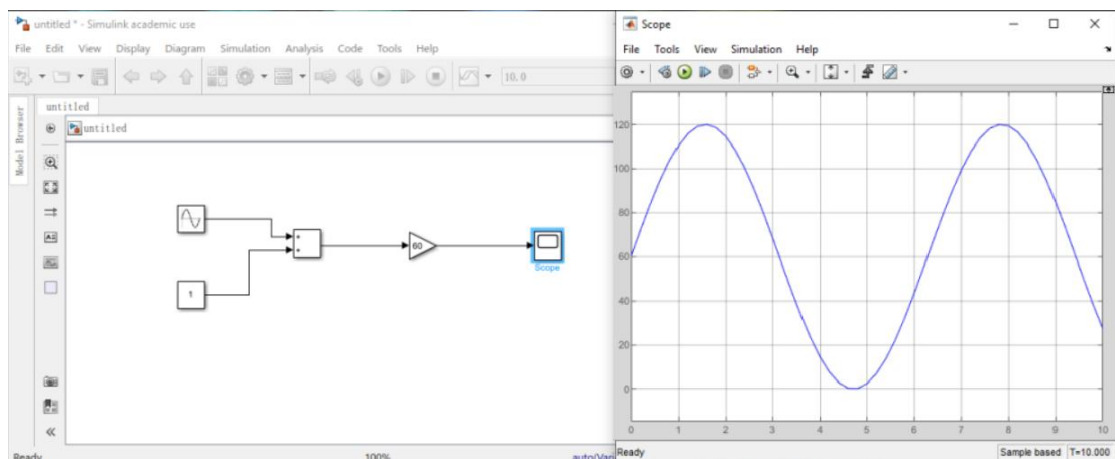


图 3-10(c) Sine 函数仿真车速 $[0, 120]$ 图

从图中可以看到，Sine 函数图像从 0 时刻开始时函数值为 60，我们可以通过调整 Sine 函数的参数 **Phase** 为 -1.5，得到图形函数值是从 0 开始的。

现在固定模型中的模块参数：Sine 模块中 **Amplitude**=1，**Bias**=0，**Frequency**=0.5，**Phase**=-1.5，**Sample time**=0，仿真时间 **T**=20，则得到下面的仿真图形，如图 3-10(d)所示。

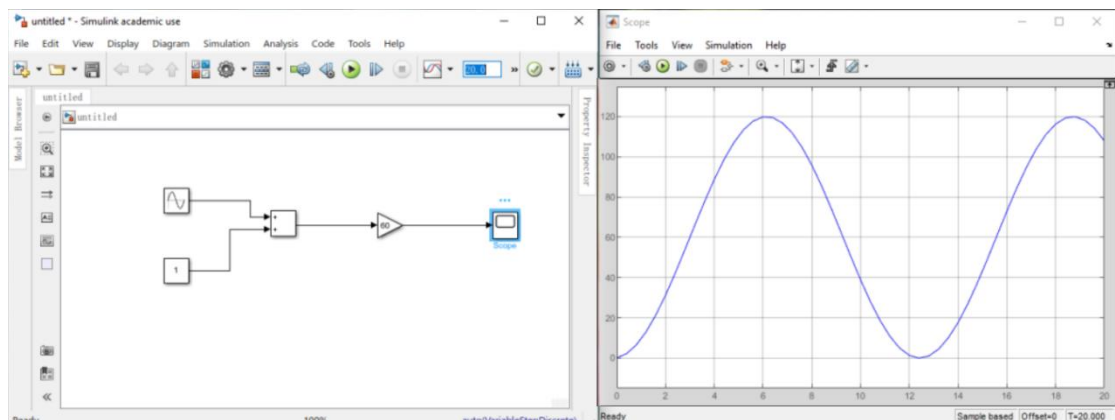


图 3-10(d) Sine 函数仿真车速 $[0, 120]$ 图(Phase=-1.5)

例 3.4 超速自动提醒系统

例 2.13 介绍了超速自动提醒系统，设置车速为 100 公里/小时。当车速超过（等于）100 公里/小时时，汽车会自动语音提醒“超速”，直到车速低于 100 公里/小时为止。现在使用 Simulink 来仿真超速自动提醒系统。

解：将 3.2.2 中 Sine 函数作为车速输入，从 Logic and Bit Operations 库中选 Compare to constant 项作为超速判断节点，再连接上输出模块 Scope，给 Scope 设置两个输入，一个为判断输入其值为 0 和 1，另一个为车速输入其值为[0,120]。Simulink 模型为图 11(a)，仿真结果见图 11(b)，图中红线为车速，蓝线为判断 0-1 线，由于采用统一的纵坐标其值为 0 到 120，因而蓝线 0 与 1 比较贴近，但还是能看出当车速超过（等于）每小时 100 公里时蓝线为 1，当车速低于每小时 100 公里时蓝线为 0。

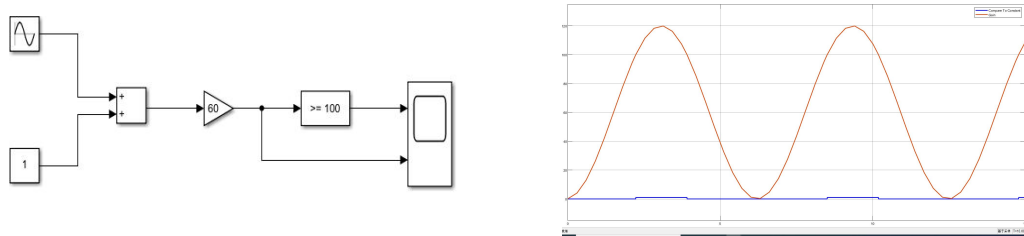


图 11(a) 例 3.4 超速提醒系统 Simulink 模型 图 11(b) 例 3.4 超速提醒系统 Simulink 仿真图

3.2.3 子系统

Simulink 在仿真时，通常是一个较大且复杂的系统，因此在建立 Simulink 模型时可以建立子系统去构建复杂系统。从 Commonly Used Blocks 中选中 subsystems 模块放在模型中，单击子系统模块输出端增加输出端口，单击子系统模块输入端增加输入端口。这个子系统也是一个模块，只不过这个模块是需要重新构建。如图 3-12(a)中，包含了一个子系统 subsystem，它有一个输入端口，两个输出端口。其中子系统内部结构如图 3-12(b)所示。

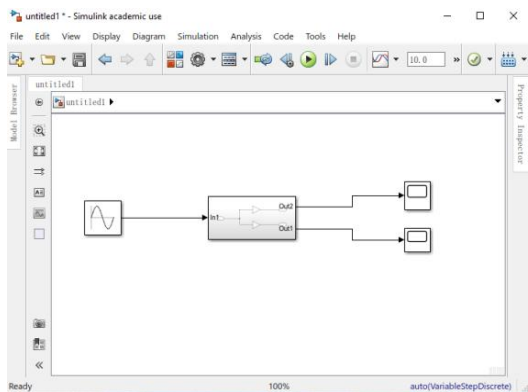


图 3-12(a) 包含子系统

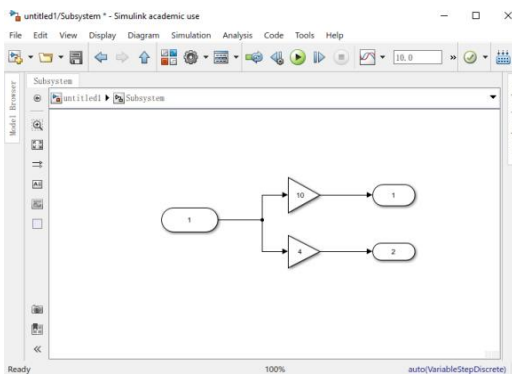


图 3-12(b) 子系统内部结构

最后，点击运行按钮，双击 Scope 块时，可获得对应的仿真结果，具体如图 3-13 (a/b) 所示，其差异体现在函数图像值得范围，(a)图像取值范围为 $[-10,10]$ ，而(b)为 $[-4,4]$ ，这是由子系统中模块 Gain 中参数值分别为 10 和 4 决定。

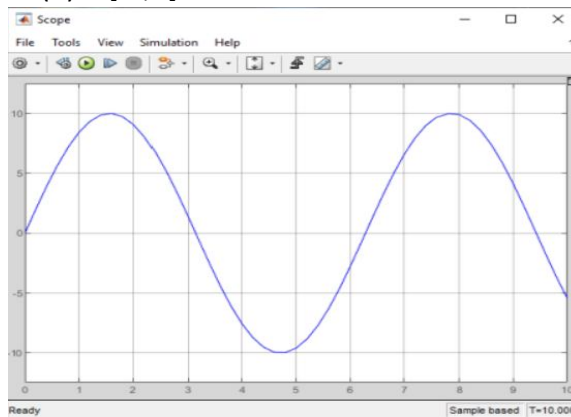


图 3-13(a) 模块 Gain 中参数值=10

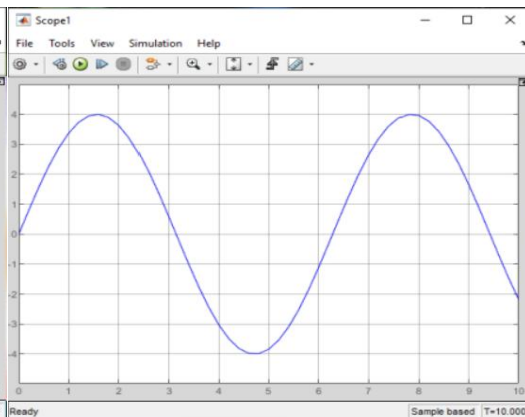


图 3-13(b) 模块 Gain 中参数值=4

3.2.4 自定义模块

从 Simulink 的 User-Defined Function 库中选取 Fcn 模块，拖到 Simulink 模型设计界面，双击在 Block Parameter: Fcn 中输入表达式：如 $21 \cdot \exp(-0.02 \cdot u)$ ， u 是输入变量。点击 Ok 就定义好了一个自定义的函数模块。

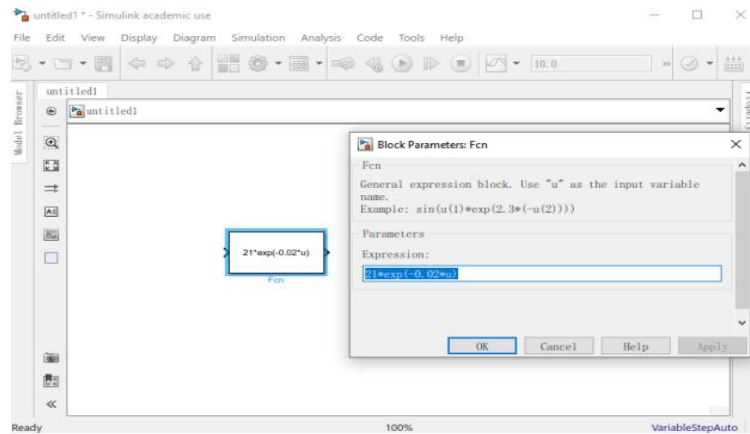


图 3-14(a) 房间温度自然下降自定义模块

现在来对自定义函数模块进行仿真。

由于这个自定义模块是以时间为自变量的，因此需要建立它的输入模块来代表时间的变化。定义 Fcn 模块中的函数 $f(u)=21*\exp(-0.02*u)$ 。从 Sources 库中选择 Ramp 模块，定义其参数为 Slope=1, Start time=0, initial output=0，这样模块 Ramp 的输出为时间。从 Sinks 库中选取 Scope 模块，定义 1 个输入口，连接模块 Fcn 为函数 $f(u)=21*\exp(-0.02*u)$ 的输出。设置仿真时间 T=5，则 Fcn 函数值 =19，即房间温度自然从摄氏 21 度下降到摄氏 19 度，共需 5 个单位时间。

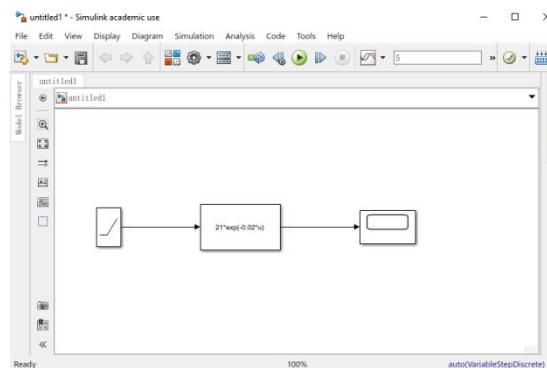


图 3-14(b) 房间温度自然下降自定义模块

仿真结果图，在时间 $T=5$ 时函数值为 19。尽管定义是指数方幂函数，但仿真图形看起来是直线，这与仿真时间 T 取值有关，图 3-14 (c)。若 $T=100$ 则可以看出图形不再是直线了，而且函数值等于 3，图 3-14 (d)。

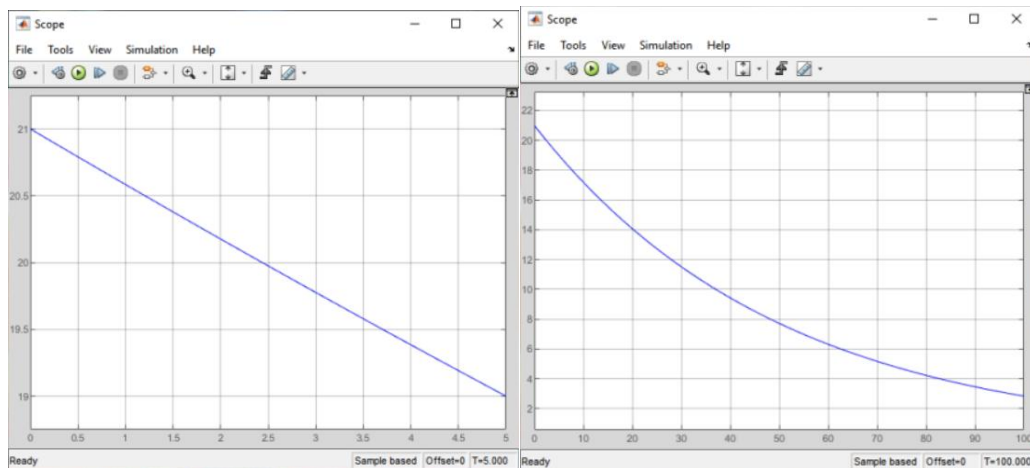


图 3-14(c) 房间温度自然下降仿真(T=5) 图 3-14(d) 房间温度自然下降仿真(T=100)

再设空调工作时房间温度函数 $T = -15 \cdot \exp(-0.08t) + 30$ ，房间的初始温度为摄氏 15 度，其 Simulink 模型以及仿真图形。设时间 $T = 6.4$ ，则经过 6.4 个单位时间工作后房间温度从摄氏 15 度上升到摄氏 21 度。

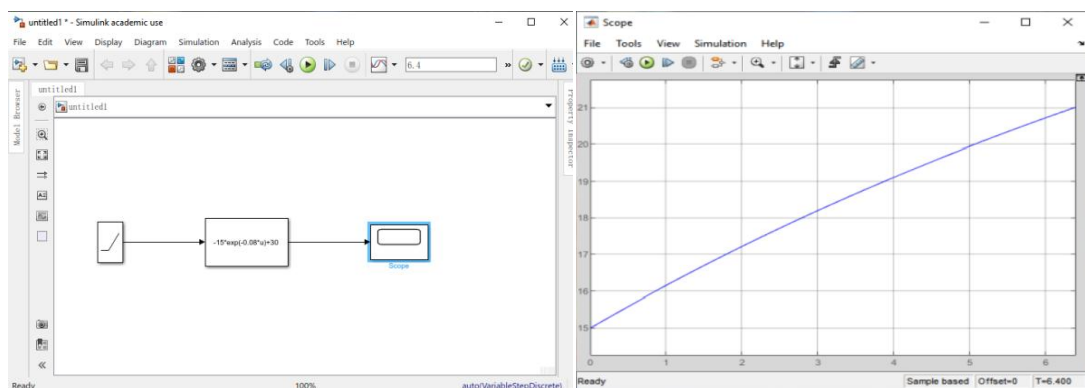


图 3-15(a) 空调工作房间温度自定义模块 图 3-15(b) 空调工作房间温度仿真图

3.2.5 状态图 Stateflow

在混成自动机中，有离散控制也有连续控制，使用 Simulink 中 StateFlow 库建立混成自动机的仿真。

Stateflow 是一个基于状态机和流程图对组合和顺序决策逻辑进行建模和仿真的环境。Stateflow 允许用户组合图形和表格表示，包括状态转换图、流程图、状态转换表和真值表，可以对系统如何对事件、基于时间的条件和外部输入信号做出反应进行建模。

每个状态都有 entry 和 during 两个模式，每个模式下都有变量以及变量函数，刻画了在该模式下变量变化规律。其中 entry 模式相当于初始模式其变量只执行

一次，如 $t=0$ ， $x=21*\exp(-0.02*t)$ ，得到 $x=21$ 。**during** 模式是该状态下变量变化的主体，可以一直在执行直到变量值符合状态转换条件为止，如 $t=t+1$ ， $x=21*\exp(-0.02*t)$ ， t 是一个计数器，步长为 1，当 t 的单位很小时（如纳秒），就可以看到 x 的变化是连续型的。**Stateflow** 中状态间的转换依赖于状态中数据满足状态转换条件，如 $x \leq 19$ 。

例 3.5 制热空调系统 Stateflow 模型

图 3-16 是空调系统中加热控制的状态图例子，它描述了加热空调系统在房间温度变化下的空调 **Stateflow** 模型。

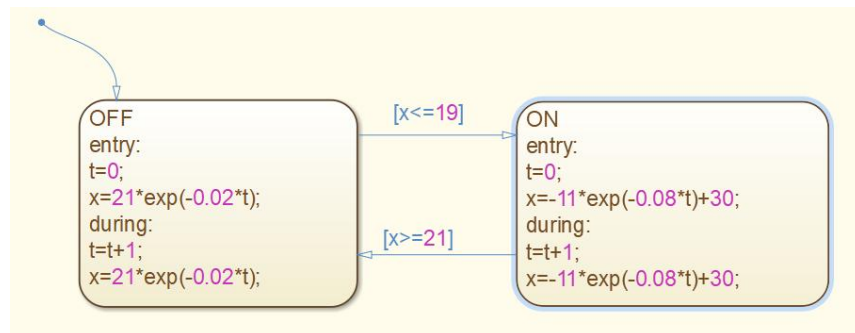


图 3-16 空调系统的 Stateflow 图

该状态图由两个状态 **OFF** 和 **ON** 组成，其中 **OFF** 是初始状态，从 **OFF** 状态到 **ON** 状态的转换条件是 $x \leq 19$ ，从 **ON** 状态到 **OFF** 状态转换条件是 $x \geq 21$ 。在两个状态里有两个关键字：**entry** 和 **during**，它们都是定义动作命令，但其语义不同。**entry** 下的动作是一次性执行，而且是状态一启动就执行，如 $t=0$ 表示在这个状态下 t 重置为 0， $x=21*\exp(-0.02*t)$ 是定义了变量 x 关于自变量 t 的函数。**during** 下的动作一直在执行，直到转移到其他状态，如 $t=t+1$ 定义了 t 是一个计数器，从初始值 $t=0$ 开始计数，函数 x 也是从初始值 $t=0$ 开始计算，直到状态转移到其他状态为止。在 **OFF** 状态下 x 的初始值是 21，而在状态 **ON** 下 x 的初始值为 19。

例 3.6 制热空调系统仿真

在例 3.5 的制热空调系统 **Stateflow** 模型基础上，建立 **Simulink** 模型并进行仿真。

解：**Stateflow** 状态图在 **Simulink** 模型中作为块运行。**Stateflow** 块使用输入和输出信号与模型中其他块进行交互。**Stateflow** 块空调-chart 为 **Simulink** 的输入模

块, 增加 Simulink 中 Sinks 库中输出模块 Scope, 将 Scope 模块两个数据入口, 分别连接状态图的两个输出口 x 和 t 。通过这些联系, Stateflow 和 Simulink 软件共享数据并响应在模型和图表之间传播的数据中。例如, Stateflow Air-Conditioning 块与 Simulink 输入输出模块集成, 构成空调系统 Simulink 设计模型, 如图 3-17(a)所示。

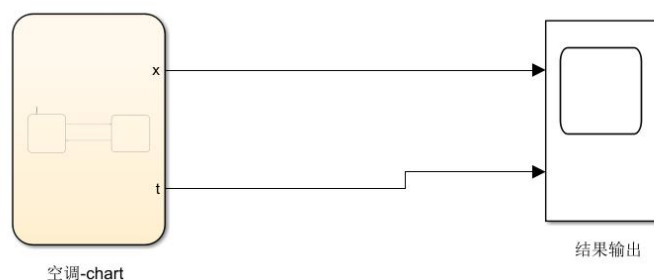


图 3-17(a) 空调系统 Simulink 模型

对这个 Simulink 进行仿真, 设置仿真时间 $T=20$, 仿真图形如图 3-17(b):

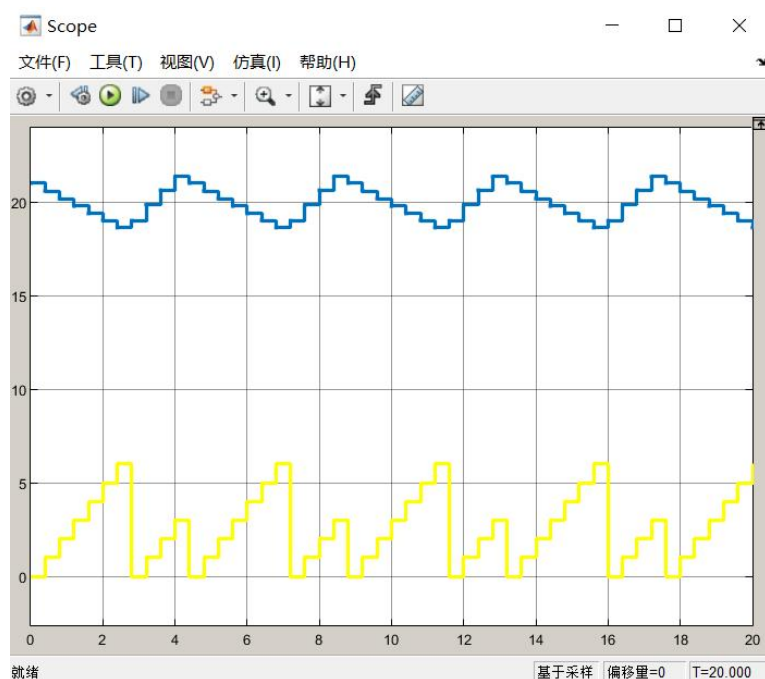


图 3-17(b) 空调系统 Simulink 模型仿真图

蓝色是房间温度变化曲线, 黄色是时间变化曲线。温度变化是从摄氏 21 度开始的, 此时为 OFF 状态, 经过 6 个多单位时间温度降到摄氏 19 度, 状态机触发状态转换条件 $x \leq 19$, 状态从 OFF 状态转移到 ON 状态, 空调开始加热房间温度从摄氏 19 度逐步上升摄氏 21 度, 大约用了 4 个单位时间, 同时触发状态转换条件 $x \geq 21$, 状态机从 ON 状态转移到 OFF 状态, 一直这样循环下去。

例 3.7 水缸系统的混成自动机仿真。

例 2.15 由两个水缸（缸 1 和缸 2）组成的水缸自动控制系统，水是按照一个常速通过一个软管流进水缸，在任何时刻水只能流进其中一个水缸，假定软管在两个水缸瞬间转换，并保持两个水缸中的水在一定容量以上。两个水缸中的水都是按照常速 $1/2$ 流出，即 $v_1=v_2=1/2$ ，水流进水缸恒均速度 $w=3/4$ 。起始状态是缸 1 水容量为 0，缸 2 水容量为 1。

情形 1. 两个水缸最低容量 $r_1=r_2=0$ 。

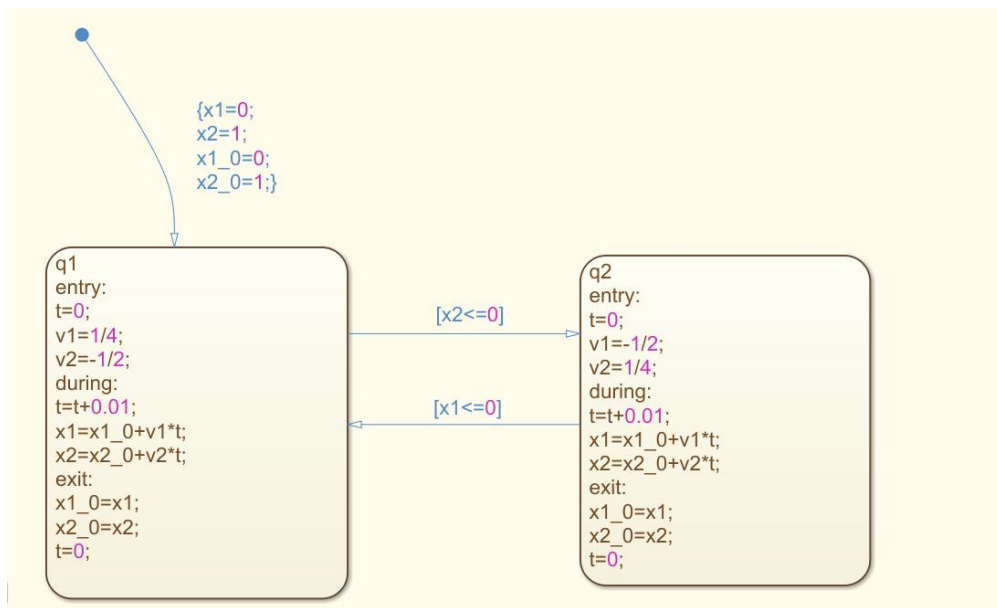


图 3-18(a) 水缸系统 stateflow 模型图

Simulink 模型：在 Stateflow 模型的基础上，增加输出模块，得到了水缸 Simulink 模型。

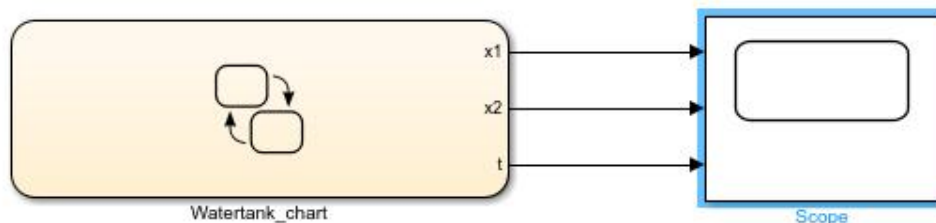


图 3-18(b) 水缸系统 Simulink 模型图

对这个 Simulink 模型进行仿真，得到仿真图，见图 3-18(c)。其中蓝色线是时间 t 的变化曲线，黄色线是第 1 水缸容量 x_1 的变化曲线，红色线是第 2 水缸容量

x_2 变化曲线。仿真总体时间是 45 个单位时间，从仿真图中可以看到，在放置时间 40 以后，两个水缸容量很难保持，进入软管在两个水缸间不停地转换，不能再往任何一个水缸注水了。

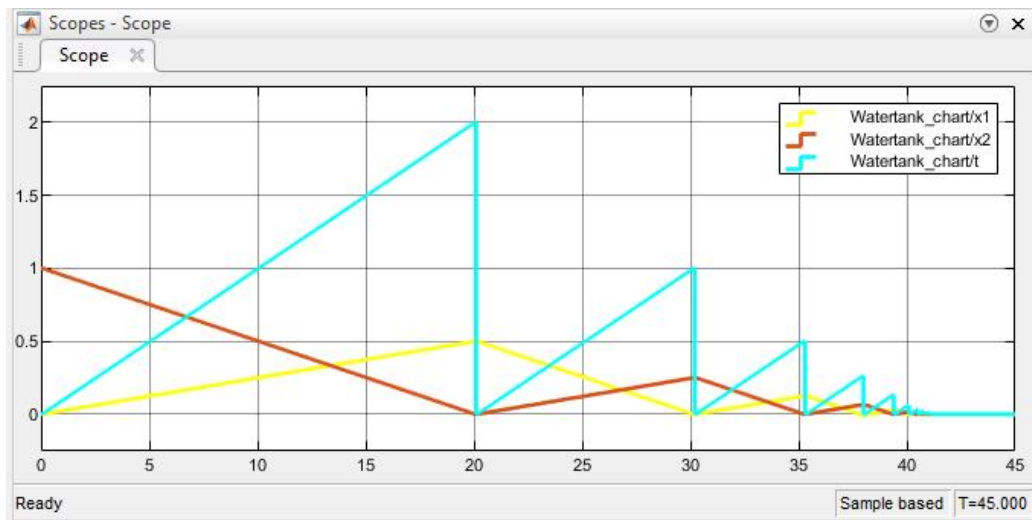


图 3-18(c) 水缸系统仿真图(最低容量 $r_1=r_2=0$)

情形 2. 两个水缸最低容量 $r_1=0$, $r_2=0.1$ 。

调整水缸的最低容量，如第 1 水缸最低容量 $r_1=0$ ，第 2 水缸最顶容量 $r_2=0.1$ ，其余参数不变。则两个水缸变换条件为：第 1 水缸变换到第 2 水缸条件为 $r_2 \leq 0.1$ ，第 2 水缸变换到第 1 水缸条件为 $r_1 \leq 0$ ，Stateflow 图为图 3-18(d)，仿真时间 45，仿真结果图为图 3-18(e)。

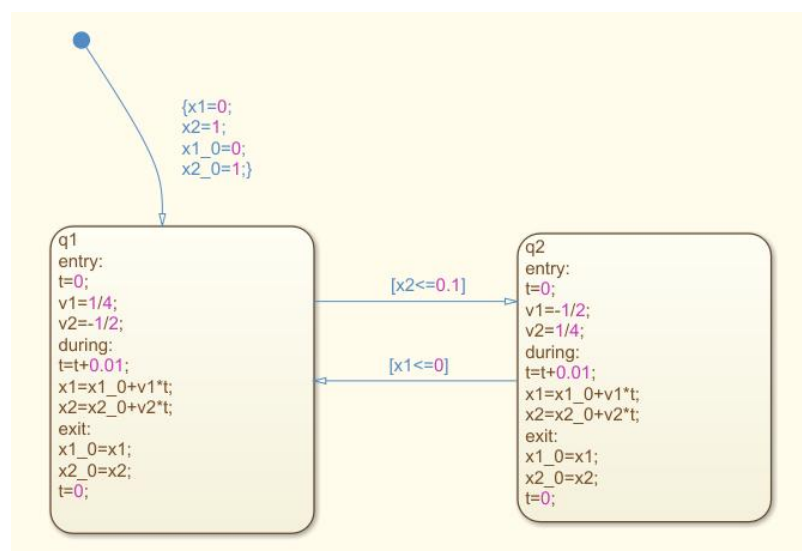
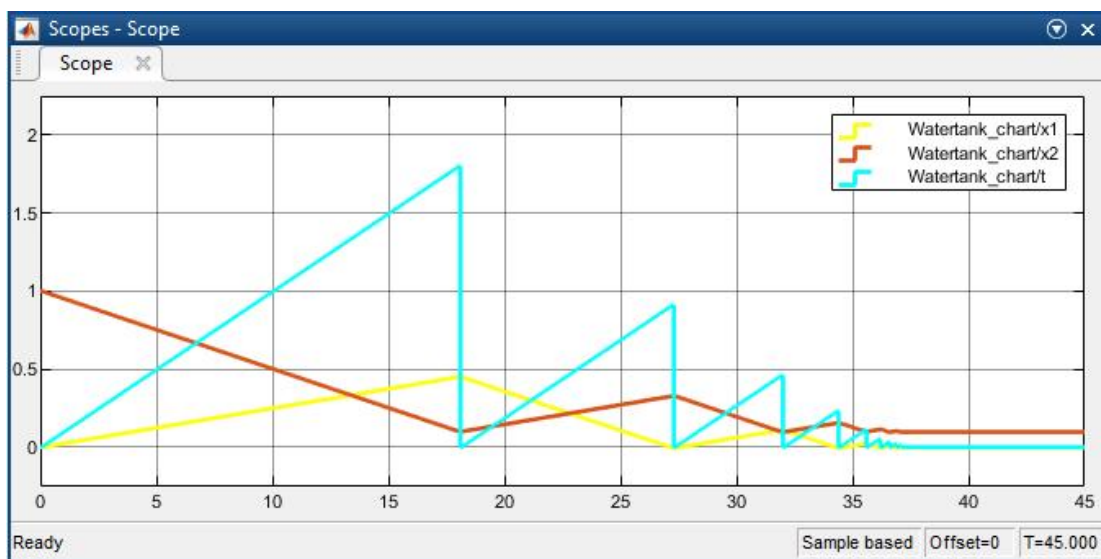


图 3-18(d) 情形 2 水缸系统 stateflow 模型图(最低容量 $r_1=0$, $r_2=0.1$)

图 3-18(e) 情形 2 仿真图(最低容量 $r_1=0$, $r_2=0.1$)

第 3.3 节 本章小结

本章介绍了智能嵌入式系统的仿真语言和工具。使用硬件描述语言 Verilog 以及仿真工具 Modelsim 对离散控制系统进行仿真，使用 Simulink 建模语言和仿真工具对混成系统进行建模和仿真。这种仿真在第 2 章系统建模的基础上，设置系统的状态和输入输出数据，依据仿真图验证系统建模的正确性。仿真测试程序中时序设计是很关键，体现了时序关系是否正确和合理，可以多设计仿真测试程序进行多次验证系统建模的正确性，以及时序关系正确和合理。

硬件描述语言 Verilog 是硬件实现的设计语言，也是第 9 章硬件设计硬件 IP 软核的描述语言，可以由 Vivado 工具自动生成。也可以使用 Vivado 工具进行离散系统仿真。

习题

3.1 对例 2.6 时序检测器(Sequence)系统分别进行 Moore 型自动机模型和 Mealy 型自动机模型的 Modelsim 仿真。将模型、仿真代码和仿真结果写成实验报告。

3.2 使用 Modelsim 仿真工具对饮料售货机进行建模仿真。该饮料售货机可以售 3 种饮料：咖啡、水、可乐。每瓶咖啡售 5 元、每瓶可乐 4 元、每瓶水售 2 元，可接收 1 元、5 元、10 元，支持找零。将模型、仿真代码和仿真结果写成实验报告。

3.3 使用 Modelsim 仿真工具对交通路口红绿灯控制系统进行建模仿真。该南北

东西两个方向交通路口红绿灯正交控制系统：南北方向直行绿灯 40 秒，东西方向直行绿灯 30 秒，在直行时可以左转，右转始终是自由的。（正交控制系统是指南北方向为绿灯时东西方向为红灯，南北方向为红灯时东西方向为绿灯。）将模型、仿真代码和仿真结果写成实验报告。

3.4 使用 Simulink 仿真

- (1) 调整图 Gain 中的参数为 90， Gain1 为 30， Gain2 为 15 进行仿真
- (2) 在(1)的基础上将 Sine Wave 中频率调整到 6 进行仿真。

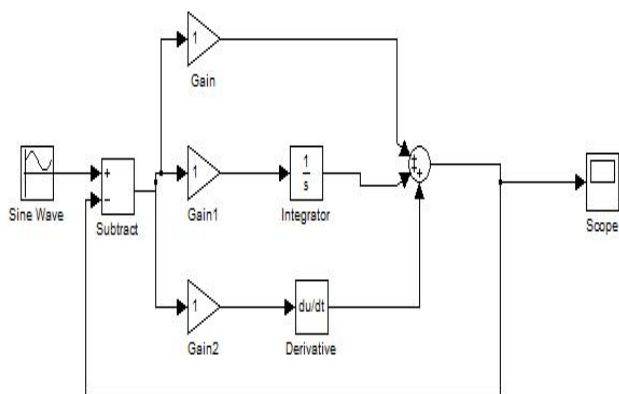


图 3-19 习题 3.4 图

3.5 制热空调参数调整

对制热空调系统的 Simulink 模型进行参数调整：OFF 状态下 $\exp(-0.02 \cdot t)$ 公式中参数值 -0.02 和 ON 状态下 $\exp(-0.08 \cdot t)$ 中参数值 -0.08 调整为组合对 (-0.01, -0.05) 和 (-0.05, -0.02)，对比参数调整后的仿真图。将模型、仿真代码和仿真结果写成实验报告。

3.6 使用 Simulink 对汽车自动停车系统进行仿真

汽车自动停车系统按照汽车分成三个阶段进行，第一阶段是匀减速行驶，减速的加速度是 $dv/dt = -1.35$ （米/秒平方），当车速到达每小时 20 公里速度时，进入第二阶段，第二阶段也是减速行驶，减速的加速度是 $dv/dt = 0.09t - 4.36$ （米/秒平方），当车速到达为零时，汽车进入第三阶段，停车。车速初始速度为 100 公里/小时。将建模与仿真结果写成实验报告。（此题注意量纲的一致性）

3.7 使用 Simulink 对水缸系统进行仿真

由两个水缸（缸 1 和缸 2）组成的水缸自动控制系统，水是按照一个常速通过一个软管流进水缸，在任何时刻水只能流进其中一个水缸，假定软管在两个水

缸瞬间转换,并保持两个水缸中的水在一定容量。两个水缸中的水都是按照常速 $V1$ 和 $V2$ 流出,水流进水缸恒均速度 w ,两个水缸容量都为 1 ,保持两个水缸容量为 $r1=r2$ 。设置两组起始状态是两个水缸的水容量,在每组起始状态下,设置三组流出速度 $V1$ 和 $V2$ 、水流速度 w 、以及水缸最低容量 $r1$ 和 $r2$,并对各组情况进行仿真。将含参数的 Simulink 模型以及对应的仿真结果写作实验报告。

3.8 调查了解业界离散系统、离散-连续系统仿真工具情况,阅读文献[17]掌握学术界模型驱动的仿真验证研究情况。