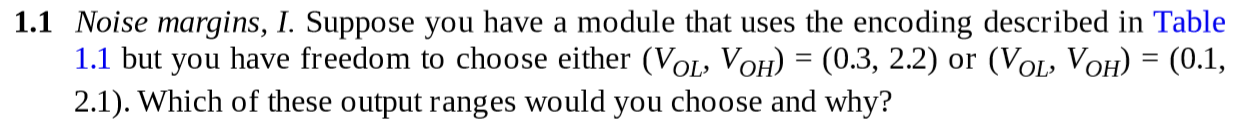
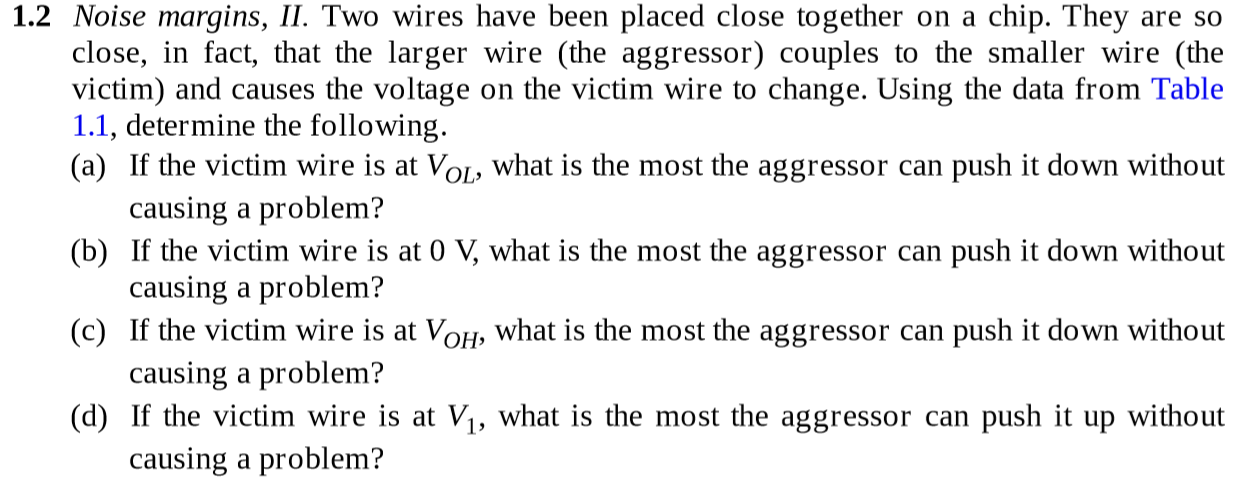
Ian Parker

Homework 1



I would choose (0.1,2.1) since it has a larger output range meaning there is a larger noise margin allowed before the system is damaged.

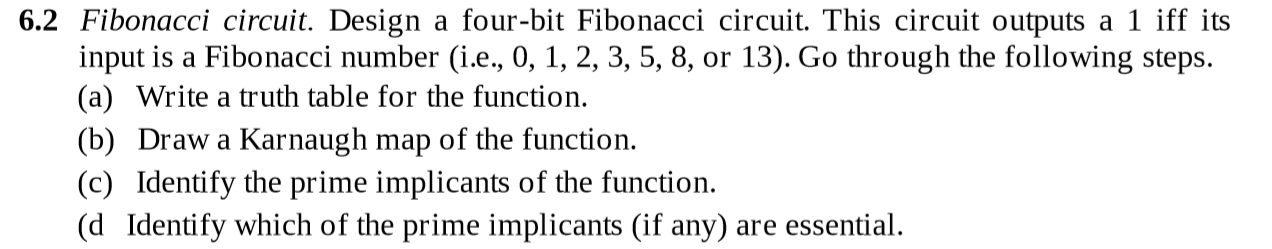


a) 0.5V

b) 0.3V

c) 0.4V

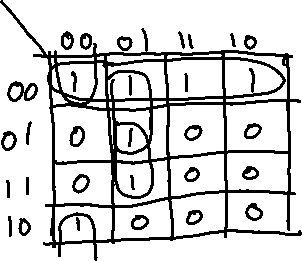
d) 0.3V?



a)

|  |  |  |
| --- | --- | --- |
| Number | Input | Output |
| 0 | 0000 | 1 |
| 1 | 0001 | 1 |
| 2 | 0010 | 1 |
| 3 | 0011 | 1 |
| 5 | 0101 | 1 |
| 8 | 1000 | 1 |
| 13 | 1101 | 1 |
| Otherwise |  | 0 |

b)



Function: BC’D + A’CD + B’C’D’ + A’B’ = F

c) Prime Implicants:

BC’D, A’CD, B’C’D’, A’B’

d) Essential Prime Implicants:

BC’D, B’C’D’, A’B’

Problem 6.2 VHDL Code

entity Fibonacci is

port (I: in STD\_LOGIC\_VECTOR(3 downto 0);

F: out STD\_LOGIC);

end entity;

architecture Behavior of Fibonacci is

begin

process (I)

begin

case I is

when “0000” => F <= ‘1’;

when “0001” => F <= ‘1’;

when “0010” => F <= ‘1’;

when “0011” => F <= ‘1’;

when “0101” => F <= ‘1’;

when “0000” => F <= ‘1’;

when “1000” => F <= ‘1’;

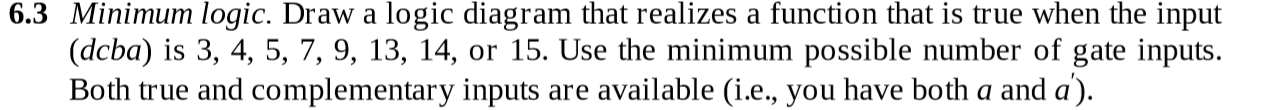
when “1101” => F <= ‘1’;

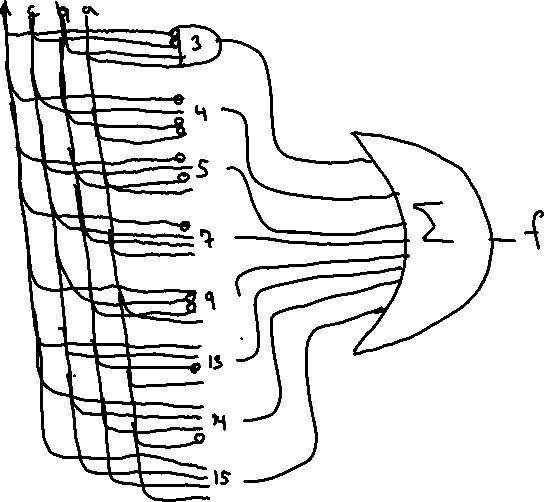
when others => F <= ‘0’;

end case;

end process;

end Behavior;





Problem 6.3 VHDL Code

entity minimumLogic is

port ( I: in STD\_LOGIC;

F : out STD\_LOGIC);

end entity;

architecture Behavior of minimumLogic is

begin

process (I)

begin

case I is

when “0011” => F <= ‘1’; --3

when “0100” => F <= ‘1’; --4

when “0101” => F <= ‘1’; --5

when “0111” => F <= ‘1’; --7

when “1001” => F <= ‘1’; --9

when “1101” => F <= ‘1’; --13

when “1110” => F <= ‘1’; --14

when “1111” => F <= ‘1’; --15

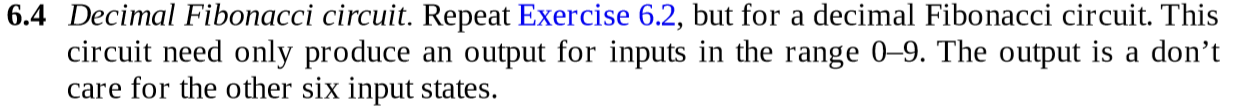
when others => F <= ‘0’; -- otherwise 0

end case;

end process;

end Behavior;





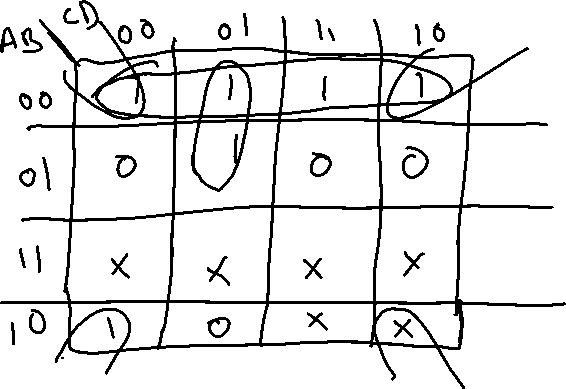


1. Truth Table



|  |  |  |
| --- | --- | --- |
| Number | Input | Output |
| 0 | 0000 | 1 |
| 1 | 0001 | 1 |
| 2 | 0010 | 1 |
| 3 | 0011 | 1 |
| 4 | 0100 | 0 |
| 5 | 0101 | 1 |
| 6 | 0110 | 0 |
| 7 | 0111 | 0 |
| 8 | 1000 | 1 |
| 9 | 1001 | 0 |
| 10 | 1010 | X |
| 11 | 1011 | X |
| 12 | 1100 | X |
| 13 | 1101 | X |
| 14 | 1110 | X |
| 15 | 1111 | X |

1. Karnaugh Map



Function: B’D’ + A’C’D + A’B’ = F

1. Prime Implicants:

B’D’, A’C’D, A’B

1. Essential Prime Implicants:

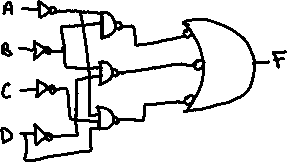
B’D, A’C’D, A’B’

1. Cover of Function:

F(A, B, C, D) = ((B’ D’) (A’ C’ D) (A’ B’))



1. CMOS Gate Circuit:



Problem 6.4 VHDL Code

entity DecimalFib is

port ( I : in STD\_LOGIC\_VECTOR(3 downto 0);

F : out STD\_LOGIC);

architecture Behavior of DecimalFIb is

begin

process (I)

begin

case I is

when “0000” => F <= ‘1’; --0

when “0001” => F <= ‘1’; --1

when “0010” => F <= ‘1’; --2

when “0011” => F <= ‘1’; --3

when “0101” => F <= ‘1’; --5

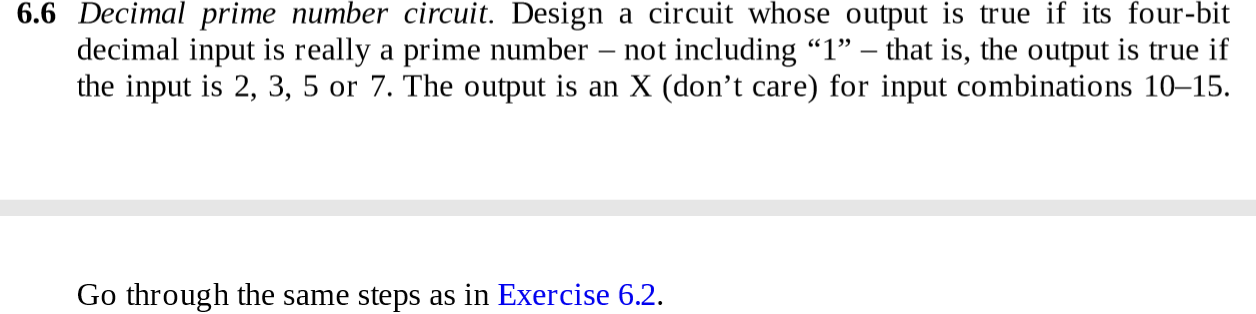
when “1000” => F <= ‘1’; --8

when others => F <= ‘0’; --otherwise 0

end case;

end process;

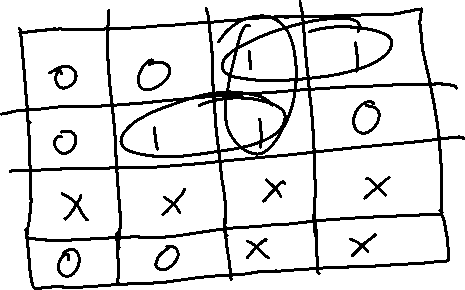
end Behavior;



1. Truth Table

|  |  |  |
| --- | --- | --- |
| Number | ABCD | F |
| 0 | 0000 | 0 |
| 1 | 0010 | 0 |
| 2 | 0011 | 1 |
| 3 | 0100 | 1 |
| 4 | 0100 | 0 |
| 5 | 0101 | 1 |
| 6 | 0110 | 0 |
| 7 | 0111 | 1 |
| 8 | 1000 | 0 |
| 9 | 1001 | 0 |
| 10 | 1010 | X |
| 11 | 1011 | X |
| 12 | 1100 | X |
| 13 | 1101 | X |
| 14 | 1110 | X |
| 15 | 1111 | X |

1. Karnaugh Map



Function: A’BD + A’CD + A’B’C = P

1. Prime Implicants:

A’BD, A’CD, A’B’C

1. Essential Prime Implicants:

A’BC, A’B’C

Problem 6.6 VHDL Code:

entity isPrime is

port (I : in STD\_LOGIC\_VECTOR(3 downto 0);

F: STD\_LOGIC);

end entity;

architecture Behavior of isPrime is

begin

process(I)

begin

case I is

when “0011” => F <= ‘1’; --2

when “0100” => F <= ‘1’; --3

when “0101” => F <= ‘1’; --5

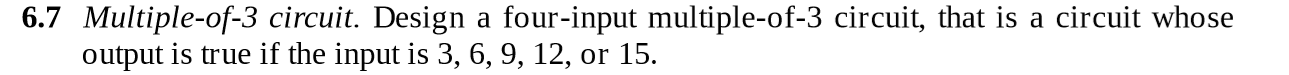
when “0111” => F <= ‘1’; --7

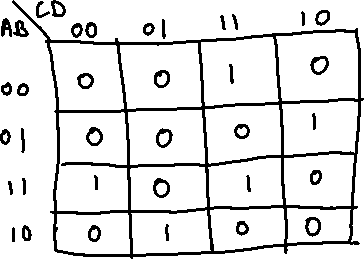
when others => F <= ‘0’; --otherwise 0

end case;

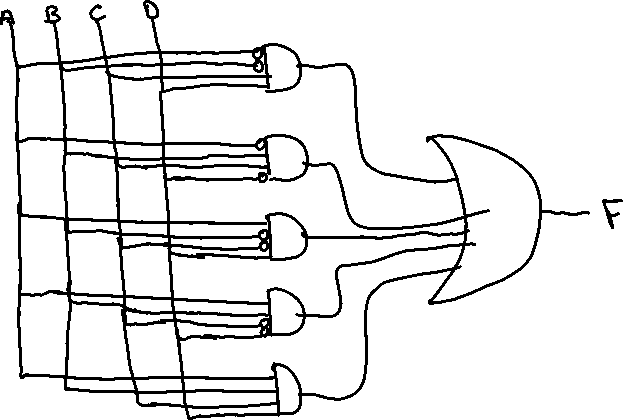
end process;

end behavior;





Function: A’B’CD + A’BCD’ + AB’C’D + ABC’D’ + ABCD = F



Problem 6.7 VHDL Code

entity multiples\_of\_three is

port ( I : in STD\_LOGIC\_VECTOR(3 downto 0)l

F: out STD\_LOGIC);

end entity;

architecture Behavior of multiples\_of\_three is

begin

process (I)

begin

case I is

when “0011” => F <= ‘1’;

when “0110” => F <= ‘1’;

when “1001” => F <= ‘1’;

when “1100” => F <= ‘1’;

when “1111” => F <= ‘1’;

when others => F <= ‘0’;

end case;

end process;

end Behavior;