

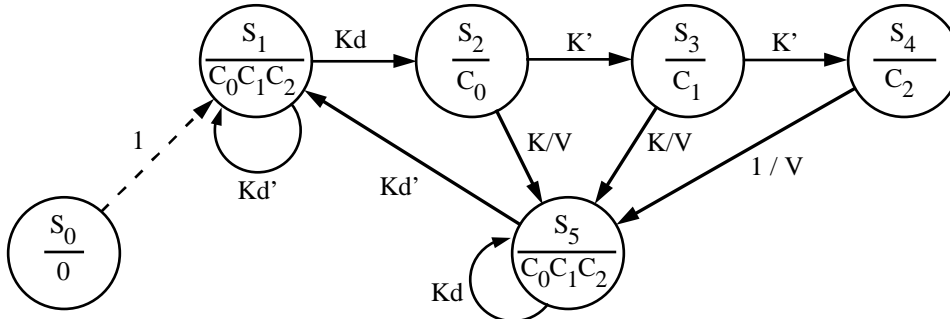
Corrections for *Digital Systems Design Using VHDL*, 3rd printing

Corrected VHDL code can be found on the web page:

<http://www.brookscole.com/engineering/ee/roth.html>

line -7 means 7th line from bottom, etc.

p. 112, Figure 3-25, replace with the following:



p. 226, line -15: interchange G and F

p. 259, Fig. 7-13, lower right, add label to xor gate output: S4

p. 262, line 12: are are should be are

p. 292, lines 5-7, replace "added Nextstate <= '0'; ... unwanted latch." with "changed the if statements so that Nextstate is always specified even if there is no change of state. For example, in the fifth line of the first process, we added **else** Nextstate <= 0; before **end if**, and we added Nextstate <= 1; at the end of the seventh line."

p. 311, line 24, add: next_state <= 3;

p. 324, line 7: now /= 0 ns should be state = T2

p. 324, line -7 should read: where dbus'last_event is the time since the

p. 324, line -10: when the chip is selected and now = 0 should be in state T2

p. 335, problem 9.2(b), last line: unknown should be the value just stored

p. 414, line -14: ADDR, should be ADDR0, and line -8: ADDR should be ADDR2

p. 441, line 20: OP=JMP should be (OP=JMP and reg_mem)

p. 441, line 22: OP=JSR should be (OP=JSR and reg_mem)

p. 447, line 1, add before end if: selMd2 <= '0';

p. 449, line -13: JMP='1' should be (JMP='1' and reg_mem)

p. 449, line -11: JSR='1' should be (JSR='1' and reg_mem)

p. 451, add after line 4: end process;

Corrections for *Digital Systems Design Using VHDL*, 1st and 2nd printing

Corrected VHDL code can be found on the web page:

<http://www.brookscole.com/engineering/ee/roth.html>

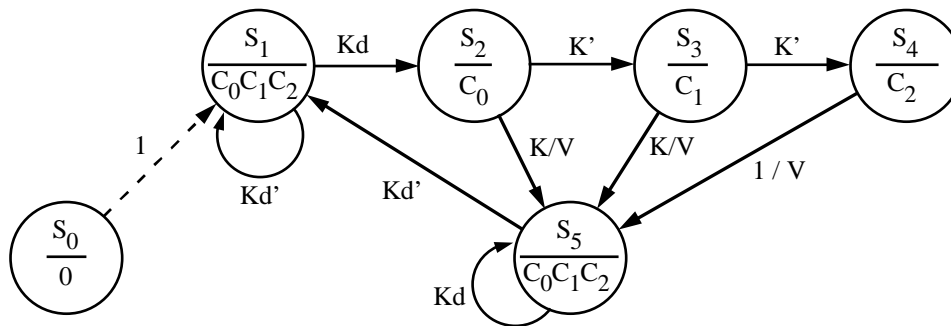
line -7 means 7th line from bottom, etc.

p. 9, line 18: "ABC' or AC'D. Thus one" should be "AB. Thus the"

p. 9, line 20: ABC' should be AB

p. 9, line 22: (A' + B' + C) should be (A' + B')

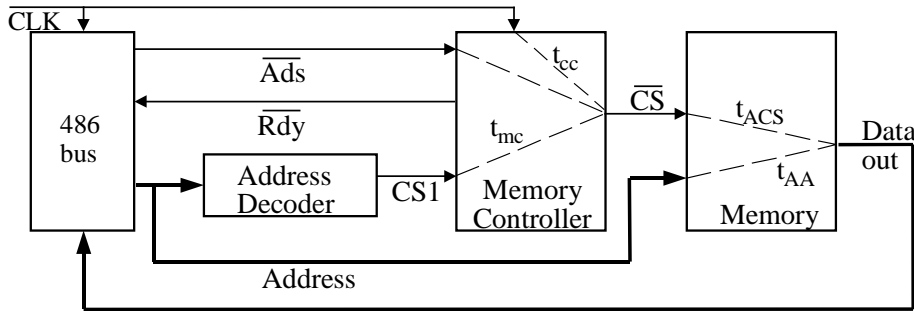
- p. 21, Figure 1-19, map for Z, within loop on right: change 0 to 1
- p. 26, line 18: $e \neq f$ should be $a \neq g$; line 19: $e-f$ should be $e-g$
- p. 41, top figure: insert > in front of CK
- p. 53, line 13: -- see Note 3 should be moved right to align with other comments
- p. 55, first line after Figure 2-9: *conditional assignment* should be *conditional signal assignment*
- p. 55, lines 11 & 12 should be: A behavioral model of the 4-to-1 MUX of Figure 2-9 using a *conditional signal assignment statement* is
- p. 67, line -6: vlaues should be values
- p. 81, Problem 2.4, line -7: $A \leq '1'$ should be $A = '1'$
- p. 81, Problem 2.4, line -4: $< =$ should be $<=$
- p. 82, Problem 2.9(b): Z should be D and B
- p. 82, Prob. 2.10(a): Problem 1.12 should be Problem 1.10;
- p. 82, Prob. 2-10(b): Z is should be S and V are
- p. 89, Figure 3-3, line 6: omit blank line above **end ROM1_2**;
- p. 105, Figure 3-19, line 3: **out** should be **inout**
- p. 106, line 9: 30 should be 20
- p. 110, Figure 3-22: add horizontal line at top of figure
- p. 112, Figure 3-25, replace with the following:



- p. 115, 10th line above Figure 3-27, near end of line, q3 should be Q3.
- p. 120, Problem 3-10(c): an SM chart should be a state graph
- p. 133, line -5: move 1.110001 right to align decimal point with line above
- p. 136, Figure 4-11: add a self-loop labeled $S_t/0$ to state S_0 ; line -2: use should be in bold face type
- p. 137, line -10: move "--output product" to end of previous line
- p. 140, line -2: delete "part of the"
- p. 140, line -1: replace "After" with "In the second process, after"
- p. 141-142, Figure 4-16: Replace with attached copy. (Although the code in the book is correct, it is not consistent with the test bench of Figure 4-14. It is better to use a two-process model with one process for the combinational logic and one for the register updates.)
- p. 143, Figure 4-18, add before begin: **alias** M: bit is B(0);
- p. 145, first row of boxes below figure: add a 1 in a box at the right end
- p. 145, at bottom of page: move dashed line one place to the right
- p. 146, line 4: move 1 1 0 1 one place to the left
- p. 150, Figure 4-22, two occurrences of: Co1 should be Cm1
- p. 154, line -3: multiplier should be divider

- p. 157, problem 4.8(a): delete last sentence. Add: Use a serial adder to add the multiplicand to the accumulator.
- p. 158, Problem 4-13(b): 3 states should be 5 states; 4-13(c): 4-5 should be 4-12
- p. 158, Problem 4.13 (d): 4-12 should be 4-16
- p. 164, Figure 5-4(b): add horizontal line at top of figure
- p. 165, line -7: 10 should be 11
- p. 179, line -2: A'BM' should be A'BM
- p. 194, Problem 5.3: add 0 to right of X_3 diamond, add 1 below X_3 diamond
leftmost output box: Z_1 should be Z_3
- p. 197, line 2: both NST and NSF should be NST
- p. 220, Figure 6-20, right side: interchange XQ and YQ; interchange X and Y
- p. 221, Figure 6-21, add horizontal line from right edge of box H to nearest vertical line.
- p. 225, last two lines: interchange G and F
- p. 230, line -15: Figure 5-32(a) should be Figure 5-33(a)
- p. 251, line 2: move multiply to end of previous line
- p. 259, Fig. 7-13, lower right, add label to xor gate output: S4
- p. 262, Problem 7.4, add: Fractions are 8 bits including sign, and exponents are 5 bits including sign.
- p. 262, Problem 7.5, add: Fractions are 5 bits including sign, and exponents are 4 bits including sign
- p. 267, Table 8-3, Returns column, line 10: move "reversed" down below "Nth index range"
- p. 268, line 2: if has should be if it has
- p. 283, line -9; **for** should be **until**
- p. 291, line -7: $Z \leq '0'$ should be $Z \leq '1'$
- p. 292, lines 5-7, replace "added Nextstate $\leq '0'$; ... unwanted latch." with "changed the if statements so that Nextstate is always specified even if there is no change of state. For example, in the fifth line of the first process, we added **else** Nextstate ≤ 0 ; before **end if**, and we added Nextstate ≤ 1 ; at the end of the seventh line."
- p. 293, Figure 8-20: delete first two lines
- p. 296, line -10: (buff, test_data) should be (test_data, buff)
- p. 299, Figure 8-21, line 7: test should be testfill; line 8: 2047 should be 8191
- p. 299, Figure 8-21, line 17: is open read_mode should be **open read_mode is**
- p. 299, lines 23 and -10: addr should be addr1
- p. 301, Problem 8.6, for signal B: delete 'X' **after** 8 ns, ; also delete '1' **after** 12 ns,
- p. 301, Problem 8.6(b): replace ", and draw a timing chart for each" with ". Assume that C drives an open-collector bus with a pull-up resistor (see Problem 8.5)."
- p. 308, Figure 9-6, last waveform: old should be any
- p. 311, line 24, add: next_state ≤ 3 ;
- p. 313, insert after line 20: **if** CS_b = '0' **then** insert after line 23: **end if**;
- p. 314, line 3: delete CS_b'delayed = '0' **and**
- p. 314, add after line 8: -- The following code only checks for a WE_b controlled write:
- p. 314, line 9: insert **and** CS_b'delayed='0' before **then**
- p. 314, line -4 and p. 315, line 1: std_logic should be bit

- p. 315, line 10: replace "write(2) with CS pulse" with "WE-controlled write"
- p. 324, line 7: now /= 0 ns should be state = T2
- p. 324, line 12: delete now –
- p. 324, line –10: when the chip is selected and now = 0 should be in state T2
- p. 324, line –8: (now – dbus'event) should be dbus'last_event
- p. 324, line –7 should read: where dbus'last_event is the time since the
- p. 326, replace Figure 9-23 with the following:



- p. 326, line –4 should read: Delay from CLK high to \overline{CS} low = $t_{cc} = 5$ ns
- p. 327, lines –1, –4, –14: t_{mc} should be t_{cc}
- p. 328, Figure 9-25: t_{6max} should be t_{10max} ; t_{mc} should be t_{cc}
- p. 329, line 10: delete new_we_b, line 14: delete new_we_b <= '1';
- p. 331, line 1, end of line 1: delete |
- p. 335, Problem 9.1(c): delete "and latest"
- p. 335, replace Problem 9.1(d) with: For a write cycle, what is the minimum time that valid data must be driven onto the data bus?
- p. 335, problem 9.2(b), last line: unknown should be the value just stored
- p. 335, Problem 9.3(a): replace "Add code that" with "Verify that the code"
- p. 335, Problem 9.5: Figure 9-5 should be Figure 9-6
- p. 338, lower figure: add a vertical line from the small circle down to the RW line
- p. 344, lines –8 and –9: k should be k+1
- p. 346, line –7: S3 should be S1
- p. 349, line 6: shift 010 110 011 111 left to align with column headings
- p. 354: replace Figure 10-16 with attached figure
- p. 355, top flip-flop in IC1: D1 Q1 should be D0 Q0
- p. 357, line 9 of item 6: Update-IR should be Update-DR
- p. 363, Table 10-4, for n = 5: delete Q1 \oplus
- p. 363, line above Figure 10-24, add: as for Figure 10-23
- p. 364, 3rd and 6th lines after Figure 10-25: 1010 should be 0010; 5th line: 1000 should be 0000
- p. 367, line 2: 10 should be 01
- p. 369, line 16, add after bit_vector: (3 **downto** 0)
- p. 376, line above Figure 11-5: add semi-colon at end.
- p. 377, line 3: 7 should be 8; line –7: TSRout should be TSR(0); line 12: **of** should not be bold

- p. 384, lines 3 and -9: CONVERT_INT should be CONV_INTEGER
- p. 385, table above Figure 11-12: complement all entries in R_W column
- p. 385, Figure 11-12, line 4: add rst_b, after clk,
- p. 386, line 4: BclkX8, should be BclkX8: **buffer** std_logic;
- p. 386, lines 14, 16, and 18: map should be in bold face type
- p. 387, lines 1, 2, 3, 5, 6, and 7: ADDR should be ADDR2
- p. 391, Table 11-2(b), line 8: LSR should be LSL
- p. 395, Table 11-4, 1st column: INC dir should be INC dir_m, INC ix should be INC ix_m,
INC ix1 should be INC ix_{m1}
- p. 396, "JSR ext" row, 4th column: $P \leftarrow SP - 1$ should be $SP \leftarrow SP - 1$
- p. 396, "SWI" row, 3rd column: $SP \vee SP - 1$ should be $SP \leftarrow SP - 1$
- p. 397, line -7: \leq should be \leftarrow
- p. 398, line -3 of Table 11-5: 42 should be 57
- p. 398, last line of Table 11-5: 0305 should be 0304 (two places)
- p. 403, line 11: **asssert** should be **assert**
- p. 410, line 2: that should be than
- p. 414, line -17: insert cs, before cpu_wr; line -4: cs, should be cs1, cs2,
- p. 414, line -14: ADDR, should be ADDR0, and line -8: ADDR should be ADDR2
- p. 414, Figure 11-24, lines -1 and -2: others should be in bold face type type
- p. 415, lines 2, 3, 4, 5, 7, and 8: map should be in bold face type
- p. 415, 1st and 5th lines below the VHDL code: 4020 should be 4010
- p. 428, line -14: delete ;
- p. 441, line 20: OP=JMP should be (OP=JMP and reg_mem)
- p. 441, line 22: OP=JSR should be (OP=JSR and reg_mem)
- p. 443, delete lines 4 and 5
- p. 444, insert after line 11: **alias** JMP : bit is opd(12); **alias** JSR : bit is opd(13);
- p. 446, line 18: CONV_INEGER(TO_INTEGER should be conv_integer(TO_stdlogicvector
- p. 447, line 1, add before end if: selMd2 \leq '0';
- p. 449, line -13: JMP='1' should be (JMP='1' and reg_mem)
- p. 449, line -11: JSR='1' should be (JSR='1' and reg_mem)
- p. 451, add after line 4: end process;
- p. 456, last line of table: 32 should be 132

Please report any additional errors to: roth@ece.utexas.edu

Figure 4-16 Model for 2's Complement Multiplier with Control Signals

```

-- This architecture of a 4-bit multiplier for 2's complement numbers
-- uses control signals.

architecture behave2 of mult2Cs is
    signal State, Nextstate: integer range 0 to 5;
    signal A, B: bit_vector(3 downto 0);
    signal AdSh, Sh, Load, Cm: bit;
    signal addout: bit_vector(4 downto 0);
    alias M: bit is B(0);
begin
    process (state, st, M)
    begin
        Load <= '0'; AdSh <= '0'; Sh <= '0'; Cm <= '0'; Done <= '0';
        case State is
            when 0=> -- initial State
                if St='1' then Load <= '1'; Nextstate <= 1; end if;
            when 1 | 2 | 3 => -- "add/shift" State
                if M = '1' then AdSh <= '1';
                    else Sh <= '1';
                end if;
                Nextstate <= State + 1;
            when 4 => -- add complement if sign
                -- bit of multiplier is 1
                if M = '1' then
                    Cm <= '1'; AdSh <= '1';
                else Sh <= '1';
                end if;
                nextstate <= 5;
            when 5 => -- output product
                done <= '1';
                nextstate <= 0;
        end case;
    end process;

    addout <= add4(A, Mcand, '0') when Cm = '0'
        else add4(A, not Mcand, '1');

    process
    begin
        wait until CLK = '1'; -- executes on rising edge
        if Load = '1' then -- load the multiplier
            A <= "0000";
            B <= Mplier;
        end if;
        if AdSh = '1' then -- Add multiplicand to A and shift
            A <= (Mcand(3) xor Cm) & addout(3 downto 1);
            B <= addout(0) & B(3 downto 1);
        end if;
        if Sh = '1' then
            A <= A(3) & A(3 downto 1);
            B <= A(0) & B(3 downto 1);
        end if;
        State <= Nextstate;
    end process;

    Product <= A(2 downto 0) & B;
end behave2;

```

Figure 10-16. State machine for TAP controller

