UNIVERSITI TEKNOLOGI MALAYSIA

MEL 1153: CAD FOR ELECTRONIC DESIGN (SEC 03 -PESISIR-)

-VENDING MACHINE-

LECTURER
DR MUHAMMAD NADZIR BIN MARSONO

GROUP MEMBER

WAN AHMAD ZAINIE BIN WAN MOHAMAD	(ME131135)
AZFAR 'AIZAT BIN MOHD ISA	(ME131032)

SEM02 20132014

Table of Contents

1 DESIGN SPECIFICATION

1.1 DESCRIPTION

A vending machine is a machine which dispenses items such as snacks and beverages to customers automatically, after the customer inserts currency into the machine. By using computer-aided-design (CAD) tools and hardware description language (HDL), which is Quartus II version 13.1 to design Vending Machine system. The technique been used register transfer level (RTL) design.

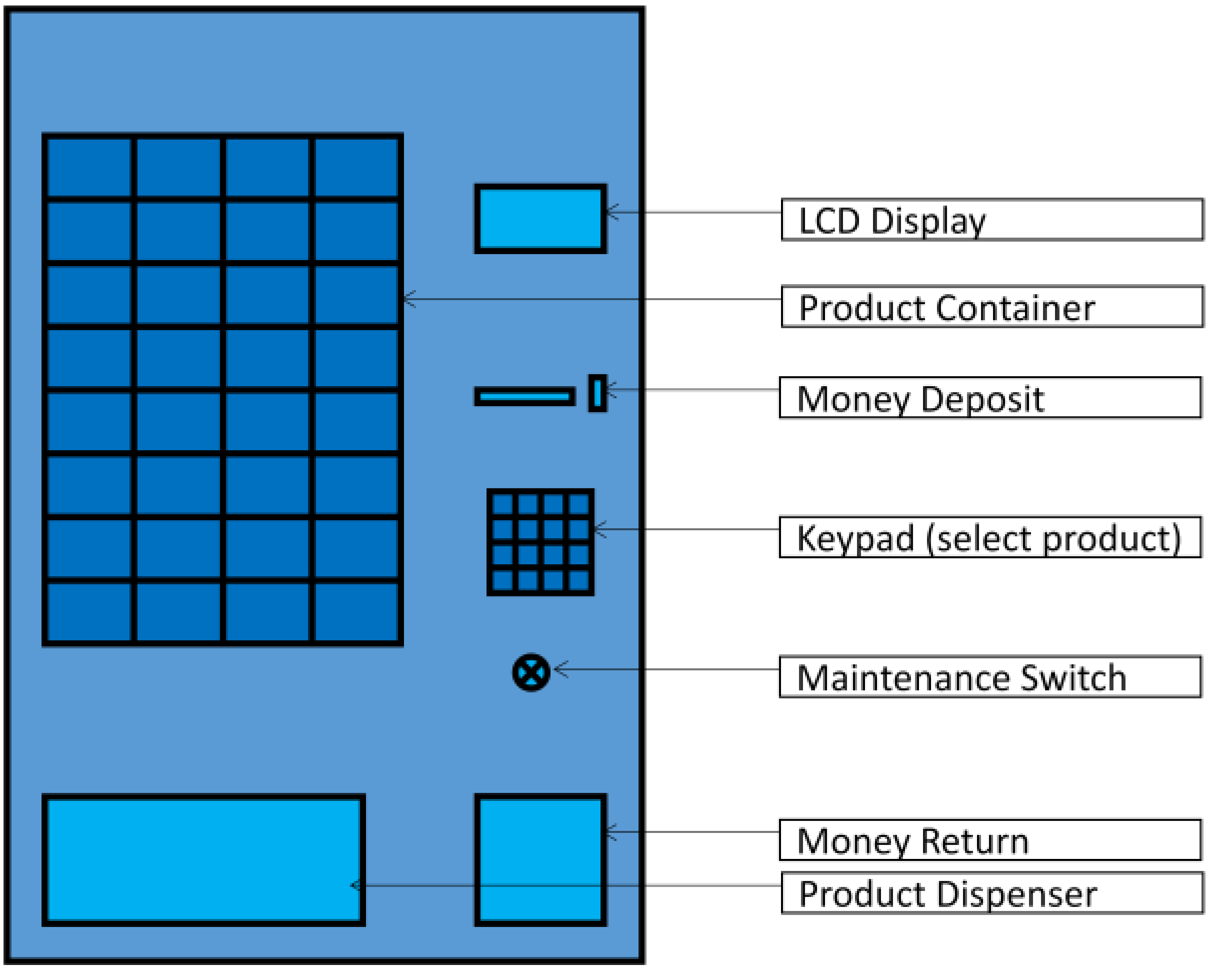
The specifications

Notes / Coin : Can receive RM0.10, RM0.20 RM0.50, RM1.00, RM5.00, RM10.00, RM20.00 and RM50.00

Product : Maximum 32 items

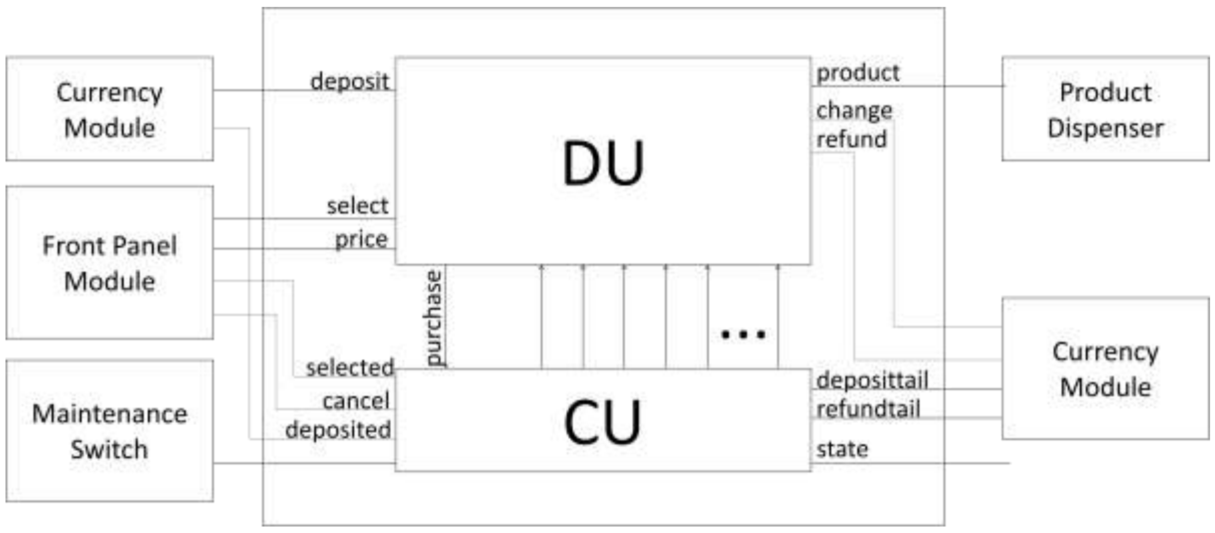
Maintenance : Can change price of products

1.2 INTERFACE



2 CONCEPTUAL ARCHITECTURE

2.1 FUNCTIONAL BLOCK DIAGRAM



Currency Module:

Deposit: represent amout of note/coin inserted:

Notes / Coins	Represent
RM0.10	1
RM0.20	2
RM0.50	5
RM1.00	10
RM5.00	50
RM10.00	100
RM20.00	200
RM50.00	500

Deposited: signal indicating note/coin inserted

Change: dispense the balance after purchase, assuming always enough note/coin is larger

Refund: signal to dispense last inserted note/coin is larger than RM50.00

Deposittail : store the inserted note/coin on purchase

Front Panel Module:

Select: represent button for selecting the product. Maximum 32 items

Selected: signal to indicate one of the buttons is pressed

Cancel: to cancel purchase process, get all money back

Maintenance Switch:

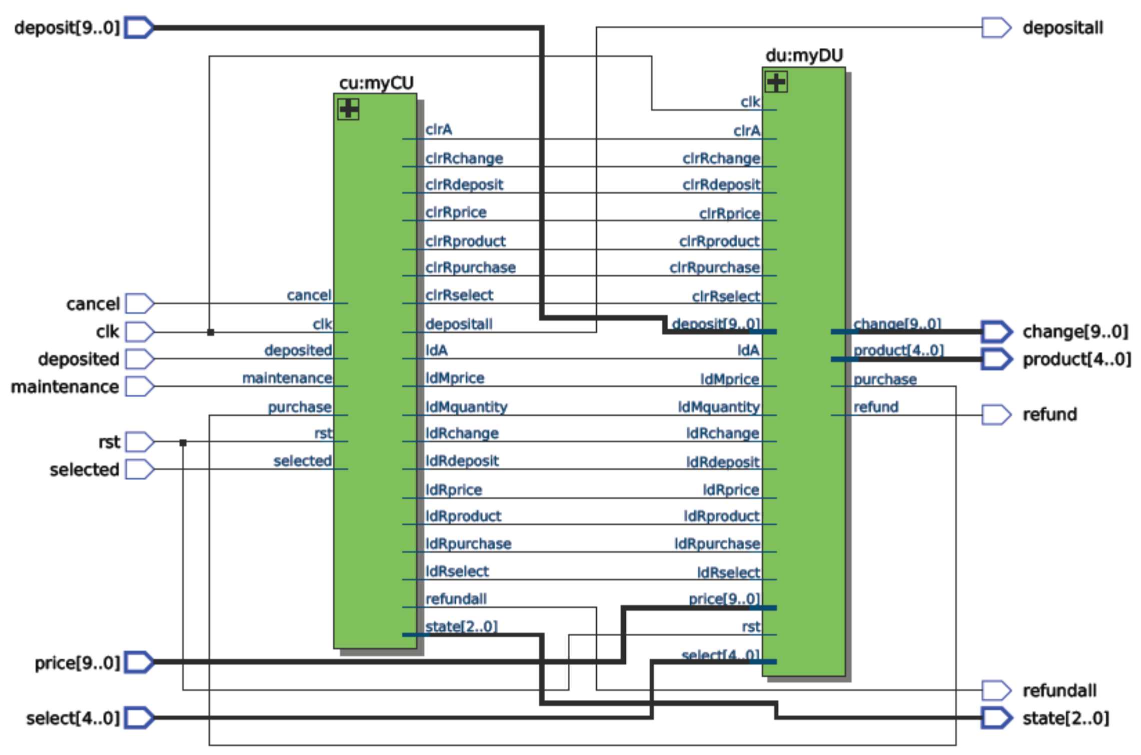
Maintenance: signal to enter maintenance mode

Price: key in price, assume there is keypad, and press select button to change the price of that particular product

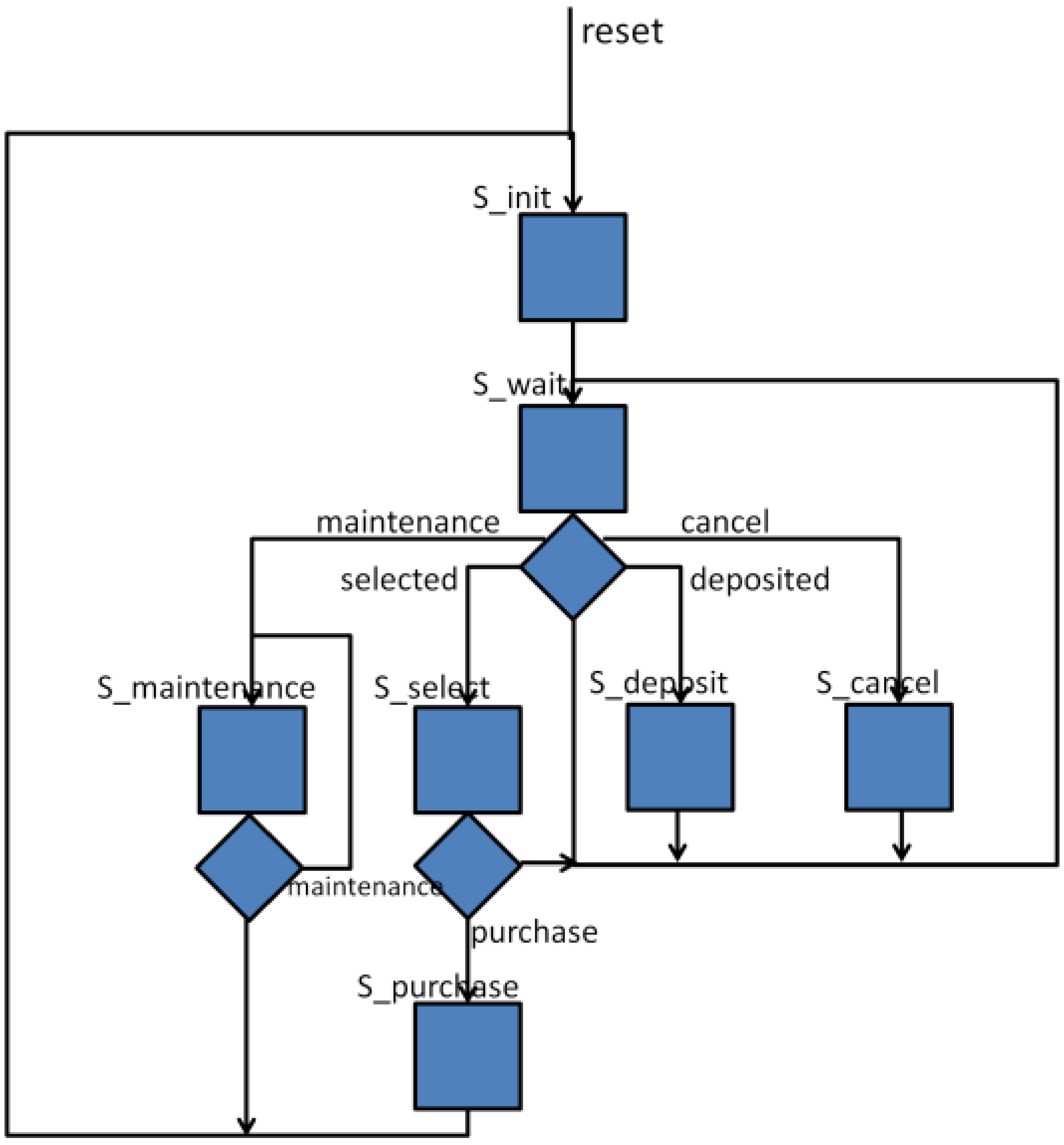
Product Dispenser:

Product: represent the selected product number to dispense

2.2 INPUT AND OUTPUT BLOCK DIAGRAM



2.3 ASM


reset
S_init
S_wait
maintenance
cancel
selected
deposited
S_maintenance
S_select
S_deposit
S_cancel
maintenance
purchase
S_purchase

2.4 RTL-CS

RTL Operations		**Control Vector**															
		ldMquantity	ldMprice	clrRpurchase	clrRchange	clrRproduct	clrA	clrRprice	clrRselect	clrRdeposit	ldRpurchase	ldRchange	ldRproduct	ldA	ldRprice	ldRselect	ldRdeposit
S_init	Go to S_wait	16'b0011_1111_1000_0000															
S_wait	(maintenance)/ Go to S_maintenance	16'b0011_1000_0000_0111															
	(deposited)/ Go to S_deposit	16'b0011_1000_0000_0111															
	(cancel)/ Go to S_cancel	16'b0011_1000_0000_0111															
	(selected)/ Go to S_select	16'b0011_1000_0000_0111															
	Go to S_wait	16'b0011_1000_0000_0111															
S_maintenance	(maintenance)/ Go to S_maintenance;	16'b0111_1100_1000_0110															
	Go to S_init;	16'b0011_1100_1000_0000															
S_select	(purchase) / Go to S_purchase	16'b0001_1010_1100_0000															
	Go to S_wait	16'b0001_1010_1100_0000															
S_deposit	Go to S_init;	16'b0011_1011_0000_1001															
S_cancel	Go to S_init;	16'b0011_1111_1000_0000															
S_purchase	Go to S_init;	16'b1010_0010_1011_0000															

3 IMPLEMENTATION

3.1 DATA PATH UNIT

```
///////////////////////////////////////////////////////////////////////-
// Design unit: Data Path Unit (Module)
//            :
// File name  : du.v
//            :
// Description: Data Path Unit for Vending Machine
//            :
// Limitations: None
//            :
// System     : Verilog
//            :
// Author     : 1. Wan Ahmad Zainie bin Wan Mohamad (ME131135)
//            :    wanahmadzainie@gmail.com
//            : 2. Azfar 'Aizat bin Mohd Isa (ME131032)
//            :    aaizat5@gmail.com
//
// Revision   : Version 0.1 2014-05-30 Initial
//            : Version 1.0 2014-06-09 Ready for submission
//            : Version 2.0 2014-06-10 Change to behavioral
///////////////////////////////////////////////////////////////////////-

module du(clk, rst, deposit, select, price, ldRdeposit, ldRselect, ldRprice,
                    ldA, ldRproduct, ldRchange, ldRpurchase, ldMprice, ldMquantity,
                    clrRdeposit, clrRselect, clrRprice, clrA, clrRproduct,
clrRchange,
                    clrRpurchase, purchase, refund, product, change);
       input  clk, rst;
       input  [9:0] deposit, price;
       input  [4:0] select;
       input  ldRdeposit, ldRselect, ldRprice, ldA, ldRproduct, ldRchange;
       input  ldRpurchase, ldMprice, ldMquantity, clrRdeposit, clrRselect;
       input  clrRprice, clrA, clrRproduct, clrRchange, clrRpurchase;
       output reg   purchase, refund;
       output reg   [4:0] product;
       output reg   [9:0] change;
       reg    [9:0] Rdeposit, Rprice, Adeposit;
       reg    [4:0] Rselect;
       reg    [15:0] mem [0:31];
       integer      i;

       initial begin
              for (i=0;i<32;i=i+1) begin
                     mem[i] = 16'h2864;
              end
              mem[0] = 16'b0000_0000_0011_0010; // quantity=0, price=50(RM5)
              mem[1] = 16'b0010_1001_1001_0000; // quantity=10, price=400(RM40)
       end
       //initial begin $readmemh("default.dat", mem); end

       // Register deposit
       always @ (negedge rst or posedge clk) begin
              if (rst == 0)              Rdeposit <= 0;
              else if (ldRdeposit)       Rdeposit <= deposit;
              else if (clrRdeposit)      Rdeposit <= 0;
       end

       // Register select
       always @ (negedge rst or posedge clk) begin
              if (rst == 0)              Rselect <= 0;
              else if (ldRselect)        Rselect <= select;
              else if (clrRselect)       Rselect <= 0;
       end

       // Register price
```

```
	always @ (negedge rst or posedge clk) begin
		if (rst == 0)		Rprice <= 0;
		else if (ldRprice)	Rprice <= price;
		else if (clrRprice)	Rprice <= 0;
	end

	// Accumulator accumulate deposit, and restore previous if exceed threshold
	always @ (negedge rst or posedge clk) begin
		if (rst == 0)	Adeposit <= 0;
		else if (ldA)	Adeposit <= Adeposit + Rdeposit;
		else if (clrA)		Adeposit <= 0;
		else if (refund)	Adeposit <= Adeposit - Rdeposit;
	end

	// Comparator Adeposit > maximum accepted deposit
	always @ (Adeposit) begin
		if (Adeposit > 500)	refund = 1;
		else				refund = 0;
	end

	// Comparator Adeposit >= price, quantity > 0
	always @ (Adeposit) begin
		for (i=0; i<32;i=i+1) begin
			if (0 < mem[i][13:10] && Adeposit >= mem[i][9:0])
					mem[i][15] = 1;
			else	mem[i][15] = 0;
		end
	end

	// Logic to indicate purchase
	always @ (negedge rst or posedge clk) begin
		if (rst == 0)				purchase <= 0;
		else if (ldRpurchase)
			if (mem[Rselect][15])	purchase <= 1;
			else					purchase <= 0;
		else if (clrRpurchase)		purchase <= 0;
	end

	// Substractor calculate change
	always @ (negedge rst or posedge clk) begin
		if (rst == 0)			change <= 0;
		else if (ldRchange)		change <= Adeposit - mem[Rselect][9:0];
		else if (clrRchange)	change <= 0;
	end

	// Register selected product
	always @ (negedge rst or posedge clk) begin
		if (rst == 0)			product <= 0;
		else if (ldRproduct)	product <= Rselect;
		else if (clrRproduct)	product <= 0;
	end

	// Register array update price or reduce quantity by 1
	always @ (posedge clk) begin
		if (ldMquantity)	mem[Rselect][13:10] <= mem[Rselect][13:10] - 1'b1;
		if (ldMprice)		mem[Rselect][9:0]	<= Rprice;
	end
endmodule
```

3.2 CONTROL UNIT

```
////////////////////////////////////////////////////////////////////-
// Design unit: Control Unit (Module)
//            :
// File name  : cu.v
//            :
// Description: Control Unit of Vending Machine
//            :
// Limitations: None
//            :
// System     : Verilog
//            :
// Author     : 1. Wan Ahmad Zainie bin Wan Mohamad (ME131135)
//            :    wanahmadzainie@gmail.com
//            : 2. Azfar 'Aizat bin Mohd Isa (ME131032)
//            :    aaizat5@gmail.com
//
// Revision   : Version 0.1 2014-05-30 Initial
//            : Version 1.0 2014-06-09 Ready for submission
//            : Version 2.0 2014-06-10 Change to behavioral
////////////////////////////////////////////////////////////////////-

module cu(clk, rst, deposited, selected, cancel, maintenance, purchase,
                    ldRdeposit, ldRselect, ldRprice, ldA, ldRproduct, ldRchange,
                    ldRpurchase, ldMprice, ldMquantity, clrRdeposit, clrRselect,
                    clrRprice, clrA, clrRproduct, clrRchange, clrRpurchase,
                    refundall, depositall, state);
       input  clk, rst;
       input  deposited, selected, cancel, maintenance, purchase;
       output ldRdeposit, ldRselect, ldRprice, ldA, ldRproduct, ldRchange;
       output ldRpurchase, ldMprice, ldMquantity, clrRdeposit, clrRselect;
       output clrRprice, clrA, clrRproduct, clrRchange, clrRpurchase;
       output refundall, depositall;
       output [2:0] state;
       reg          [2:0] pstate, nstate;
       reg          [15:0] cv;
       parameter    S_init = 3'b000, S_wait = 3'b001, S_deposit = 3'b010,
                          S_cancel = 3'b011, S_select = 3'b100, S_purchase =
3'b101,
                          S_maintenance = 3'b110;

       // state register submodule
       always @ (negedge clk or negedge rst) begin
              if (rst == 0) pstate <= S_init;
              else                pstate <= nstate;
       end

       // next state logic
       always @ (pstate or cancel or maintenance or deposited or selected or
purchase) begin
              case (pstate)
                     S_init:
                            nstate = S_wait;
                     S_wait:
                            if (maintenance)    nstate = S_maintenance;
                            else if (deposited) nstate = S_deposit;
                            else if (cancel)    nstate = S_cancel;
                            else if (selected)  nstate = S_select;
                            else                       nstate = S_wait;
                     S_select:
                            if (purchase)       nstate = S_purchase;
                            else                       nstate = S_wait;
                     S_purchase:                       nstate = S_init;
                     S_maintenance:
                            if (maintenance)    nstate = S_maintenance;
                            else                       nstate = S_init;
                     default:                          nstate = S_wait;
```

```
			endcase
		end

		// output logic
		always @ (pstate or selected) begin
			case (pstate)
				S_init:					cv = 16'b0011_1111_1000_0000;
				S_wait:					cv = 16'b0011_1000_0000_0111;
				S_deposit:			cv = 16'b0011_1011_0000_1001;
				S_cancel:			cv = 16'b0011_1111_1000_0000;
				S_select:			cv = 16'b0001_1010_1100_0000;
				S_purchase:			cv = 16'b1010_0010_1011_0000;
				S_maintenance:
					if (selected) cv = 16'b0111_1100_1000_0110;
					else			cv = 16'b0011_1100_1000_0000;
			endcase
		end

		assign state = pstate;
		assign ldRdeposit   = cv[0];
		assign ldRselect    = cv[1];
		assign ldRprice		= cv[2];
		assign ldA			= cv[3];
		assign ldRproduct   = cv[4];
		assign ldRchange    = cv[5];
		assign ldRpurchase  = cv[6];
		assign clrRdeposit  = cv[7];
		assign clrRselect   = cv[8];
		assign clrRprice    = cv[9];
		assign clrA			= cv[10];
		assign clrRproduct  = cv[11];
		assign clrRchange   = cv[12];
		assign clrRpurchase = cv[13];
		assign ldMprice		= cv[14];
		assign ldMquantity  = cv[15];
		assign refundall    = (state == S_cancel || state == S_maintenance) ? 1'b1 :
1'b0;
		assign depositall   = (state == S_purchase)   ? 1'b1 : 1'b0;
endmodule
```

3.3 INTEGRATION BETWEEN DATAPATH UNIT WITH CONTROL UNIT

```
/////////////////////////////////////////////////////////////////////-
// Design unit: vm (All In One)
//            :
// File name  : vm.v
//            :
// Description: RTL Design of Vending Machine
//            :
// Limitations: None
//            :
// System     : Verilog
//            :
// Author     : 1. Wan Ahmad Zainie bin Wan Mohamad (ME131135)
//            :    wanahmadzainie@gmail.com
//            : 2. Azfar 'Aizat bin Mohd Isa (ME131032)
//            :    aaizat5@gmail.com
//
// Revision   : Version 0.1 2014-06-01
//            : Version 1.0 2014-06-09 Ready for submission
//            : Version 2.0 2014-06-10 Change to behavioral
/////////////////////////////////////////////////////////////////////-

module vm(clk, rst, deposit, deposited, select, selected, price, cancel,
maintenance,
                    refund, refundall, depositall, product, change, state);
      input  clk, rst;
      input  [9:0] deposit, price;
      input  [4:0] select;
      input  deposited, selected, cancel, maintenance;
      output refund, refundall, depositall;
      output [4:0] product;
      output [9:0] change;
      output [2:0] state;
      wire   ldRdeposit, ldRselect, ldRprice, ldA, ldRproduct, ldRchange;
      wire   ldRpurchase, ldMprice, ldMquantity, clrRdeposit, clrRselect;
      wire   clrRprice, clrA, clrRproduct, clrRchange, clrRpurchase, purchase;

      du myDU(clk, rst, deposit, select, price, ldRdeposit, ldRselect, ldRprice,
                    ldA, ldRproduct, ldRchange, ldRpurchase, ldMprice, ldMquantity,
                    clrRdeposit, clrRselect, clrRprice, clrA, clrRproduct,
clrRchange,
                    clrRpurchase, purchase, refund, product, change);

      cu myCU(clk, rst, deposited, selected, cancel, maintenance, purchase,
                    ldRdeposit, ldRselect, ldRprice, ldA, ldRproduct, ldRchange,
                    ldRpurchase, ldMprice, ldMquantity, clrRdeposit, clrRselect,
                    clrRprice, clrA, clrRproduct, clrRchange, clrRpurchase,
                    refundall, depositall, state);
endmodule
```

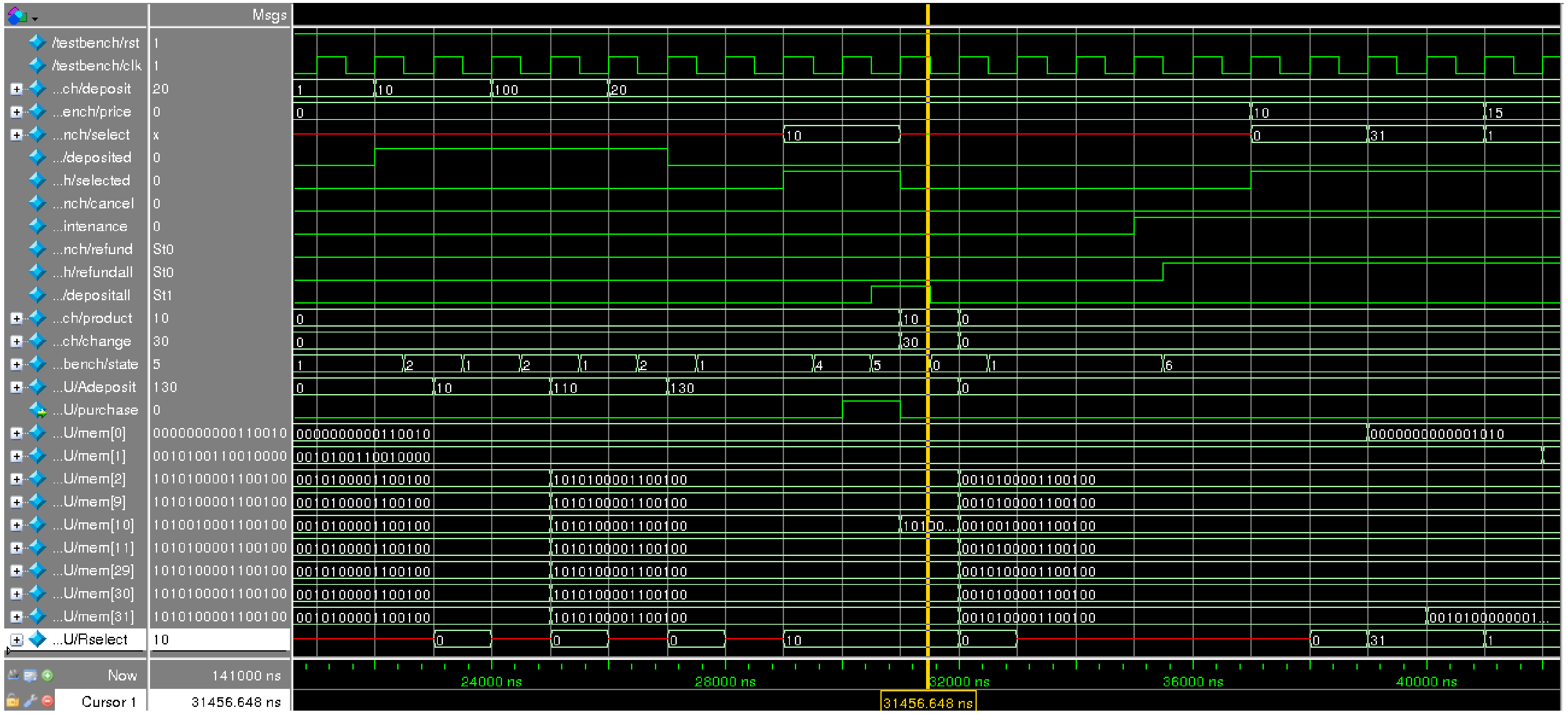
3.4 TESTBENCH

```
/////////////////////////////////////////////////////////////////////-
// Design unit: testbench (Module)
//            :
// File name  : testbench.v
//            :
// Description: Test Bench for RTL Vending Machine
//            :
// Limitations: None
//            :
// System     : Verilog
//            :
// Author     : 1. Wan Ahmad Zainie bin Wan Mohamad (ME131135)
//            :    wanahmadzainie@gmail.com
//            : 2. Azfar 'Aizat bin Mohd Isa (ME131032)
//            :    aaizat5@gmail.com
//
// Revision   : Version 0.1 2014-06-01
//            : Version 1.0 2014-06-09 Ready for submission
//            : Version 2.0 2014-06-10 Change to behavioral - more simulation
/////////////////////////////////////////////////////////////////////-

`timescale 100ns / 1ns

module testbench();
       // inputs
       reg    clk, rst;
       reg    [9:0] deposit, price;
       reg    [4:0] select;
       reg    deposited, selected, cancel, maintenance;

       // outputs
       wire   refund, refundall, depositall;
       wire   [4:0] product;
       wire   [9:0] change;
       wire   [2:0] state;

       // instantiation
       vm myVM(
              .clk(clk),
              .rst(rst),
              .deposit(deposit),
              .deposited(deposited),
              .select(select),
              .selected(selected),
              .price(price),
              .cancel(cancel),
              .maintenance(maintenance),
              .refund(refund),
              .refundall(refundall),
              .depositall(depositall),
              .product(product),
              .change(change),
              .state(state)
       );

// for future use
//     initial begin
//            $display("time: rst, clk");
//            $monitor(" %0d:  %b,  %b", $time, rst, clk);
//     end
```

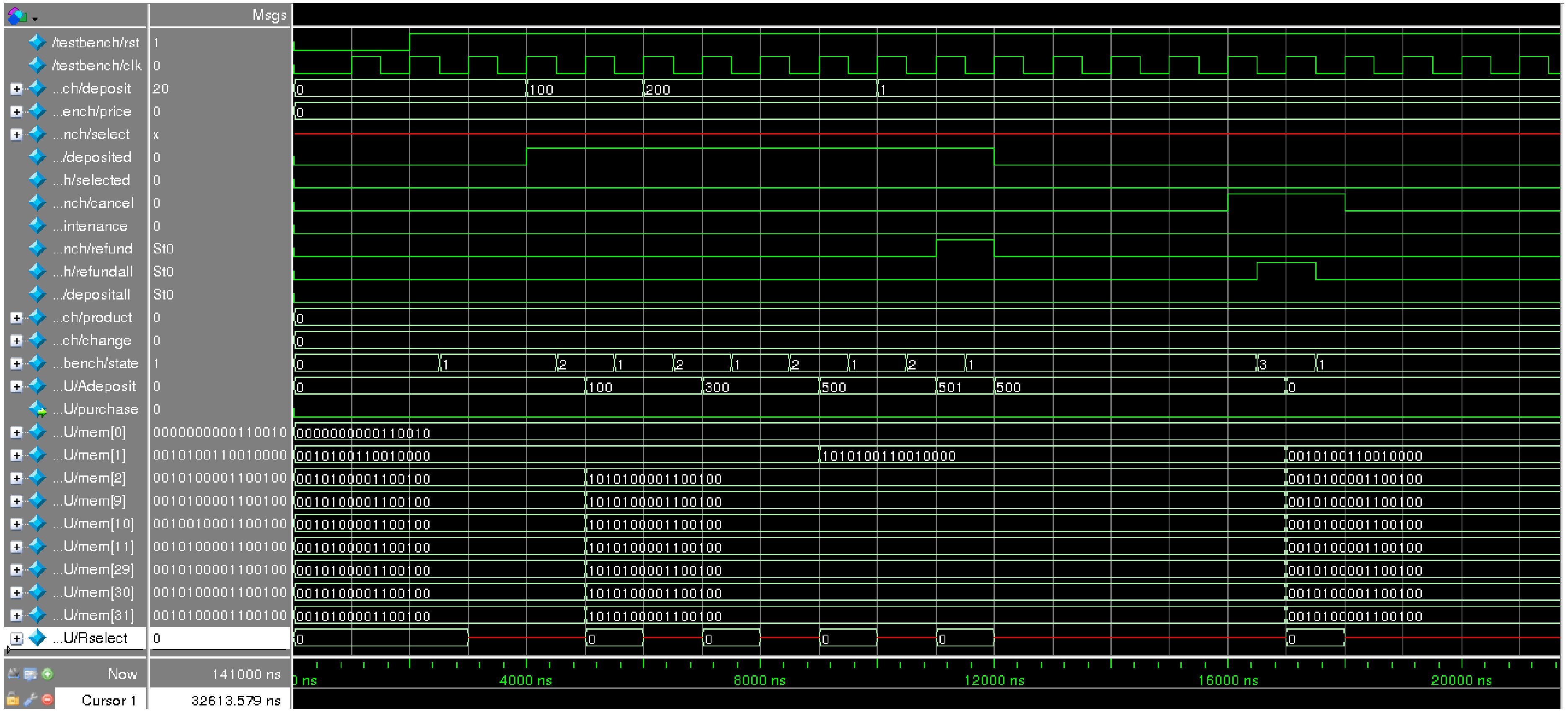
```
       initial begin
              clk= 0;
              #5;
              forever #5 clk = ~clk;
       end

       initial begin
              rst= 0;
              deposited= 0; deposit= 0; selected= 0; select= 'bx; price= 0;
              cancel= 0; maintenance= 0;
              #10;
              #10           rst= 1;
// simulate overflow, insert payment more than threshold value
              #20           deposited= 1; deposit= 100;
              #20           deposited= 1; deposit= 200;
              #20           deposited= 1; deposit= 200;
              #20           deposited= 1; deposit=   1;
              #20           deposited= 0;
              #20;
// simulate cancel
              #20           cancel= 1;
              #20           cancel= 0;
              #20;
// simulate purchase
              #20           deposited= 1; deposit=  10;
              #20           deposited= 1; deposit= 100;
              #20           deposited= 1; deposit=  20;
              #10           deposited= 0;
              #20           selected= 1; select= 10;
              #20           selected= 0; select= 'bx;
              #20;
// simulate maintenance changing price
              #20           maintenance= 1;
              #20           selected= 1; select =  0; price=  10;
              #20           selected= 1; select = 31; price=  10;
              #20           selected= 1; select =  1; price=  15;
              #20           selected= 1; select =  2; price=  10;
              #20           selected= 1; select =  3; price=  20;
              #20           selected= 1; select =  4; price=  10;
              #20           selected= 1; select =  5; price=  15;
              #20           selected= 1; select =  6; price=  15;
              #20           selected= 1; select =  7; price=  15;
              #20           selected= 1; select =  8; price=  10;
              #20           selected= 1; select =  9; price=  10;
              #20           selected= 1; select = 10; price=  10;
              #20           selected= 1; select = 11; price=  10;
              #20           selected= 1; select = 12; price=  10;
              #20           selected= 1; select = 13; price=  10;
              #20           selected= 1; select = 14; price=  10;
              #20           selected= 1; select = 15; price=  10;
              #20           selected= 1; select = 16; price=  10;
              #20           selected= 1; select = 17; price=  10;
              #20           selected= 1; select = 18; price=  10;
              #20           selected= 1; select = 19; price=  10;
              #20           selected= 1; select = 20; price=  10;
              #20           selected= 1; select = 21; price=  10;
              #20           selected= 1; select = 22; price=  10;
              #20           selected= 1; select = 23; price=  10;
              #20           selected= 1; select = 24; price=  10;
              #20           selected= 1; select = 25; price=  10;
              #20           selected= 1; select = 26; price=  10;
              #20           selected= 1; select = 27; price=  10;
              #20           selected= 1; select = 28; price=  20;
              #20           selected= 1; select = 29; price=  20;
              #20           selected= 1; select = 30; price=  20;
              #20           selected= 1; select = 31; price=  99;
              #20           selected= 0; select= 'bx; maintenance = 0;
              #20;
// simulate purchase after changing price, item 30, from RM10 to RM2 (balance RM28)
              #20           deposited= 1; deposit= 200;
              #20           deposited= 1; deposit= 500;
              #20           deposited= 1; deposit= 100;
              #20           deposited= 0;
              #20           selected= 1; select= 30;
              #20           selected= 0; select= 'bx;
              #20           selected= 1; select= 3;
              #20           selected= 0; select= 'bx;
              #200   $stop;
       end

endmodule
```

4 RESULT AND VERIFICATION

4.1 DEPOSIT AND PURCHASE

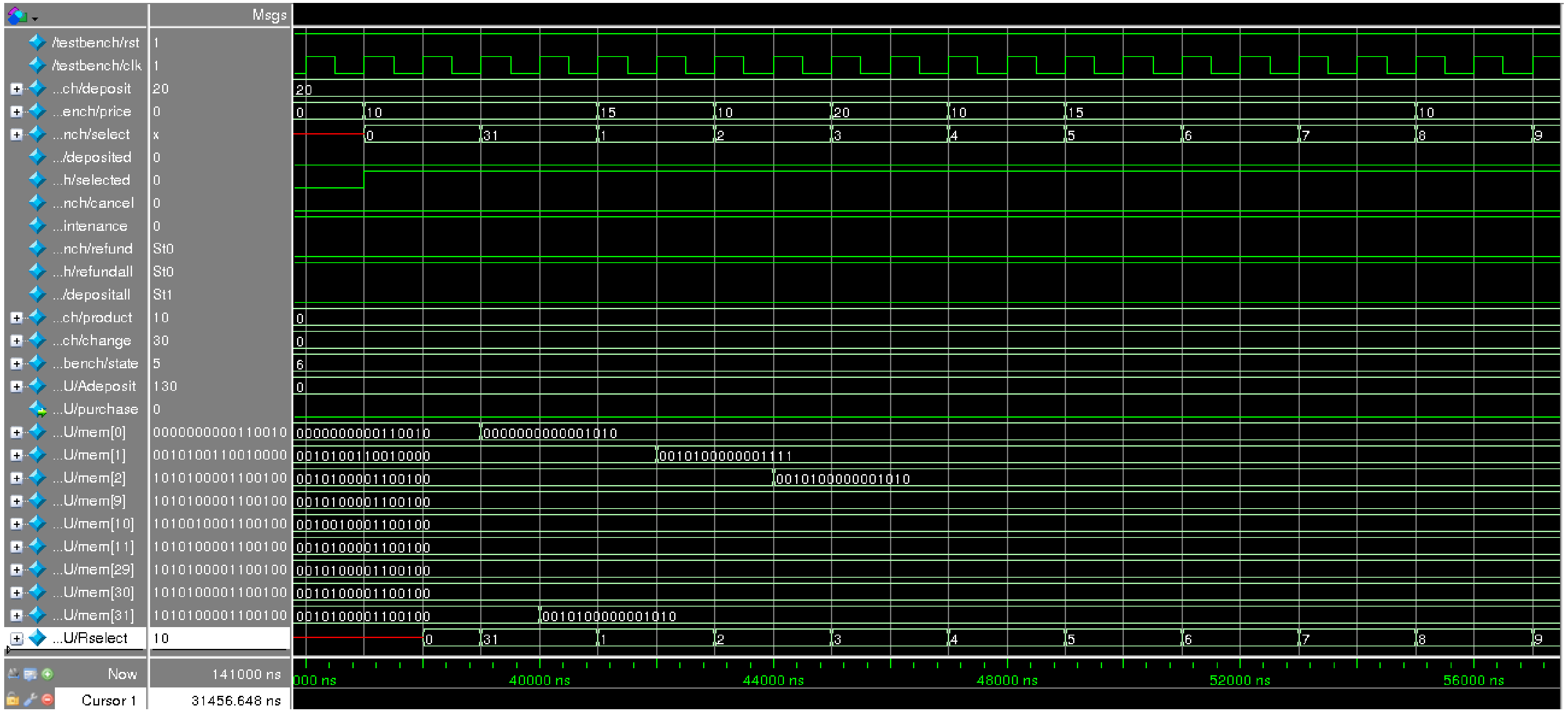


4.2 DEPOSIT OVERFLOW AND CANCEL



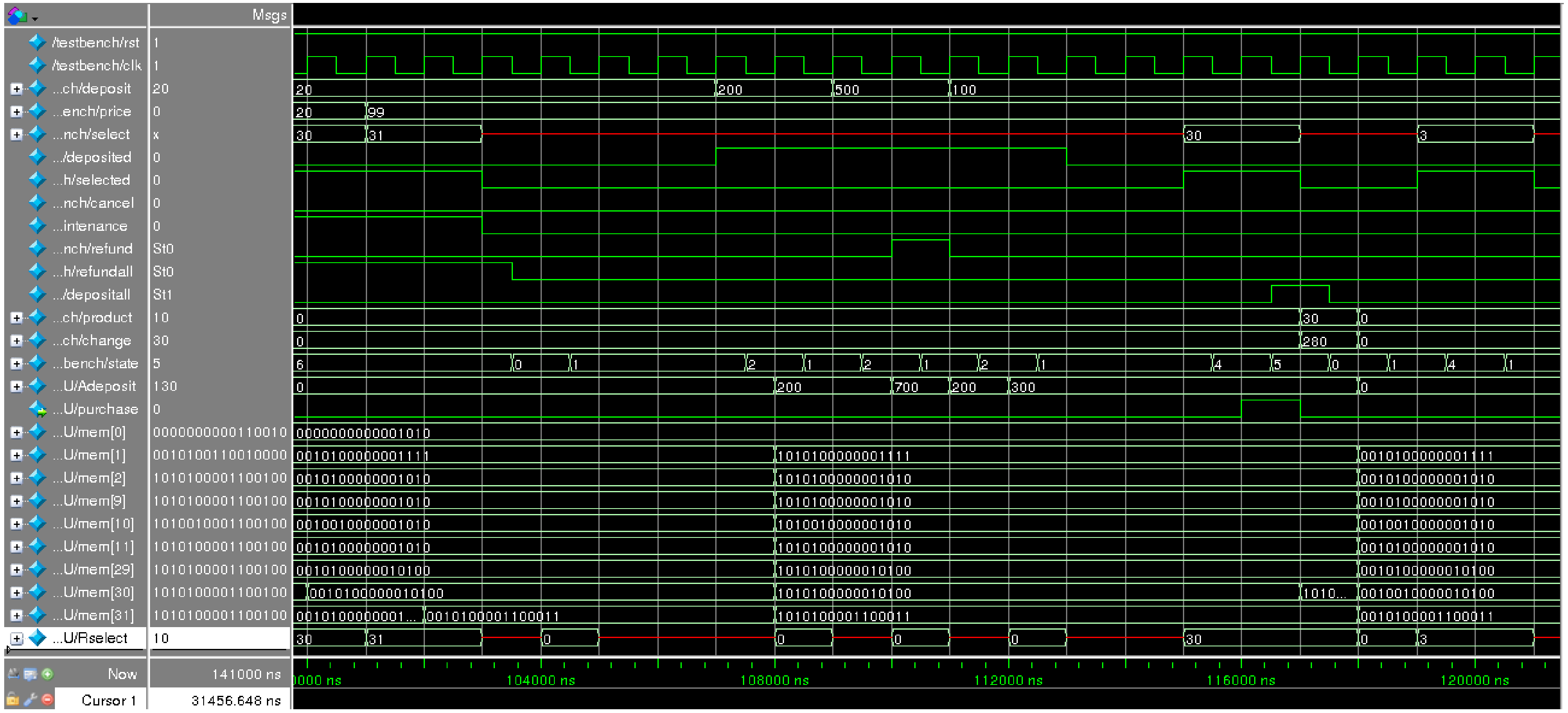
```
                      Msgs
/testbench/rst        1
/testbench/clk        0
...ch/deposit         20
...ench/price         0
...nch/select         x
.../deposited         0
...h/selected         0
...nch/cancel         0
...intenance          0
...nch/refund         St0
...h/refundall        St0
.../depositall        St0
...ch/product         0
...ch/change          0
...bench/state        1
...U/Adeposit         0
...U/purchase         0
...U/mem[0]           0000000000110010
...U/mem[1]           0010100110010000
...U/mem[2]           0010100001100100
...U/mem[9]           0010100001100100
...U/mem[10]          0010010001100100
...U/mem[11]          0010100001100100
...U/mem[29]          0010100001100100
...U/mem[30]          0010100001100100
...U/mem[31]          0010100001100100
...U/Rselect          0

Now                   141000 ns
Cursor 1              32613.579 ns
```

4.3 MAINTENANCE CHANGE PRICE



```
                      Msgs
/testbench/rst        1
/testbench/clk        1
...ch/deposit         20
...ench/price         0
...nch/select         x
.../deposited         0
...h/selected         0
...nch/cancel         0
...intenance          0
...nch/refund         St0
...h/refundall        St0
.../depositall        St1
...ch/product         10
...ch/change          30
...bench/state        5
...U/Adeposit         130
...U/purchase         0
...U/mem[0]           0000000000110010
...U/mem[1]           0010100110010000
...U/mem[2]           1010100001100100
...U/mem[9]           1010100001100100
...U/mem[10]          1010010001100100
...U/mem[11]          1010100001100100
...U/mem[29]          1010100001100100
...U/mem[30]          1010100001100100
...U/mem[31]          1010100001100100
...U/Rselect          10

Now                   141000 ns
Cursor 1              31456.648 ns
```

4.4 NEW PRICE DEPOSIT PURCHASE



	Msgs
/testbench/rst	1
/testbench/clk	1
...ch/deposit	20
...ench/price	0
...nch/select	x
.../deposited	0
...h/selected	0
...nch/cancel	0
...intenance	0
...nch/refund	St0
...h/refundall	St0
.../depositall	St1
...ch/product	10
...ch/change	30
...bench/state	5
...U/Adeposit	130
...U/purchase	0
...U/mem[0]	0000000000110010
...U/mem[1]	0010100110010000
...U/mem[2]	1010100001100100
...U/mem[9]	1010100001100100
...U/mem[10]	1010010001100100
...U/mem[11]	1010100001100100
...U/mem[29]	1010100001100100
...U/mem[30]	1010100001100100
...U/mem[31]	1010100001100100
...U/Rselect	10
Now	141000 ns
Cursor 1	31456.648 ns

4.5 FLOW SUMMARY

```
+------------------------------------------------------------------+
; Flow Summary                                                     ;
+---------------------------------+--------------------------------+
; Flow Status                     ; Successful - Wed Jun 11 13:10:05
2014      ;
; Quartus II 64-Bit Version       ; 13.1.0 Build 162 10/23/2013 SJ
Web Edition ;
; Revision Name                   ; vm
;
; Top-level Entity Name           ; vm
;
; Family                          ; Cyclone V
;
; Device                          ; 5CGXFC7C7F23C8
;
; Timing Models                   ; Final
;
; Logic utilization (in ALMs)     ; 425 / 56,480 ( < 1 % )
;
; Total registers                 ; 538
;
; Total pins                      ; 52 / 268 ( 19 % )
;
; Total virtual pins              ; 0
;
; Total block memory bits         ; 160 / 7,024,640 ( < 1 % )
;
; Total DSP Blocks                ; 0 / 156 ( 0 % )
;
; Total HSSI RX PCSs              ; 0 / 6 ( 0 % )
;
; Total HSSI PMA RX Deserializers ; 0 / 6 ( 0 % )
;
; Total HSSI TX PCSs              ; 0 / 6 ( 0 % )
;
; Total HSSI TX Channels          ; 0 / 6 ( 0 % )
;
; Total PLLs                      ; 0 / 13 ( 0 % )
;
; Total DLLs                      ; 0 / 4 ( 0 % )
;
+---------------------------------+--------------------------------+
```

4.6 RESOURCE USAGE SUMMARY

```
+-----------------------------------------------------------+
; Analysis & Synthesis Resource Usage Summary               ;
+-----------------------------------------------+-----------+
; Resource                                      ; Usage     ;
+-----------------------------------------------+-----------+
; Estimate of Logic utilization (ALMs needed)   ; 458       ;
;                                               ;           ;
; Combinational ALUT usage for logic            ; 629       ;
;     -- 7 input functions                      ; 10        ;
;     -- 6 input functions                      ; 241       ;
;     -- 5 input functions                      ; 73        ;
;     -- 4 input functions                      ; 104       ;
;     -- <=3 input functions                    ; 201       ;
;                                               ;           ;
; Dedicated logic registers                     ; 522       ;
;                                               ;           ;
; I/O pins                                      ; 52        ;
; Total MLAB memory bits                        ; 0         ;
; Total block memory bits                       ; 160       ;
; Total DSP Blocks                              ; 0         ;
; Maximum fan-out node                          ; clk~input ;
; Maximum fan-out                               ; 527       ;
; Total fan-out                                 ; 4332      ;
; Average fan-out                               ; 3.44      ;
+-----------------------------------------------+-----------+
```

4.8 RESOURCE UTILIZATION BY ENTITY

```
+----------------------------------------------------------------------------------------------------------------------
; Analysis & Synthesis Resource Utilization by Entity
+--------------------------------------------+--------------------+-------------+-------------------+------------
; Compilation Hierarchy Node                 ; LC Combinationals ; LC Registers ; Block Memory Bits ; DSP Blocks
+--------------------------------------------+--------------------+-------------+-------------------+------------
; |vm                                        ; 629 (0)            ; 522 (0)      ; 160               ; 0
;    |cu:myCU|                               ; 18 (18)            ; 7 (7)        ; 0                 ; 0
;    |du:myDU|                               ; 611 (611)          ; 515 (515)    ; 160               ; 0
;       |altsyncram:mem[0][12]__2|           ; 0 (0)              ; 0 (0)        ; 64                ; 0
;          |altsyncram_ovm1:auto_generated| ; 0 (0)              ; 0 (0)        ; 64                ; 0
;       |altsyncram:mem[0][9]__1|            ; 0 (0)              ; 0 (0)        ; 96                ; 0
;          |altsyncram_qvm1:auto_generated| ; 0 (0)              ; 0 (0)        ; 96                ; 0
+--------------------------------------------+--------------------+-------------+-------------------+------------
```

```
---------------------------------------------------------------------------------------------------------------+
                                                                                                              ;
+------+--------------+-------------------------------------------------------------------------+--------------+
; Pins ; Virtual Pins ; Full Hierarchy Name                                                     ; Library Name ;
+------+--------------+-------------------------------------------------------------------------+--------------+
; 52   ; 0            ; |vm                                                                     ; work         ;
; 0    ; 0            ; |vm|cu:myCU                                                             ; work         ;
; 0    ; 0            ; |vm|du:myDU                                                             ; work         ;
; 0    ; 0            ; |vm|du:myDU|altsyncram:mem[0][12]__2                                    ; work         ;
; 0    ; 0            ; |vm|du:myDU|altsyncram:mem[0][12]__2|altsyncram_ovm1:auto_generated ; work         ;
; 0    ; 0            ; |vm|du:myDU|altsyncram:mem[0][9]__1                                     ; work         ;
; 0    ; 0            ; |vm|du:myDU|altsyncram:mem[0][9]__1|altsyncram_qvm1:auto_generated  ; work         ;
+------+--------------+-------------------------------------------------------------------------+--------------+
```

```
Note: For table entries with two numbers listed, the numbers in parentheses indicate the number of resources of the
given type used by the specific entity alone. The numbers listed outside of parentheses indicate the total resources
of the given type used by the specific entity and all of its sub-entities in the hierarchy
```