Remote Phone Appliance Control

Gary Plunkett (9913622)
Electrical Engineering 4OI4, McMaster University, Hamilton ON

Abstract:

Our project was to design a home phone appliance control system, which interfaces with a phone. Our design was required to answer the phone and take input from a touch tone keypad in order to turn appliances on and off. The UP1 board's FLEX chip was programmed using VHDL to interface with some external hardware, as well as a phone line, in order to complete this project. Though our design allowed for 10 devices to be controlled, only one remote-controlled AC switch was obtained to turn an AC appliance on and off. The other outputs control turning LEDs on and off. Currently, our device provides no feedback to the user and has no recording features for messages as an answering machine would. The design does, however, answer the phone without the aid of a standard answering machine. Our current design is fully functional and can be used as intended, but is not user friendly and would need the feedback features to become easier to use.

I. Introduction

A private telephone line consists of two wires. When the phone is not in use, there is a constant DC voltage across these wires of about 50V. When the phone rings, there is a 'ring voltage' sent down one of the wires. This 'ring voltage' is an AC voltage in the range of 100V. When the phone is taken off the hook, the DC voltage across the wires drops to about 12V to 15V as a resistive short (the phone) is placed on the line. Analog signals are sent between users when a call is set up.

Most phones in use now have touch tone keypads. For each key pressed these keypads generate two pure frequencies which can be heard as tones. Each tone has a unique set of frequencies. This is known as dual tone multiple frequency (DTMF). A table showing these frequencies with the corresponding key is shown in table 1 below.

Table 1: Table showing DTMF and keys[1]

	1209 Hz	1336 Hz	1477 Hz
697 Hz	1	2	3
770 Hz	4	5	6
852 Hz	7	8	9
941 Hz	*	0	#

Our design uses a DTMF decoder chip to detect the touch tone signals, as well as relay switches in detection of rings and answering the phone. The UP1 board and programmed FLEX chip control the relay for answering the phone as well as a remote relay switch for an appliance. These controls are based on inputs from the DTMF decoder and the ring detection circuitry.

II. Design Methodology

The design for our project consisted of external hardware circuitry and components as well as programmed functional units.

External Hardware

Our external hardware circuitry had to implement four functions. One was DTMF decoding, the next was ring detection, and the third was answering and hanging up the phone. The final function was turning a device on and off. The design and use of these will be reviewed here.

1. DTMF decoder:

For this function, we obtained a chip

manufactured for this purpose. The chip we used was Teltone's M-8870-01 chip. This chip takes an analog input signal and for each DTMF or touch tone signal it receives, it outputs a particular 4 bit binary code. The hexadecimal digit corresponding to each key pressed in given in table 2 below.

Table 2: DTMF decoded hex values[2]

KEY	1	2	3	4	5	6
HEX Digit	1	2	3	4	5	6
KEY	7	8	9	0	*	#
HEX Digit	7	8	9	A	B	C

The M-8870-01 chip also has an output pin named StD called delayed steering output. This pin outputs a high when the DTMF 4 bit signal is valid (when a touch tone key is pressed) and a low when the DTMF signal is not valid. When the DTMF 4 bit signal is not valid, it is in a high impedance state. This chip can be connected directly to a phone line. This chip and supporting circuitry which were given in the datasheet for the chip are shown in figure 1 below.

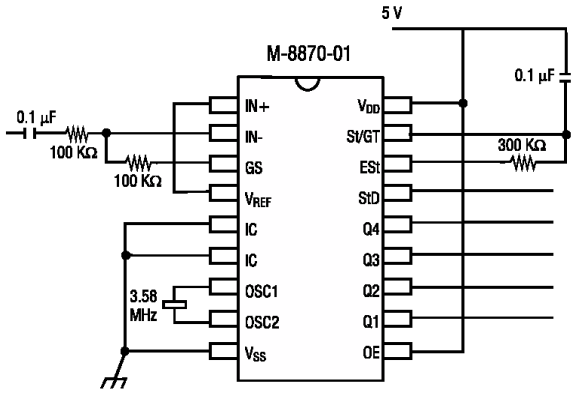

Figure 1: M-8870-01 chip with supporting circuit[2]

The external circuitry shown in figure 1did not work, however, for our project. Since there are two wires which must be connected from the phone we also had to use the IN+ pin. In this scheme the V_{REF} pin is not used. The chip with the supporting wiring used in our project is shown in figure 2 below.

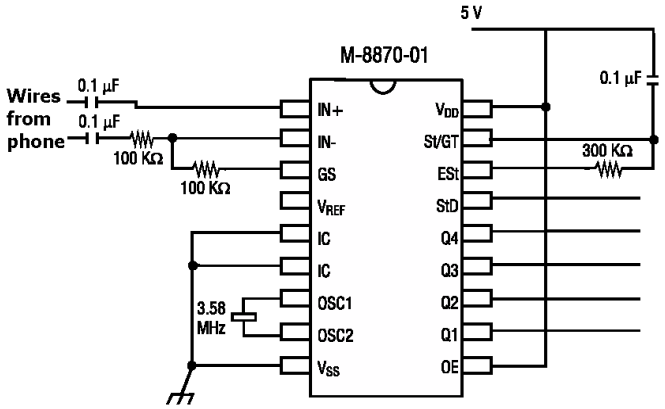

Figure 2: M-8870-01 chip and circuitry used in our implementation[2]

2. Ring detection:

The next hardware circuit which had to be implemented in our design was a ring detection circuit. Since a ring on a phone is a short 100VAC signal, an AC to DC transformer can be used to convert this to a DC pulse. In our case we used a 120VAC/12VDC 500mA transformer. Since a resistive load on the phone will take the phone off the hook, a capacitor with a large capacitance and high voltage rating is used in series with the transformer to create a near infinite DC impedance. Since the ring voltage may vary on different phone lines, the output of the transformer is sent to a relay switch. When the ring voltage powers the transformer, the relay switch is closed. In the switched state, the relay can be connected to a 5V source and in the non-switched state, the relay can be connected to ground. A schematic of this is shown in figure 3 below.

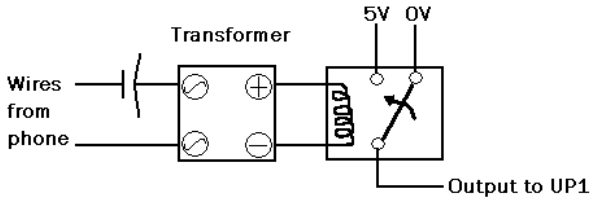

Figure 3: Ring detection circuit

3. Phone line control:

Our last external hardware feature is the phone line control. In order to answer the phone, a resistive short must be placed between the two phone wires. A straight short will not work because the analog signals will be shorted out as well and the DTMF signal will not be decoded by the decoding circuit. For our implementation we used a small 45 Ohm speaker in series with a ~3 KOhm resistor. In order to protect the UP1 board and FLEX chip, the switching is done through a relay. The UP1 board is not powerful enough to switch the relay, so an amplifier circuit is used to do this. The schematic for this circuit is shown in figure 4. The op-amp circuit used for this is shown in figure 5.

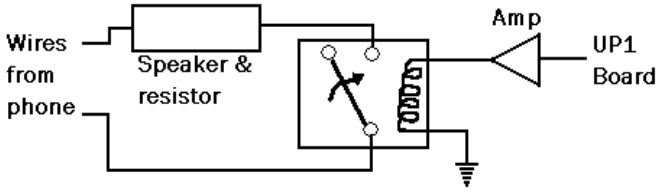

Figure 4: Phone answer circuit

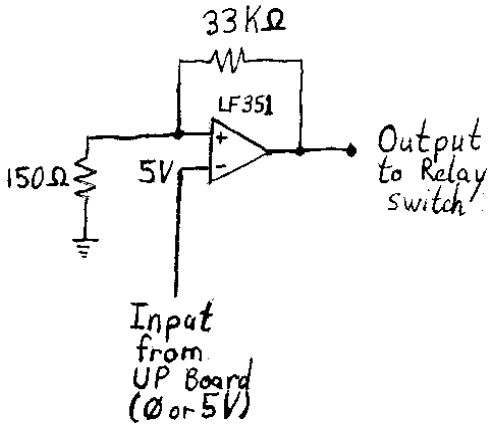

Figure 5: Op-amp circuit for phone answer circuit

4. Device Control

In this stage a remotely switched AC outlet was obtained. Wires were extended from the switch on the remote to plug into the UP1 board. This switch carried only a small DC voltage so was safe to use with the UP1 board. The particular one which we purchased worked by sending an RF signal to a switched outlet. To turn the outlet on, the switch needed to be closed for 3 to 5 seconds. To turn the device off, the switch had to be closed for a second or less. If the device was on and the switch was held for about 5 seconds, the switched outlet would turn off, and then back on again. This had to be considered in designing a controller for this switch in VHDL.

Programmed Functions

For our design we programmed 12 functional blocks in VHDL. Along with these, the debounce and clk_div UP1core functions were also used in our design. The twelve functional units which we programmed are listed as follows:

1. compare_unit
2. control_unit(a-j)
3. time_out_unit
4. count_rings
5. ring_time_out
6. phone_control
7. one_minute
8. select_dtmf
9. convert_to_zeros
10. my_pulse
11. my_pulse_2
12. pulse_five_clock

1. Compare Unit:

This block in our design was required to take input from the DTMF decoder to create a password, as well as verify a password. Once the password is verified, it must check the incoming DTMF codes (which are valid when the StD signal on the DTMF decoder chip is high) as the control codes.

For our implementation we had a 4 key PIN number. Once this is entered, a user will select a device to control from 0 to 9 since there are ten controllers. This is the first key entry in the control code. The next key entered selects turning the device on or off. In our implementation if the user enters a 1, the device should be turned on, and a 0 should turn the device off. The third key in the control code can be either the pound key (#) or the first digit of a time. The latter of these is only used for turning a device on since in our project, a device may be turned on with a time limit. If no time is necessary, the third key is all that is entered. With a time entry another key for the least significant digit for minutes is entered followed by the pound key (#). This key entry format is shown in table 3.

Table 3: key entry format for compare unit

4 Digit PIN	1 Digit Device ID	1 Digit On/Off	(#)	
4 Digit PIN	1 Digit Device ID	1 Digit On	2 Digit time	(#)

Once the compare unit receives these codes (terminated by #), it must forward them to the control units.

2. Control Units:

Our project implemented 10 control units. Each of these had a unique identity given by a hexadecimal number. This was done so that the compare unit could forward the device ID directly from the input to each controller to compare to its own ID. When the compare unit forwards a valid control code, each controller checks the ID first and if this matches, decides whether it should turn a device on or off and if a timer should be loaded. If a timer should be loaded, the controller passes the 2 hex digits from the control code which correspond to the time to be loaded to the timers.

3. Time Out Unit:

Each control unit has its own time out unit. These time out units will receive a load command as well as a time to be loaded into them. They will then begin timing down from this value in minutes. Once zero has been reached, the timer will send a signal back to the control unit informing the control unit that it has timed out and the device should now be turned off.

4. Count Rings:

In order to answer the phone, our design needs to be able to count how many rings have occurred consecutively. The count rings block will receive (debounced) pulses from the ring detection circuit. When the first pulse is received, it will start a timer. If four rings occur before the timer times out, a signal will be sent saying that the phone should be picked up.

5. Ring Time Out:

The ring time out circuit counts down the time in which four consecutive phone rings should occur in. When this time is over it sends a signal to the count rings circuit.

6. Phone Control:

This block controls answering and hanging up the phone. It will receive a signal from the count rings block and will answer the phone (by sending a high signal to the phone answer circuit). This high will also load a timer that will tell the phone control unit to hang up the line if this time limit is reached (which is done by sending a low signal to the phone answer circuit). It will also receive a signal from the compare unit which allows it to know when a valid control code has been entered. If a valid control code has been entered, the phone control will hang up the phone.

7. One Minute:

This is another timer unit which times the length of time the phone will be answered for. This

prevents the phone line from staying busy if a proper control code is not entered and the user hangs up.

8. Select DTMF:

This unit prevents the compare unit from detecting a valid DTMF code. This is required for when the controller has not answered the phone or a password is not being created. When the phone is answered or a password is being created, it allows the StD (or valid) signal from the DTMF decoder to be passed to the compare unit. When this is not the case the StD signal is not passed on.

9. Convert To Zeros:

When the control codes are passed to the control units, the time digits are passed in the form they were entered in from the keypad. From table 2, zero on the keypad will be passed as a hexadecimal 'A'. This time is also passed onto the timer units. The timer units will read this time as 10. So if a user has entered the time '01' on the keypad, the timer will read it as 10+1 or 11 minutes. To prevent this from happening when the control code is passed the bits corresponding to the time are checked and if there is a hexadecimal 'A', this is converted to a hexadecimal '0'.

10. My Pulse:

This is the same as the UP1core Library Function which pulses a signal for one clock cycle no matter how long the signal stays high for. This is used between blocks, such as the timers and control units. This is also used in our remote switch to turn our outlet off.

11. My Pulse 2:

This is the same as my pulse except that it introduces a delay of three clock cycles before the pulse occurs. This is used where a valid signal is sent and there are other values which must be set up and passed on a bus such as between the compare unit and the control units.

12. Pulse Five Clock:

This is the same as my pulse except that the pulse is high for five clock cycles instead of one. This was used to control the remote switch to turn the outlet on.

III. Analysis

Our project used roughly 3/4 of the programming space on the FLEX chip. In order to approximate the gates used by each block, the report files from the compiled units were looked at and the number of LCs and percentage of the chip used was determined. From this, and the knowledge that the FLEX chip is a 20,000 gate chip, we were able to obtain an estimate for the number of gates used by each unit. Table 4 summarizes these findings. The total percentage of each unit used was added together to determine an estimate for total percentage of chip used for the entire project. This number turned out to be larger than the estimate obtained from the report file of the completed project. However these percentages are rounded. The total number of LCs was also much higher when the total from all the units was compared to the actual. This may be because each unit, when put together with other units, may use parts of an LC that another unit is not using (ie for flip flops).

The timing analysis of our project was not very complex since the only units which used it were the timers which used a clock with a period of a second. However, certain aspects of the design had to be examined to ensure proper communication between units. When the compare unit passes codes to the control units, it sets a valid bit high indicating that the codes needed are on the bus. From the timing analyser, it was discovered that the time for the flag signal to propagate after the last key is pressed is 15.8ns. The slowest code bit propagated in 15.9ns. Also the code bits travel through the convert to zeros unit. The maximum propagation delay through this unit was 15.9ns as well. This meant that at least a 16ns delay was required for the flag from

the compare unit to the control units. In our design we used my_pulse_2 to accomplish this delay. This unit delays for three clock cycles. This was also connected to a 100Hz clock. This meant that the my_pulse_2 unit provided a 30ms delay for the flag, which was more than required. Since the user can only enter a valid code within a period of time measured in seconds, this is suitable for our design. From our timing analysis, however, a two clock cycle delay would have also been more than sufficient. A pulse unit with fewer clock cycle delays also uses less code and fewer gates, which would have made our design smaller.

Table 4: Functional units and estimated gates used

Unit	LCs	%	Qty.	Gates
count_rings	6	1	1	200
ring_time_out	14	2	1	400
phone_control	2	0	1	<150
select_dtmf	1	0	1	<150
control_unit	21	3	10	6000
my_pulse	2	0	17	~1200
one_minute	37	6	1	1200
compare_unit_5	124	21	1	4200
clk_div	38	6	1	1200
debounce	5	0	1	<150
convert_to_zeros	18	3	1	600
pulse_five_clock	6	1	1	200
Totals	2408	76		15200
remote_phone_actual	851	73		14600

IV. Simulations

Certain parts of our project were difficult to simulate properly since there were events occurring at various time scales. For instance, the compare unit can have a clock with a period in the range of microseconds and the user will enter codes in a range of seconds. A simulation with a clock of that period over that length cannot be achieved due to memory limitations of the computer.

Compare Unit:

For the compare unit various simulations were tested. For the simulation shown in figure 6, a 1 ms period clock was used and the simulation was run over a 10-second period of time. A PIN is created with the pressing and holding of pb1, and PIN entry and control code entry is simulated. The flag out goes high at the end of the entry and the control code is placed on the output of the compare unit. The valid flag out goes low again as soon as a new input is entered. This was the desired behaviour of the compare unit.

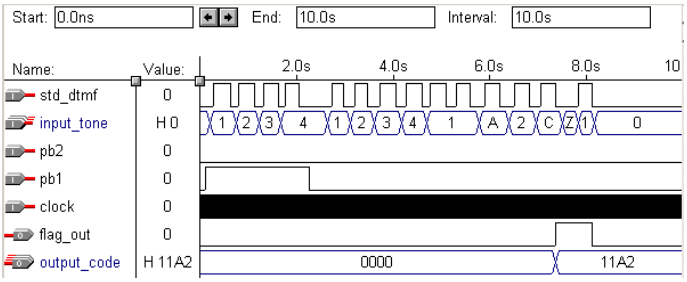

Figure 6: Timing simulation for compare unit

Control Units:

For the control units, a similar test was used as the one for the compare unit. A 1ms clock was used and various control codes were inputted to test the behaviour. The simulation was run over a 1 second period of time. The time out unit's flag had to be simulated as well. In figure 7, some control codes are inputted consecutively. In this case, the control codes are entered such that the controller turns the device off and then on again before the

timer times out. In this case the device is desired to stay turned on. This is shown to happen in our simulation.

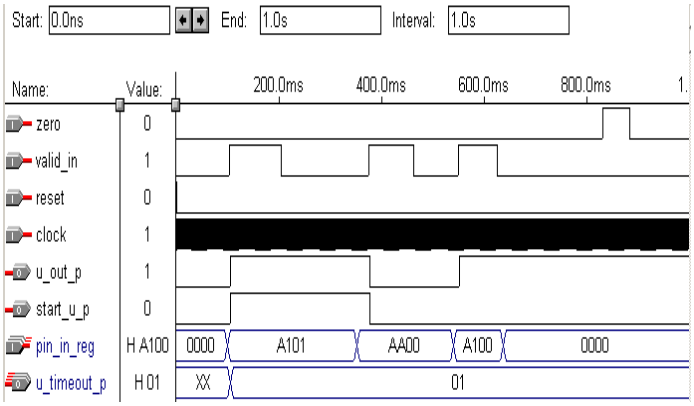

Figure 7: Control unit simulation with codes entered

Time Out Unit:

The time out unit uses a one second period clock to time the minutes which are loaded into it. When the time out unit first powers up it will output a high after a second. This will not affect the control unit as shown in the simulation of figure Y where the device was on and since a time wasn't loaded the time out flag did not turn the device off. The simulation in figure 8 shows the timer sending a high flag after a minute when a time of '01' in hexadecimal is loaded in.

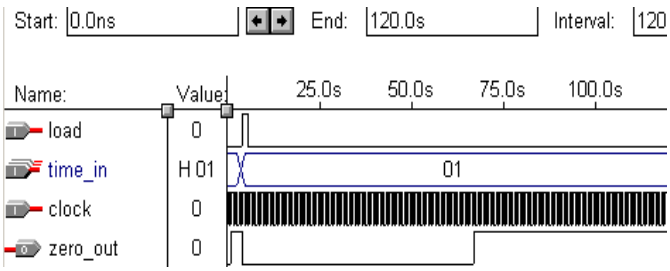

Figure 8: time out unit simulation

The other timer units work basically in the same manner.

V. Implementation

The VHDL portion of our project's design was put together using the schematic editor. Each unit was connected individually to other units. This approach led to a rather large schematic file. However, for testing and debugging purposes it was easier to make changes to the schematic in this form.

This schematic was compiled using Altera's MAX+plus II software and downloaded into the FLEX chip.

For our external circuitry, an old microprocessor kit was used for its power supply and breadboard. The kit has a +5V, +12V, -12V, and ground on its voltage supply. After compiling our code and doing some initial testing we found that the ground on the kit needed to be connected to a ground on the UP1 board in order for the inputs to the UP1 board to be read properly.

The DTMF decoder chip (M-8870-01) was connected with supporting circuitry on the breadboard and tested on a phone line and simulated usage with a phone connected to the +12V power supply. When connected directly to the phone, the M-8870-01 chip outputted proper codes for touch tone keys pressed. When connected to a phone powered by the +12V supply, the signal from the phone had to be amplified with an op-amp circuit.

The ring detection circuitry was connected to a phone line and tested for ability to switch the relay. The relay switched fine and the non-switched input on the relay was connected to the ground of the kit, while the switched input was connected to the +5V of the kit.

The resistive circuit (resistor and speaker) of the answer circuit was tested for ability to answer the phone. This was tested with a physical switch. With the resistive network connected, the phone was taken off the hook. Next, the amp circuit was tested for ability to switch the relay with a 5V input (from the UP1 board). This test was also successful. All the connections were soldered together with jumper wire. Once this was done, a phone cord was cut and

the (red and green) wires were soldered to the three circuits. Electrical tape was used to cover soldered wires and hold the parts to the kit.

For the remote AC switch, the remote was taken apart and jumper wires were soldered to the switch connections. These wires were connected directly to the UP1 board through the expansion header pins.

The listing of pin connections made on the UP1 board and flex chip are given in appendix A.

Once the circuitry was set up, tests were done in the lab in a simulated fashion. A push button on the UP1 board was used to simulate rings from the phone instead of from the ring detection circuit. The phone was connected directly to the +12V supply on the kit and connections were made from this to an amplifier and the output of this was connected to the DTMF decoder. LEDs on the UP1 board were used to determine the states of the devices (on or off) and the state of the phone line (off the hook or hung up). The remote AC switch was connected to device a which corresponds to selecting '0' on the keypad.

Various tests were done, turning devices in the lab on and off and for given lengths of time. A computer monitor across the room displaying a game implemented on the UP1 board was turned on and off using the phone connected to our device. With the remote-controlled switch it was possible to turn the monitor on and off manually as well using the remote. If our controller had previously turned the monitor off and we turned it back on manually, then used our device to try to turn it on, the monitor would turn off for a couple of seconds and back on again. This is due to the feature of the remote that was used. Since there was only one switch on the remote, a signal sent for 1 second turned the device off. In order to turn it on the switch had to be held for about 5 seconds. When our controller closed the switch for 5 seconds, the AC switch turned off and then on again. If a remote with two buttons had been obtained, one for on and one for off, this strange feature would not have occurred.

VI. Conclusions

Due to restrictions, mainly in time, but also in cost and perhaps in chip size, our implementation of the remote phone appliance control system had numerous limitations. It was, however, a success from what we had intended to accomplish for our project. Our main goal was to use our device in parallel with an answering machine to input codes from a phone to turn devices on and off. We surpassed this goal with the added feature of controlling the answering of the phone.

We have not provided any mechanism for feedback, however. A few simple methods of feedback have been thought of, such as an oscillator circuit which is enabled by the FLEX chip, to allow the user to know when a valid PIN has been entered and when a device has been successfully controlled. A more complicated version of this would be to play voice messages for greetings, instructions and confirmations. Some digital recorders could be used to this end. This may involve a significant amount of time researching how to interface with such a device, although it may not be beyond the scope of a future project in this course.

This project gave us the opportunity to create a practical device controlled using a chip programmed in VHDL.

VII. References

[1] **DTMF**, http://www.boondog.com/\tutorials\dtmf\dtmf.htm October 15, 2002

[2] **Clare M-8870 DTMF Receiver**, http://www.claremicronix.com/pdfs/tone_signaling/M8870-R3.pdf October 15, 2002

[3] James O. Hamblen and Michael D. Furman, *Rapid Prototyping Of Digital Systems* (Kluwer: Boston, MA, 2001).

Appendix A

Pin Connections

Pin	Name	Input/Output	Comments
6	output_a	output	display state of device 0 on segment of LED
7	output_b	output	display state of device 1 on segment of LED
8	output_c	output	display state of device 2 on segment of LED
9	output_d	output	display state of device 3 on segment of LED
11	output_e	output	display state of device 4 on segment of LED
12	output_f	output	display state of device 5 on segment of LED
13	output_g	output	display state of device 6 on segment of LED
17	output_h	output	display state of device 7 on segment of LED
18	output_i	output	display state of device 8 on segment of LED
19	output_j	output	display state of device 9 on segment of LED
20*	phone_hook	output	display state of phone line on segment of LED
28	set_password	input	Push button 1 to set PIN
29*	Ring_in	input	Push button 2 to simulate ring
91	Clock	input	
126	dtmf[0]	input	LSB from DTMF converter
127	dtmf[1]	input	
128	dtmf[2]	input	
129	dtmf[3]	input	MSB from DTMF converter
131	dtmf_std	input	valid signal for DTMF converter
132	ground	output	common ground for external circuitry
133	switch_1	output	one wire of remote controlled AC switch
134	switch_2	input	second wire of remote controlled AC switch
136**	Ring_in	input	Pin to ring detection circuit
137**	phone_hook	output	pin to amp for phone answer circuit

* These pin assignments are for simulation without a phone line

** These pin assignments are for use with an actual phone line

Appendix B
VHDL CODE

```
-----------------------------------------
--compare_unit
-----------------------------------------
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
USE ieee.std_logic_arith.all;
USE ieee.std_logic_unsigned.all;

-------------------------------------------------

entity comparing_unit_5 is
PORT(
            pb1, pb2, std_dtmf, clock     : in STD_LOGIC;
            input_tone                    : in std_logic_vector(3 downto 0);
            output_code                   : out std_logic_vector(15 downto 0);
            flag_out                      : out std_logic
);
end comparing_unit_5;

-------------------------------------------------

architecture a of comparing_unit_5 is

type password_type is array(0 to 3) of std_logic_vector(3 downto 0);

signal password         : password_type;
signal out_code         : std_logic_vector(15 downto 0);
signal pass_count : std_logic_vector(3 downto 0);
signal code_count : std_logic_vector(3 downto 0);
signal flag             : std_logic;
signal flag_2           : std_logic;
signal input            : std_logic_vector(3 downto 0);
signal input_2          : std_logic_vector(3 downto 0);
-------------------------------------------------
begin
--flag<=std_dtmf;
--flag_2<=std_dtmf;
-------------------------------------------------
process_clockit_1: process(clock)
begin
if (clock'event and clock='1') then
      if (pb1='1' and std_dtmf='1') then
            flag_2<='1';
            input_2<=input_tone;
      elsif (pb1='1' and std_dtmf='0') then
            flag_2<='0';
      end if;
end if;
end process process_clockit_1;
-----------------------------------------------------
process_enter_password: process(flag_2, pb2)
begin
if (pb2='1') then
      pass_count<="0000";
elsif (flag_2'event and flag_2='0') then
      if (pb1='1' and input_2/="0000") then
            pass_count<=pass_count+'1';
            if (pass_count="0000") then
                  password(0)<=input_2;
```

```
            elsif (pass_count="0001") then
                  password(1)<=input_2;
            elsif (pass_count="0010") then
                  password(2)<=input_2;
            elsif (pass_count="0011") then
                  password(3)<=input_2;
            end if;

      end if;
end if;
end process process_enter_password;
-------------------------------------------------
process_clockit: process(clock)
begin
if (clock'event and clock='1') then
      if (pb1='0' and std_dtmf='1') then
            flag<='1';
            input<=input_tone;
      elsif (pb1='0' and std_dtmf='0') then
            flag<='0';
      end if;
end if;
end process process_clockit;
-------------------------------------------------------------
-------------------------------------------------------------
process_get_input: process(flag, pb2)
begin
-----------------RESET--------------------------
if (pb2='1') then
      code_count<="0000";
      flag_out<='0';
      out_code<="0000000000000000";
-------------------------------------------------
-------------------------------------------------

------------CHECK PASSWORD----------------------

elsif (flag'event and flag='0') then
if (pb1='0' and input/="0000" and code_count="0000") then
      flag_out<='0';
      out_code<="0000000000000000";
      if (input=password(0)) then  --
            code_count<=code_count + '1';
      elsif (input/=password(0)) then --
            code_count<="0000";
      end if;

elsif (pb1='0' and input/="0000" and code_count="0001") then
      if (input=password(1)) then  --
            code_count<=code_count + '1';
      elsif (input/=password(1)) then --
            code_count<="0000";
      end if;

elsif (pb1='0' and input/="0000" and code_count="0010") then
      if (input=password(2)) then  --
            code_count<=code_count + '1';
      elsif (input/=password(2)) then --
            code_count<="0000";
```

```
      end if;

elsif (pb1='0' and input/="0000" and code_count="0011") then
      if (input=password(3)) then  --
            code_count<=code_count + '1';
      elsif (input/=password(3)) then --
            code_count<="0000";
      end if;

------END PASSWORD CHECK-------
------------------------------

-------------GET INPUT TO CONTROL DEVICES---------------

-- get device code
elsif (pb1='0' and input/="0000" and input/="1100" and input/="1011" and
code_count="0100") then
      out_code(15 downto 12)<=input;  --
      code_count<=code_count + '1';

-- device on or off, should only do if input = 1 OR input = 0
elsif (pb1='0' and (input="0001" or input="1010") and code_count="0101") then
      out_code(11 downto 8)<=input;  --
      code_count<=code_count + '1';

-- get time MSD---
elsif (pb1='0' and input/="0000" and input/="1100" and input/="1011" and
code_count="0110") then
      out_code(7 downto 4)<=input;  --
      code_count<=code_count + '1';

-- get time LSD---
elsif (pb1='0' and input/="0000" and input/="1100" and input/="1011" and
code_count="0111") then
      out_code(3 downto 0)<=input;  --
      code_count<=code_count + '1';

-- check if just device on or off ie no time # is pressed here
elsif (pb1='0' and input="1100" and code_count="0110") then
      output_code<=out_code;
      flag_out<='1';
      code_count<="0000";

--get # to end input
elsif (pb1='0' and input="1100" and code_count="1000") then
      output_code<=out_code;
      flag_out<='1';
      code_count<="0000";

end if;
end if;


end process process_get_input;

end a;
```

```
-----------------------------------
--control unit a
-----------------------------------
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
USE ieee.std_logic_arith.all;
USE ieee.std_logic_unsigned.all;
entity control_unit_a is
PORT(
      pin_in_reg                    : in std_logic_vector(15 downto 0);
      valid_in, clock               : in std_logic;
      zero : in std_logic;
      u_out_p : out std_logic;
      start_u_p: out std_logic;
      reset : in std_logic;
      u_timeout_p : out std_logic_vector(7 downto 0)
      );
end control_unit_a;

architecture a of control_unit_a is
signal identity :std_logic_vector(3 downto 0);
signal pin_in     :std_logic_vector(15 downto 0);
signal a_out      : std_logic;
signal start_a    : std_logic;
signal a_timeout  : std_logic_vector(7 downto 0);
signal valid      : std_logic;

begin

u_out_p<=a_out;

start_u_p<=start_a;

u_timeout_p<=a_timeout;

pin_in<=pin_in_reg;
-----------------------------------
process_clock_input: process(clock)
begin
if (clock'event and clock='1') then
      if (valid_in='1') then
            valid<='1';
      elsif (valid_in='0') then
            valid<='0';
      end if;
end if;
end process process_clock_input;
-----------------------------------


process_main: process(reset, valid, zero)
begin

if (reset='1') then
      a_out<='0';
      start_a<='0';

elsif (zero='1' and start_a='1') then
      a_out<='0';
```

```
      start_a<='0';

elsif (valid='1') then
      if (pin_in(11 downto 8)="1010") then
            if (pin_in(15 downto 12)="1010") then
                  a_out<='0';
                  start_a<='0';
            end if;
      elsif (pin_in(11 downto 8)="0001") then
            if (pin_in(15 downto 12)="1010") then
                  a_out<='1';
                  if (pin_in(7 downto 0)/="00000000") then
                        a_timeout<=pin_in(7 downto 0);
                        start_a<='1';
                  end if;
            end if;
      end if;
end if;


end process process_main;



end a;
```

```
-----------------------------------
--control unit b
-----------------------------------
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
USE ieee.std_logic_arith.all;
USE ieee.std_logic_unsigned.all;


entity control_unit_b is
PORT(
      pin_in_reg                    : in std_logic_vector(15 downto 0);
      valid_in, clock                     : in std_logic;
      zero : in std_logic;
      u_out_p : out std_logic;
      start_u_p: out std_logic;
      reset : in std_logic;
      u_timeout_p : out std_logic_vector(7 downto 0)
      );
end control_unit_b;

architecture a of control_unit_b is
signal identity :std_logic_vector(3 downto 0);
signal pin_in     :std_logic_vector(15 downto 0);
signal a_out      : std_logic;
signal start_a    : std_logic;
signal a_timeout  : std_logic_vector(7 downto 0);
signal valid      : std_logic;

begin
identity<="0001";

u_out_p<=a_out;

start_u_p<=start_a;

u_timeout_p<=a_timeout;

pin_in<=pin_in_reg;
-----------------------------------
process_clock_input: process(clock)
begin
if (clock'event and clock='1') then
      if (valid_in='1') then

            valid<='1';
      elsif (valid_in='0') then
            valid<='0';
      end if;
end if;
end process process_clock_input;
-----------------------------------


process_main: process(reset, valid, zero)
begin

if (reset='1') then
      a_out<='0';
```

```
      start_a<='0';

elsif (zero='1' and start_a='1') then
      a_out<='0';
      start_a<='0';

elsif (valid='1') then
      if (pin_in(11 downto 8)="1010") then
            if (pin_in(15 downto 12)=identity) then
                  a_out<='0';
                  start_a<='0';
            end if;
      elsif (pin_in(11 downto 8)="0001") then
            if (pin_in(15 downto 12)=identity) then
                  a_out<='1';
                  if (pin_in(7 downto 0)/="00000000") then
                        a_timeout<=pin_in(7 downto 0);
                        start_a<='1';
                  end if;
            end if;
      end if;
end if;


end process process_main;



end a;
```

```
---------------------------------------
--control unit c
---------------------------------------
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
USE ieee.std_logic_arith.all;
USE ieee.std_logic_unsigned.all;


entity control_unit_c is
PORT(
      pin_in_reg                    : in std_logic_vector(15 downto 0);
      valid_in, clock                     : in std_logic;
      zero : in std_logic;
      u_out_p : out std_logic;
      start_u_p: out std_logic;
      reset : in std_logic;
      u_timeout_p : out std_logic_vector(7 downto 0)
      );
end control_unit_c;

architecture a of control_unit_c is
signal identity :std_logic_vector(3 downto 0);
signal pin_in     :std_logic_vector(15 downto 0);
signal a_out      : std_logic;
signal start_a    : std_logic;
signal a_timeout  : std_logic_vector(7 downto 0);
signal valid      : std_logic;

begin
identity<="0010";

u_out_p<=a_out;

start_u_p<=start_a;

u_timeout_p<=a_timeout;

            pin_in<=pin_in_reg;
-----------------------------------
process_clock_input: process(clock)
begin
if (clock'event and clock='1') then
      if (valid_in='1') then

            valid<='1';
      elsif (valid_in='0') then
            valid<='0';
      end if;
end if;
end process process_clock_input;
-----------------------------------


process_main: process(reset, valid, zero)
begin

if (reset='1') then
      a_out<='0';
```

```
      start_a<='0';

elsif (zero='1' and start_a='1') then
      a_out<='0';
      start_a<='0';

elsif (valid='1') then
      if (pin_in(11 downto 8)="1010") then
            if (pin_in(15 downto 12)=identity) then
                  a_out<='0';
                  start_a<='0';
            end if;
      elsif (pin_in(11 downto 8)="0001") then
            if (pin_in(15 downto 12)=identity) then
                  a_out<='1';
                  if (pin_in(7 downto 0)/="00000000") then
                        a_timeout<=pin_in(7 downto 0);
                        start_a<='1';
                  end if;
            end if;
      end if;
end if;


end process process_main;



end a;
```

```
-----------------------------------
--control unit d
-----------------------------------
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
USE ieee.std_logic_arith.all;
USE ieee.std_logic_unsigned.all;


entity control_unit_d is
PORT(
      pin_in_reg                    : in std_logic_vector(15 downto 0);
      valid_in, clock                     : in std_logic;
      zero : in std_logic;
      u_out_p : out std_logic;
      start_u_p: out std_logic;
      reset : in std_logic;
      u_timeout_p : out std_logic_vector(7 downto 0)
      );
end control_unit_d;

architecture a of control_unit_d is
signal identity :std_logic_vector(3 downto 0);
signal pin_in     :std_logic_vector(15 downto 0);
signal a_out      : std_logic;
signal start_a    : std_logic;
signal a_timeout  : std_logic_vector(7 downto 0);
signal valid      : std_logic;

begin
identity<="0011";

u_out_p<=a_out;

start_u_p<=start_a;

u_timeout_p<=a_timeout;
            pin_in<=pin_in_reg;

-----------------------------------
process_clock_input: process(clock)
begin
if (clock'event and clock='1') then
      if (valid_in='1') then

            valid<='1';
      elsif (valid_in='0') then
            valid<='0';
      end if;
end if;
end process process_clock_input;
-----------------------------------


process_main: process(reset, valid, zero)
begin

if (reset='1') then
      a_out<='0';
```

```
      start_a<='0';

elsif (zero='1' and start_a='1') then
      a_out<='0';
      start_a<='0';

elsif (valid='1') then
      if (pin_in(11 downto 8)="1010") then
            if (pin_in(15 downto 12)=identity) then
                  a_out<='0';
                  start_a<='0';
            end if;
      elsif (pin_in(11 downto 8)="0001") then
            if (pin_in(15 downto 12)=identity) then
                  a_out<='1';
                  if (pin_in(7 downto 0)/="00000000") then
                        a_timeout<=pin_in(7 downto 0);
                        start_a<='1';
                  end if;
            end if;
      end if;
end if;


end process process_main;



end a;
```

```
-----------------------------------
--control unit e
-----------------------------------
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
USE ieee.std_logic_arith.all;
USE ieee.std_logic_unsigned.all;


entity control_unit_e is
PORT(
      pin_in_reg                    : in std_logic_vector(15 downto 0);
      valid_in, clock                     : in std_logic;
      zero : in std_logic;
      u_out_p : out std_logic;
      start_u_p: out std_logic;
      reset : in std_logic;
      u_timeout_p : out std_logic_vector(7 downto 0)
      );
end control_unit_e;

architecture a of control_unit_e is
signal identity :std_logic_vector(3 downto 0);
signal pin_in     :std_logic_vector(15 downto 0);
signal a_out      : std_logic;
signal start_a    : std_logic;
signal a_timeout  : std_logic_vector(7 downto 0);
signal valid      : std_logic;

begin
identity<="0100";

u_out_p<=a_out;

start_u_p<=start_a;

u_timeout_p<=a_timeout;

            pin_in<=pin_in_reg;
-----------------------------------
process_clock_input: process(clock)
begin
if (clock'event and clock='1') then
      if (valid_in='1') then

            valid<='1';
      elsif (valid_in='0') then
            valid<='0';
      end if;
end if;
end process process_clock_input;
-----------------------------------


process_main: process(reset, valid, zero)
begin

if (reset='1') then
      a_out<='0';
```

```
      start_a<='0';

elsif (zero='1' and start_a='1') then
      a_out<='0';
      start_a<='0';

elsif (valid='1') then
      if (pin_in(11 downto 8)="1010") then
            if (pin_in(15 downto 12)=identity) then
                  a_out<='0';
                  start_a<='0';
            end if;
      elsif (pin_in(11 downto 8)="0001") then
            if (pin_in(15 downto 12)=identity) then
                  a_out<='1';
                  if (pin_in(7 downto 0)/="00000000") then
                        a_timeout<=pin_in(7 downto 0);
                        start_a<='1';
                  end if;
            end if;
      end if;
end if;


end process process_main;



end a;
```

```
----------------------
--control unit f
----------------------
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
USE ieee.std_logic_arith.all;
USE ieee.std_logic_unsigned.all;


entity control_unit_f is
PORT(
      pin_in_reg                    : in std_logic_vector(15 downto 0);
      valid_in, clock                     : in std_logic;
      zero : in std_logic;
      u_out_p : out std_logic;
      start_u_p: out std_logic;
      reset : in std_logic;
      u_timeout_p : out std_logic_vector(7 downto 0)
      );
end control_unit_f;

architecture a of control_unit_f is
signal identity :std_logic_vector(3 downto 0);
signal pin_in     :std_logic_vector(15 downto 0);
signal a_out      : std_logic;
signal start_a    : std_logic;
signal a_timeout  : std_logic_vector(7 downto 0);
signal valid      : std_logic;

begin
identity<="0101";

u_out_p<=a_out;

start_u_p<=start_a;

u_timeout_p<=a_timeout;

            pin_in<=pin_in_reg;
-----------------------------------
process_clock_input: process(clock)
begin
if (clock'event and clock='1') then
      if (valid_in='1') then

            valid<='1';
      elsif (valid_in='0') then
            valid<='0';
      end if;
end if;
end process process_clock_input;
-----------------------------------


process_main: process(reset, valid, zero)
begin

if (reset='1') then
      a_out<='0';
```

```
      start_a<='0';

elsif (zero='1' and start_a='1') then
      a_out<='0';
      start_a<='0';

elsif (valid='1') then
      if (pin_in(11 downto 8)="1010") then
            if (pin_in(15 downto 12)=identity) then
                  a_out<='0';
                  start_a<='0';
            end if;
      elsif (pin_in(11 downto 8)="0001") then
            if (pin_in(15 downto 12)=identity) then
                  a_out<='1';
                  if (pin_in(7 downto 0)/="00000000") then
                        a_timeout<=pin_in(7 downto 0);
                        start_a<='1';
                  end if;
            end if;
      end if;
end if;


end process process_main;



end a;
```

```
----------------------------
--control unit g
----------------------------
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
USE ieee.std_logic_arith.all;
USE ieee.std_logic_unsigned.all;

entity control_unit_g is
PORT(
      pin_in_reg                    : in std_logic_vector(15 downto 0);
      valid_in, clock                     : in std_logic;
      zero : in std_logic;
      u_out_p : out std_logic;
      start_u_p: out std_logic;
      reset : in std_logic;
      u_timeout_p : out std_logic_vector(7 downto 0)
      );
end control_unit_g;

architecture a of control_unit_g is
signal identity :std_logic_vector(3 downto 0);
signal pin_in     :std_logic_vector(15 downto 0);
signal a_out      : std_logic;
signal start_a    : std_logic;
signal a_timeout  : std_logic_vector(7 downto 0);
signal valid      : std_logic;

begin
identity<="0110";

u_out_p<=a_out;

start_u_p<=start_a;

u_timeout_p<=a_timeout;

            pin_in<=pin_in_reg;
-----------------------------------
process_clock_input: process(clock)
begin
if (clock'event and clock='1') then
      if (valid_in='1') then

            valid<='1';
      elsif (valid_in='0') then
            valid<='0';
      end if;
end if;
end process process_clock_input;
-----------------------------------


process_main: process(reset, valid, zero)
begin

if (reset='1') then
      a_out<='0';
      start_a<='0';
```

```
elsif (zero='1' and start_a='1') then
      a_out<='0';
      start_a<='0';

elsif (valid='1') then
      if (pin_in(11 downto 8)="1010") then
            if (pin_in(15 downto 12)=identity) then
                  a_out<='0';
                  start_a<='0';
            end if;
      elsif (pin_in(11 downto 8)="0001") then
            if (pin_in(15 downto 12)=identity) then
                  a_out<='1';
                  if (pin_in(7 downto 0)/="00000000") then
                        a_timeout<=pin_in(7 downto 0);
                        start_a<='1';
                  end if;
            end if;
      end if;
end if;


end process process_main;



end a;
```

```
----------------------------
--control unit h
----------------------------
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
USE ieee.std_logic_arith.all;
USE ieee.std_logic_unsigned.all;


entity control_unit_h is
PORT(
      pin_in_reg                    : in std_logic_vector(15 downto 0);
      valid_in, clock                     : in std_logic;
      zero : in std_logic;
      u_out_p : out std_logic;
      start_u_p: out std_logic;
      reset : in std_logic;
      u_timeout_p : out std_logic_vector(7 downto 0)
      );
end control_unit_h;

architecture a of control_unit_h is
signal identity :std_logic_vector(3 downto 0);
signal pin_in     :std_logic_vector(15 downto 0);
signal a_out      : std_logic;
signal start_a    : std_logic;
signal a_timeout  : std_logic_vector(7 downto 0);
signal valid      : std_logic;

begin
identity<="0111";

u_out_p<=a_out;

start_u_p<=start_a;

u_timeout_p<=a_timeout;

            pin_in<=pin_in_reg;
-----------------------------------
process_clock_input: process(clock)
begin
if (clock'event and clock='1') then
      if (valid_in='1') then

            valid<='1';
      elsif (valid_in='0') then
            valid<='0';
      end if;
end if;
end process process_clock_input;
-----------------------------------


process_main: process(reset, valid, zero)
begin

if (reset='1') then
      a_out<='0';
```

```
      start_a<='0';

elsif (zero='1' and start_a='1') then
      a_out<='0';
      start_a<='0';

elsif (valid='1') then
      if (pin_in(11 downto 8)="1010") then
            if (pin_in(15 downto 12)=identity) then
                  a_out<='0';
                  start_a<='0';
            end if;
      elsif (pin_in(11 downto 8)="0001") then
            if (pin_in(15 downto 12)=identity) then
                  a_out<='1';
                  if (pin_in(7 downto 0)/="00000000") then
                        a_timeout<=pin_in(7 downto 0);
                        start_a<='1';
                  end if;
            end if;
      end if;
end if;


end process process_main;



end a;
```

```
------------------------------
--control unit i
------------------------------
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
USE ieee.std_logic_arith.all;
USE ieee.std_logic_unsigned.all;


entity control_unit_i is
PORT(
      pin_in_reg                    : in std_logic_vector(15 downto 0);
      valid_in, clock                     : in std_logic;
      zero : in std_logic;
      u_out_p : out std_logic;
      start_u_p: out std_logic;
      reset : in std_logic;
      u_timeout_p : out std_logic_vector(7 downto 0)
      );
end control_unit_i;

architecture a of control_unit_i is
signal identity :std_logic_vector(3 downto 0);
signal pin_in     :std_logic_vector(15 downto 0);
signal a_out      : std_logic;
signal start_a    : std_logic;
signal a_timeout  : std_logic_vector(7 downto 0);
signal valid      : std_logic;

begin
identity<="1000";

u_out_p<=a_out;

start_u_p<=start_a;

u_timeout_p<=a_timeout;

            pin_in<=pin_in_reg;
-----------------------------------
process_clock_input: process(clock)
begin
if (clock'event and clock='1') then
      if (valid_in='1') then

            valid<='1';
      elsif (valid_in='0') then
            valid<='0';
      end if;
end if;
end process process_clock_input;
-----------------------------------


process_main: process(reset, valid, zero)
begin

if (reset='1') then
      a_out<='0';
```

```
      start_a<='0';

elsif (zero='1' and start_a='1') then
      a_out<='0';
      start_a<='0';

elsif (valid='1') then
      if (pin_in(11 downto 8)="1010") then
            if (pin_in(15 downto 12)=identity) then
                  a_out<='0';
                  start_a<='0';
            end if;
      elsif (pin_in(11 downto 8)="0001") then
            if (pin_in(15 downto 12)=identity) then
                  a_out<='1';
                  if (pin_in(7 downto 0)/="00000000") then
                        a_timeout<=pin_in(7 downto 0);
                        start_a<='1';
                  end if;
            end if;
      end if;
end if;


end process process_main;



end a;
```

```
---------------------------
--control unit j
---------------------------
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
USE ieee.std_logic_arith.all;
USE ieee.std_logic_unsigned.all;


entity control_unit_j is
PORT(
      pin_in_reg                    : in std_logic_vector(15 downto 0);
      valid_in, clock                     : in std_logic;
      zero : in std_logic;
      u_out_p : out std_logic;
      start_u_p: out std_logic;
      reset : in std_logic;
      u_timeout_p : out std_logic_vector(7 downto 0)
      );
end control_unit_j;

architecture a of control_unit_j is
signal identity :std_logic_vector(3 downto 0);
signal pin_in     :std_logic_vector(15 downto 0);
signal a_out      : std_logic;
signal start_a    : std_logic;
signal a_timeout  : std_logic_vector(7 downto 0);
signal valid      : std_logic;

begin
identity<="1001";

u_out_p<=a_out;

start_u_p<=start_a;

u_timeout_p<=a_timeout;

            pin_in<=pin_in_reg;
-----------------------------------
process_clock_input: process(clock)
begin
if (clock'event and clock='1') then
      if (valid_in='1') then

            valid<='1';
      elsif (valid_in='0') then
            valid<='0';
      end if;
end if;
end process process_clock_input;
-----------------------------------


process_main: process(reset, valid, zero)
begin

if (reset='1') then
      a_out<='0';
```

```
      start_a<='0';

elsif (zero='1' and start_a='1') then
      a_out<='0';
      start_a<='0';

elsif (valid='1') then
      if (pin_in(11 downto 8)="1010") then
            if (pin_in(15 downto 12)=identity) then
                  a_out<='0';
                  start_a<='0';
            end if;
      elsif (pin_in(11 downto 8)="0001") then
            if (pin_in(15 downto 12)=identity) then
                  a_out<='1';
                  if (pin_in(7 downto 0)/="00000000") then
                        a_timeout<=pin_in(7 downto 0);
                        start_a<='1';
                  end if;
            end if;
      end if;
end if;


end process process_main;



end a;
```

```
---------------------------
--convert to zeros
---------------------------
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
USE ieee.std_logic_arith.all;
USE ieee.std_logic_unsigned.all;

entity convert_to_zeros is
PORT(
      input : in std_logic_vector(15 downto 0);
      output      : out std_logic_vector(15 downto 0)
);
end convert_to_zeros;

architecture a of convert_to_zeros is
begin
process_name: process(input)
begin
if (input(7 downto 4)="1010") then

      output(7 downto 4)<="0000";
      output(3 downto 0)<=input(3 downto 0);
      output(15 downto 8)<=input(15 downto 8);

elsif (input(3 downto 0)="1010") then

      output(3 downto 0)<="0000";
      output(15 downto 4)<=input(15 downto 4);
else
      output<=input;
end if;
end process process_name;
end a;
```

```
----------------------------
--count rings
--REFERENCES: James O. Hamblen and Michael D. Furman,
--Rapid Prototyping Of Digital Systems (Kluwer:Boston, MA, 2001).
----------------------------
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
USE ieee.std_logic_arith.all;
USE ieee.std_logic_unsigned.all;


entity count_rings is
PORT(
      pulse_in, hang_up, time_out   : in std_logic;
      pick_up, load_timer                 : out std_logic
      );
end count_rings;

architecture a of count_rings is

signal counter    : std_logic_vector(2 downto 0);

begin

process_count_rings: process(pulse_in, time_out, hang_up)
begin
--if (hang_up='1') then
--    counter<="000";
--    pick_up<='0';
--    load_timer<='0';
if (hang_up='1' or time_out='1') then
            load_timer<='0';
            counter<="000";
            pick_up<='0';

elsif (pulse_in'event and pulse_in='1') then
      if (counter<="000") then
            pick_up<='0';
            load_timer<='1';
            counter<=counter+1;
      elsif (counter<"011") then
            pick_up<='0';
            counter<=counter + 1;
      elsif(counter="011") then
            pick_up<='1';
            counter<="000";
            load_timer<='0';
      end if;
end if;
end process process_count_rings;

end a;
```

```
------------------------
--my pulse
--REFERENCES: James O. Hamblen and Michael D. Furman,
--Rapid Prototyping Of Digital Systems (Kluwer:Boston, MA, 2001).
------------------------
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
USE ieee.std_logic_arith.all;
USE ieee.std_logic_unsigned.all;


entity my_pulse is
PORT(
      input, clock      : in std_logic;
      output                  : out std_logic
      );
end my_pulse;

architecture a of my_pulse is
signal flag : std_logic_vector(0 downto 0);

begin

process_go_low: process(clock)
begin
if (clock'event and clock='1') then
      if (input='1' and flag="0") then
            output<='1';
            flag<="1";
      elsif (input='1' and flag="1") then
            output<='0';
      elsif (input='0') then
            flag<="0";
      end if;
end if;
end process process_go_low;
end a;
```

```
-------------------------
--my pulse 2
--REFERENCES: James O. Hamblen and Michael D. Furman,
--Rapid Prototyping Of Digital Systems (Kluwer:Boston, MA, 2001).
-------------------------
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
USE ieee.std_logic_arith.all;
USE ieee.std_logic_unsigned.all;


entity my_pulse_2 is
PORT(
      input, clock      : in std_logic;
      output                  : out std_logic
      );
end my_pulse_2;

architecture a of my_pulse_2 is
signal flag : std_logic_vector(1 downto 0);

begin

process_go_low: process(clock)
begin
if (clock'event and clock='1') then
      if (input='1' and flag<"10") then
            flag<=flag+1;
      elsif (input='1' and flag="10") then
            output<='1';
            flag<=flag+1;
      elsif (input='1' and flag="11") then
            output<='0';
      elsif (input='0') then
            flag<="00";
      end if;
end if;
end process process_go_low;
end a;
```

```
---------------------
--one minute
--REFERENCES: James O. Hamblen and Michael D. Furman,
--Rapid Prototyping Of Digital Systems (Kluwer:Boston, MA, 2001).
---------------------
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
USE ieee.std_logic_arith.all;
USE ieee.std_logic_unsigned.all;


entity one_minute is
PORT(
      clock, load             : in std_logic;
      zero_out                : out std_logic
      );
end one_minute;

architecture a of one_minute is
signal seconds_1  : std_logic_vector(3 downto 0);
signal seconds_2  : std_logic_vector(3 downto 0);
signal minutes_1  : std_logic_vector(3 downto 0);
signal minutes_2  : std_logic_vector(3 downto 0);
signal max_nine : std_logic_vector(3 downto 0);
signal max_five   : std_logic_vector(3 downto 0);
signal min_zero   : std_logic_vector(3 downto 0);
signal load_flag_2      : std_logic;

begin
max_nine<="1001";
max_five<="0101";
min_zero<="0000";

process_clock: process(clock, load)
begin
if (load='1') then
      minutes_1<="0001";  --time_in(3 downto 0);
      minutes_2<="0000";  --time_in(7 downto 4);
      seconds_1<="0000";
      seconds_2<="0000";
      zero_out<='0';
elsif (clock'event and clock='1') then
      if (seconds_1>min_zero) then
      seconds_1<=seconds_1 - 1;
    elsif (seconds_2>min_zero) then
          seconds_2<=seconds_2 - 1;
      seconds_1<=max_nine;
    elsif (minutes_1>min_zero) then
      minutes_1<=minutes_1 - 1;
      seconds_2<=max_five;
      seconds_1<=max_nine;
    elsif (minutes_2>min_zero) then
      minutes_2<=minutes_2 - 1;
      minutes_1<=max_nine;
      seconds_2<=max_nine;
      seconds_1<=max_nine;
    else
      minutes_2<="0000";
      minutes_1<="0000";
```

```
      seconds_2<="0000";
      seconds_1<="0000";
      zero_out<='1';

   end if;

end if;
end process process_clock;

end a;
```

```
------------------------------
--phone control
------------------------------

LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
USE ieee.std_logic_arith.all;
USE ieee.std_logic_unsigned.all;


entity phone_control is
PORT(
      rings_in, hang_up, time_out   :     in std_logic;
      open_line                           :     out std_logic
      );
end phone_control;

architecture a of phone_control is

begin

process_control_phone: process(rings_in, hang_up, time_out)
begin
if (hang_up='1' or time_out='1') then
      open_line<='0';
elsif (rings_in'event and rings_in='1') then
      open_line<='1';
end if;

end process process_control_phone;

end a;
```

```
---------------------
--pulse five clock
--REFERENCES: James O. Hamblen and Michael D. Furman,
--Rapid Prototyping Of Digital Systems (Kluwer:Boston, MA, 2001).
---------------------

LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
USE ieee.std_logic_arith.all;
USE ieee.std_logic_unsigned.all;


entity pulse_five_clock is
PORT(
      input, clock      : in std_logic;
      output                  : out std_logic
      );
end pulse_five_clock;

architecture a of pulse_five_clock is
signal flag : std_logic_vector(2 downto 0);

begin

process_go_low: process(clock)
begin
if (clock'event and clock='1') then
      if (input='1' and flag<"101") then
            output<='1';
            flag<=flag+1;
      elsif (input='1' and flag="101") then
            output<='1';
            flag<=flag+1;
      elsif (input='1' and flag="110") then
            output<='0';
      elsif (input='0') then
            flag<="000";
      end if;
end if;
end process process_go_low;
end a;
```

```
--------------------------
--ring time out
--REFERENCES: James O. Hamblen and Michael D. Furman,
--Rapid Prototyping Of Digital Systems (Kluwer:Boston, MA, 2001).
--------------------------

LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
USE ieee.std_logic_arith.all;
USE ieee.std_logic_unsigned.all;


entity ring_time_out is
PORT(
      clock, load : in std_logic;
      zero_out    : out std_logic
      );
end ring_time_out;

architecture a of ring_time_out is
signal seconds_1  : std_logic_vector(3 downto 0);
signal seconds_2  : std_logic_vector(3 downto 0);
signal max_nine : std_logic_vector(3 downto 0);
signal max_five   : std_logic_vector(3 downto 0);
signal min_zero   : std_logic_vector(3 downto 0);
signal load_flag_2      : std_logic;

begin
max_nine<="1001";
max_five<="0101";
min_zero<="0000";




process_clock: process(clock, load)
begin
if (load='1') then
      seconds_1<="0101";
      seconds_2<="0010";
      zero_out<='0';
elsif (clock'event and clock='1') then
      if (seconds_1>min_zero) then
      seconds_1<=seconds_1 - 1;
    elsif (seconds_2>min_zero) then
      seconds_2<=seconds_2 - 1;
      seconds_1<=max_nine;
    else
      seconds_2<="0000";
      seconds_1<="0000";
      zero_out<='1';
   end if;
end if;
end process process_clock;

end a;
```

```
------------------------
--select dtmf
------------------------

LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
USE ieee.std_logic_arith.all;
USE ieee.std_logic_unsigned.all;

entity select_dtmf is
PORT(
      pick_up, set_password   :     in std_logic;
      std_valid_in                  :     in std_logic;
      std_valid_out                 :     out std_logic
      );
end select_dtmf;

architecture a of select_dtmf is
begin
process_use_dtmf: process(pick_up, set_password)
begin
if (pick_up='1' or set_password='1') then
      std_valid_out<=std_valid_in;
else
      std_valid_out<='0';
end if;
end process process_use_dtmf;

end a;
```

```
-----------------------
--time out unit
--REFERENCES: James O. Hamblen and Michael D. Furman,
--Rapid Prototyping Of Digital Systems (Kluwer:Boston, MA, 2001).
----------------------

LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
USE ieee.std_logic_arith.all;
USE ieee.std_logic_unsigned.all;

entity time_out_unit is
PORT(
      clock, load : in std_logic;
      time_in     : in std_logic_vector(7 downto 0);
      zero_out    : out std_logic
      );
end time_out_unit;

architecture a of time_out_unit is
signal seconds_1  : std_logic_vector(3 downto 0);
signal seconds_2  : std_logic_vector(2 downto 0);
signal minutes_1  : std_logic_vector(3 downto 0);
signal minutes_2  : std_logic_vector(3 downto 0);
signal max_nine : std_logic_vector(3 downto 0);
signal max_five   : std_logic_vector(2 downto 0);
signal min_zero   : std_logic_vector(3 downto 0);
signal load_flag_2      : std_logic;
begin
max_nine<="1001";
max_five<="101";
min_zero<="0000";


process_clock: process(clock, load)
begin
if (load='1') then
      minutes_1<=time_in(3 downto 0);
      minutes_2<=time_in(7 downto 4);
      seconds_1<="0001";
      seconds_2<="000";
      zero_out<='0';

elsif (clock'event and clock='1') then
    if (seconds_1>min_zero) then
      seconds_1<=seconds_1 - 1;
    elsif (seconds_2>min_zero) then
          seconds_2<=seconds_2 - 1;
      seconds_1<=max_nine;
    elsif (minutes_1>min_zero) then
      minutes_1<=minutes_1 - 1;
      seconds_2<=max_five;
      seconds_1<=max_nine;
    elsif (minutes_2>min_zero) then
      minutes_2<=minutes_2 - 1;
      minutes_1<=max_nine;
      seconds_2<=max_five;
      seconds_1<=max_nine;
    else
```

```
      minutes_2<="0000";
      minutes_1<="0000";
      seconds_2<="000";
      seconds_1<="0000";
      zero_out<='1';
    end if;

end if;
end process process_clock;

end a;
```