Friedrich-Alexander-Universität
Erlangen-Nürnberg

Informatik 12
Am Weichselgarten 3
91058 Erlangen

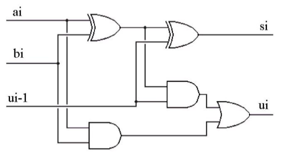
Reconfigurable Computing
Solution of Exercise 1

Answer 1

-

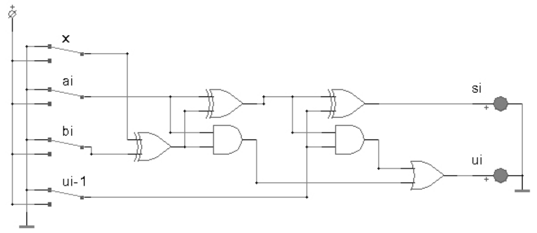
a_i	b_i	u_{i-1}	s_i	u_i
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

- The output S_i is output of one **xor**, and the inputs of **xor** are u_{i-1} and the sum of half adder with inputs of a_i and b_i.
  The output u_i will be 1,if at least 2 variable inputs are 1 . We can use intuitively two half adders



  to build the full adder circuit: the first half adder takes the inputs a_i and b_i. Its Partial-Sum, also the carry $u_i - 1$ are the inputs of the second half adder, and the outputs are S_i and u_i.

- $T_{min} = 3\tau$
  $F_{max} = 1/3\tau$

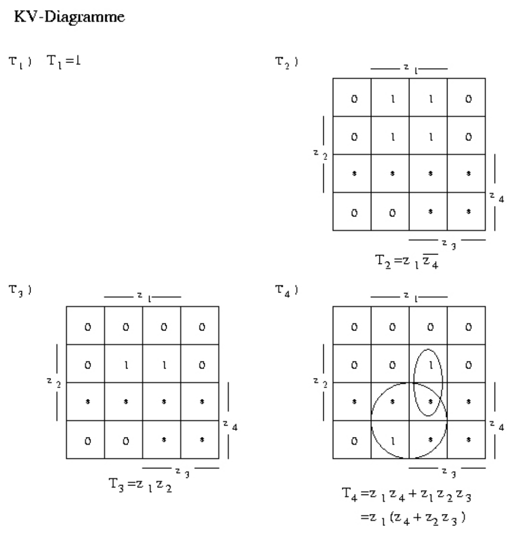


- Values in two's complement:
  Addition $a+b$: Control line $x=0$.
  Subtraction $a-b = a+\overline{b}+1$, Control line $x=1$. By XORing the control line x and b , we will have $\overline{b}$, if $x=1$. For having also the value 1 in the equation, the $0th$-add carry in the $0th$ Full adder is set one, when x is 1.

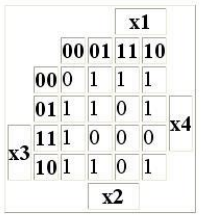
- 1. **Function of 2:1 Multiplexers (Inputs I_0, I_1, Output O, selector S):** $O(S,I_0,I_1) = \overline{S}(I_0) + S(I1)$
  2. **Shannon Expansion Theorem:** $f(S,I_0,I_1) = \overline{S} * f_1(S=0,I_0,I_1) + S * f_1(S=1,I_0,I_1)$
  3. **Function of 4:1 Multiplexer (Inputs I_0, I_0, I_2, I_3, Output O, selector S_0, S_1 :)** $O(S_0,S_1,I_0,I_1,I_2,I_3) = S_1\overline{S_0 S_1}(I_0) + \overline{S_0}S_1(I1) + S_0\overline{S_1}(I2) + S_1S_2(I3)$
  4. **Shannon Expansion Theorem (2nd Stage):**
     $F(x_1,.x_n) = F(x_1,.,x_i=1,.,x_n) * x_i + F(x_1,.,x_i=0,.,x_n) * \overline{x_i} = [F(x_1,.,x_i=1,.,x_j=1,..x_n) * x_i + F(x_1,.,x_i=0,.,x_j=1,.,x_n) * \overline{x_i}]x_j + [F(x_1,.,x_i=1,.,x_j=0,.,x_n) * x_i + F(x_1,.,x_i=0,.,x_j=0,.,x_n) * \overline{x_i}]\overline{x_j}$
     $= F(x_1,.,x_i=1,.,x_j=1,..x_n) * x_ix_j + F(x_1,.,x_i=0,.,x_j=1,.,x_n) * \overline{x_i}x_j + F(x_1,.,x_i=1,.,x_j=0,.,x_n) * x_i\overline{x_j} + F(x_1,.,x_i=0,.,x_j=0,.,x_n) * \overline{x_ix_j}$
  5. $s_i = \overline{a_i} * \overline{b_i} * u_{i-1} + \overline{a_i} * b_i * \overline{u_{i-1}} + a_i * \overline{b_i} * \overline{u_{i-1}} + a_i * b_i * u_{i-1}$
  6. $u_i = \overline{a_i} * b_i * u_{i-1} + a_i * \overline{b_i} * u_{i-1} + a_i * b_i * \overline{u_{i-1}} + a_i * b_i * u_{i-1} = \overline{a_i} * \overline{b_i} * (0) + a_i * \overline{b_i} * u_{i-1} + \overline{a_i} * b_i * u_{i-1} + a_i * b_i * (1)$

Answer 2

z	$z\prime$	T_4	T_3	T_2	T_1
0000	0001	0	0	0	1
0001	0010	0	0	1	1
0010	0011	0	0	0	1
0011	0100	0	1	1	1
0100	0101	0	0	0	1
0101	0110	0	0	1	1
0110	0111	0	0	0	1
0111	1000	1	1	1	1
1000	1001	0	0	0	1
1001	0000	1	0	0	1
1010	xxxx	x	x	x	x
1011	xxxx	x	x	x	x
1100	xxxx	x	x	x	x
1101	xxxx	x	x	x	x
1110	xxxx	x	x	x	x
1111	xxxx	x	x	x	x



Answer 3



				x1		
		00	**01**	**11**	**10**	
	00	0	1	1	1	
	01	1	1	0	1	**x4**
x3	**11**	1	0	0	0	**x4**
x3	**10**	1	1	0	1	
			x2	**x2**		

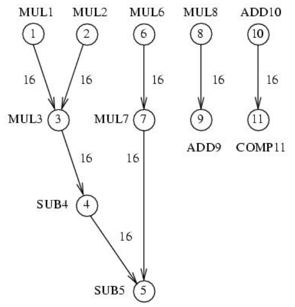
SoP: $f = x1 * \overline{x2} * \overline{x3} + x2 * \overline{x3} * \overline{x4} + \overline{x1} * \overline{x3} * x4 + \overline{x1} * \overline{x2} * x3 + \overline{x1} * x3 * \overline{x4} + \overline{x2} * x3 * \overline{x4}$;
cost = 7 gates + 24 inputs = 31 units (several other answers are equivalent)

PoS: $f = (x1 + x2 + x3 + x4) * (\overline{x1} + \overline{x2} + \overline{x3}) * (\overline{x1} + \overline{x2} + \overline{x4}) * (\overline{x1} + \overline{x3} + \overline{x4}) * (\overline{x2} + \overline{x3} + \overline{x4})$;
cost = 6 gates + 21 inputs = 27 units

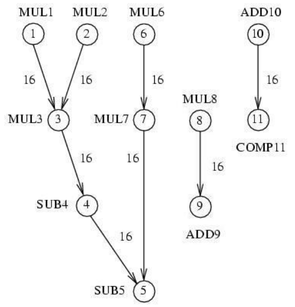
PoS is less expensive than SoP.

Answer 4

In Von-Neumann machine : 1 Instruction cycle = 5 clock cycle
$T_{Von-Neumann} = 3 * 5 + \lceil \frac{a - x_0}{dx} \rceil * 11 * 5$



$4 Mul, 2 Add/Sub/Comp$
$T_{Hardware} = 1 + \lceil \frac{a - x_0}{dx} \rceil * 4$
Optimized architecture :$3 Mul, 2 Add/Sub/Comp$



Friedrich-Alexander-Universität
Erlangen-Nürnberg

design

Informatik 12
Am Weichselgarten 3
91058 Erlangen

Reconfigurable Computing
Solution of Exercise 2

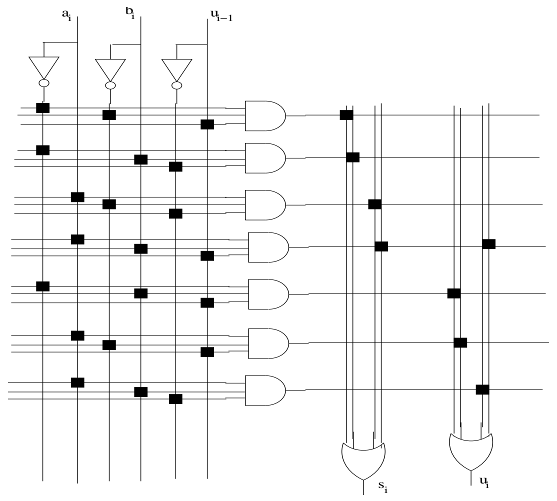
Answer 1

- Truth table for the 1-bit full adder.

a_i	b_i	u_{i-1}	s_i	u_i
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

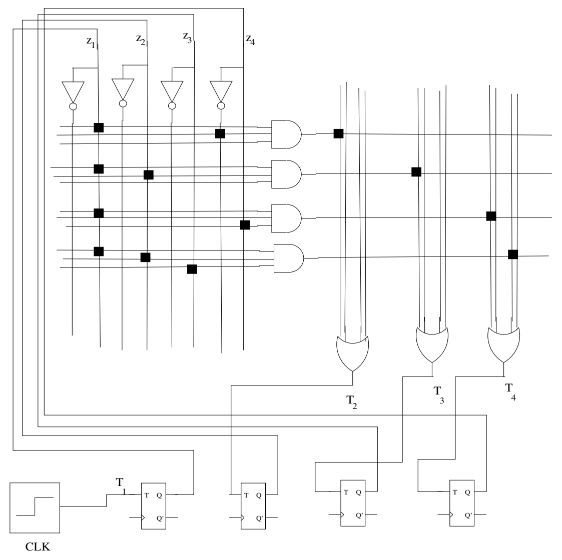
$$s_i = \overline{a_i}\,\overline{b_i}u_{i-1} + \overline{a_i}b_i\overline{u_{i-1}} + a_i\overline{b_i}\,\overline{u_{i-1}} + a_ib_iu_{i-1}$$
$$u_i = \overline{a_i}b_iu_{i-1} + a_i\overline{b_i}u_{i-1} + a_ib_i\overline{u_{i-1}} + a_ib_iu_{i-1}$$

PLA-Mapping:



Answer 2

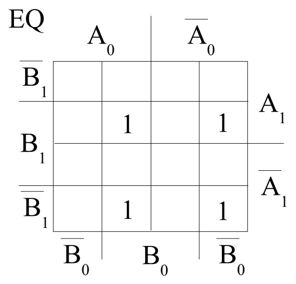
PLA-Mapping of the Mod-10 counter:

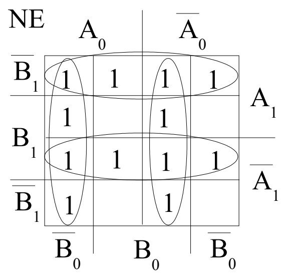


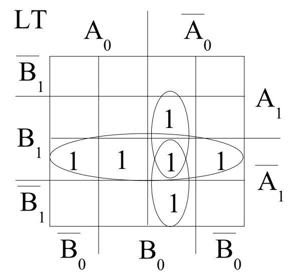
Answer 3

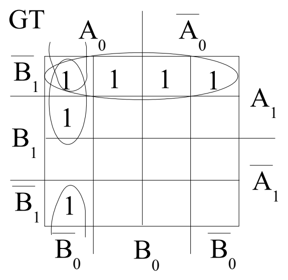
Comparator circuit for two ASCII characters:
Truth table of the 2-bit magnitude comparators

a_1	a_0	b_1	b_0	EQ	NE	LT	GT
0	0	0	0	1	0	0	0
0	0	0	1	0	1	1	0
0	0	1	0	0	1	1	0
0	0	1	1	0	1	1	0
0	1	0	0	0	1	0	1
0	1	0	1	1	0	0	0
0	1	1	0	0	1	1	0
0	1	1	1	0	1	1	0
1	0	0	0	0	1	0	1
1	0	0	1	0	1	0	1
1	0	1	0	1	0	0	0
1	0	1	1	0	1	1	0
1	1	0	0	0	1	0	1
1	1	0	1	0	1	0	1
1	1	1	0	0	1	0	1
1	1	1	1	1	0	0	0









$$EQ = A_0B_0A_1B_1 + A_0\overline{B_0}A_1\overline{B_1} + \overline{A_0}B_0\overline{A_1}B_1 + \overline{A_0}\overline{B_0}\overline{A_1}\overline{B_1}$$
$$NE = A_0\overline{B_0} + \overline{A_0}B_0 + A_1\overline{B_1} + \overline{A_1}B_1$$
$$LT = \overline{A_1}B_1 + \overline{A_0}B_0\overline{A_1} + \overline{A_0}B_0B_1$$
$$GT = A_1\overline{B_1} + A_0A_1\overline{B_0} + A_0\overline{B_1}\overline{B_0}$$

Answer 4

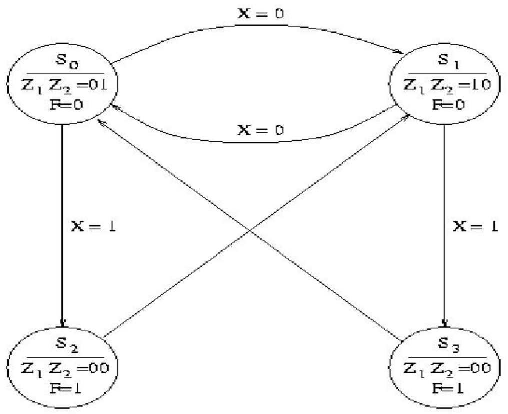
Parallel code for the GCD.

```
int GGT(int x, int y)
{
        static int n = 0;
        while(even(x) && even(y))
        {
            n = n +1;
            x = half(x);
            y = half(y);
        }
        while(x != y)
        {
            while (even(x)) x = half(x);
            while (even(y)) y = half(y);

            if (x < y)
            {
                y = half(y-x);
            }
            else
            {
                if (x != y)
                {
                    x = half(x-y);
                }
            }
        }
        while(n != 0)
        {
           n = n - 1;
           x = twice(x);
        }
}
```

Friedrich-Alexander-Universität
Erlangen-Nürnberg

Informatik 12
Am Weichselgarten 3
91058 Erlangen

Reconfigurable Computing
Solution of Exercise 1

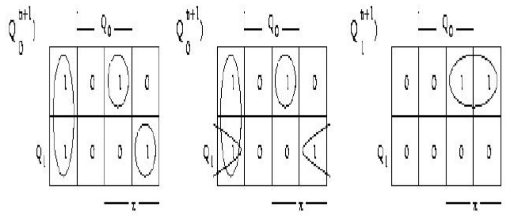


-

State	Inputs		Outputs	
	x = 0	x = 1	F	z_1z_2
s_0	s_1	s_2	0	01
s_1	s_0	s_3	0	10
s_2	s_1	s_1	1	00
s_3	s_0	s_0	1	00

State	Inputs		Outputs	
Q_1Q_0	x = 0	x = 1	F	z_1z_2
00	01	10	0	01
01	00	11	0	10
10	01	01	1	00
11	00	00	1	00

-

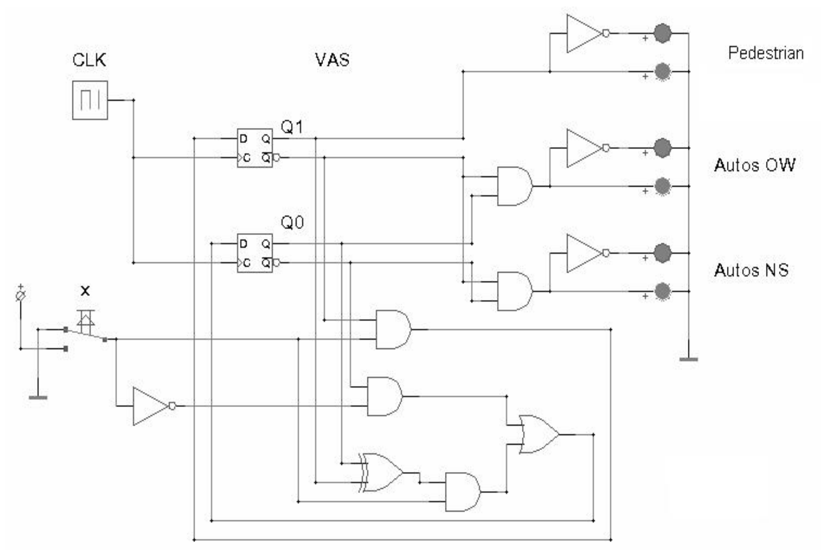


$$Q_1^{n+1} = \overline{Q_1^n} \wedge x$$

$$Q_0^{n+1} = \overline{Q_0^n} \wedge \bar{x} \vee Q_0^n \wedge \overline{Q_1^n} \wedge x \vee \overline{Q_0^n} \wedge Q_1^n \wedge x$$

$$Q_n^{n+1} = \overline{Q_0^n} \wedge \bar{x} \vee (Q_0^n \oplus Q_1^n) \wedge x$$

$$\text{alternative} \quad Q_0^{n+1} = \overline{Q_0^n} \wedge \bar{x} \vee Q_0^n \wedge \overline{Q_1^n} \vee \overline{Q_0^n} \wedge Q_1^n \wedge x$$



-

-
  ```
  entity controller
  port (x,clk,reset: in std_logic;
          F,z1,z2: out std_logic);
  end entity;

  architecture structural of controller is
   type state_type is (S0, S1, S2, S3);
   signal current_state, next_state: state_type;

   begin
      comp_state: process(x)
          begin
           case current_state is
             when S0 =>
               if (x=0) then next_state <= S1;
               else next_state <= S2; end if;
               z1 <= '0';  z2 <= '1';  F <= '0';
             when S1 =>
               if (x=0) then next_state <= S0;
               else next_state <= S3; end if;
               z1 <= '1';  z2 <= '0';  F <= '0';
             when S2 =>
               next_state <= S1;
               z1 <= '0';  z2 <= '0';  F <= '1';
             when S3 =>
               next_state <= S0;
               z1 <= '0';  z2 <= '0';  F <= '1';
            end case;
        end process;

       set_state: process (clk, reset)
        begin
          if (reset = '1') then
             current_state <= S0;
          elsif(clk´event and clk=´1´) then
             current_state <= next_state;
        end if;
       end process;
  end architecture;
  ```

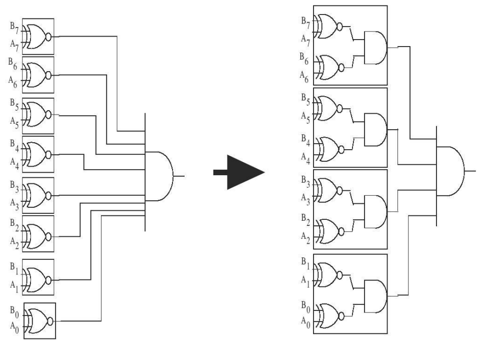
Friedrich-Alexander-Universität
Erlangen-Nürnberg

Informatik 12
Am Weichselgarten 3
91058 Erlangen

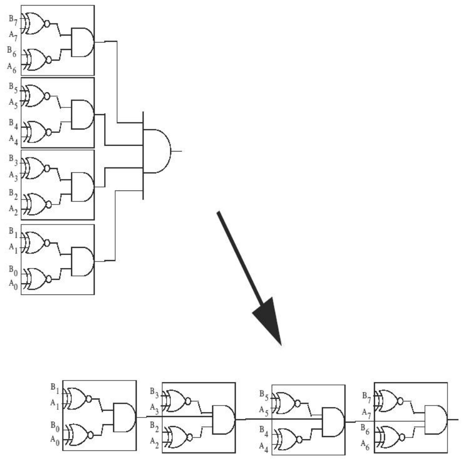
Reconfigurable Computing
Solution of Exercise 4

Answer 1

- Chortle-crf method Step 1



- Chortle-crf method Step 2

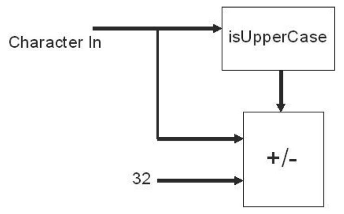


Answer 2

Labeling: All nodes are labeled with the depth of the LUT in which they will be implemented.

- Labeling is done in topological order.
- The PI are assigned the label 0.
- For each node t in the graph the cone N_t is transformed in to a network by inserting a source node.
- The label $l(t)$ of t is calculated on the basis of all k-feasible cuts in the network, where k is the number of the inputs for the LUTs. The minimum heighth_{min} over all feasible cut is computed.$l(t) = h_{min}$.
- To compute a k-feasible cut with height $p-1$, the network N_t is transformed into a new network $N^1{}_t$, by collapsing all the nodes with maximum label together with t into a new node t_1. The problem of computing a k-feasible cut with height $p-1$ N_t is now reduced to that of computing a k-feasible cut in $N^1{}_t$.
- A second transformation is done on $N^1{}_t$ to obtain a new network $N^2{}_t$. The goal is to map the node-cut problem in $N^1{}_t$ into an edge-cut problem into $N^2{}_t$, to solve the problem in $N^2{}_t$ and derive the solution for $N^1{}_t$. In $N^2{}_t$, the problem of finding a cut in $N^2{}_t$ whose edge cut-size is no more than K. This is done with the augmenting path method according to the min-cut max-flow theorem of Ford and Fulkerson.

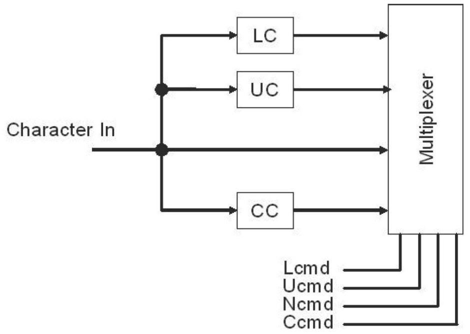
Answer 3



-

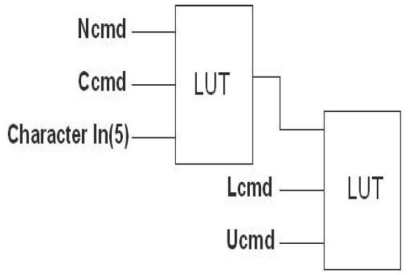
-
  ```
  entity ChangeCase
  port (isUC: in std_logic;
        Cin: in std_logic_vector(7 downto 0);
        Result: out std_logic_vector(7 downto 0));
  end entity;

  architecture structural of ChangeCase is
  ```

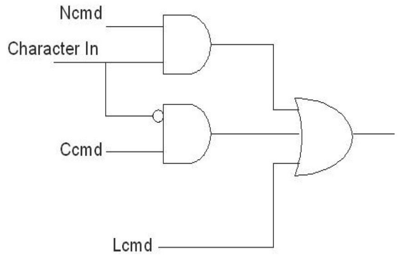
```
  begin
    if(isUC='1') then
      Result <= Cin + 32;
    else
      Result <= Cin - 32;
    end if;
 end architecture;
```



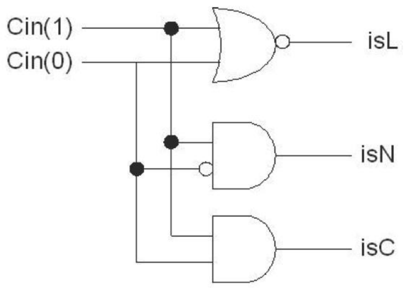
-
  ```
  entity Transfer
  port (Lcmd,Ucmd,Ncmd,Ccmd: in std_logic;Cin: in std_logic_vector(7 downto 0);
        Result: in std_logic_vector(7 downto 0));
  end entity;

  architecture structural of Transfer is

  component ChangeCase
  port (isUC: in std_logic;
        Cin: in std_logic_vector(7 downto 0);
        Result: out std_logic_vector(7 downto 0));
  end component;

  begin

   if(Lcmd='1') then
     Result <= Cin - 32;
   elsif(Ucmd='1') then
     Result <= Cin + 32;
   elsif(Ncmd='1') then
     Result <= Cin;
   elsif(Ccmd='1') then
     cc:ChangeCase
     port map(Cin(5),Cin,Result);
  end if;
  end architecture;
  ```
- The result is actually only depends on bit 5 of the input character. Suppose $Ccmd = 1$, this

implies $Ncmd = Lcmd = 0$. Then, the output is the complement of bit 5 of input. Similarly, we can analyze the other three cases. Also, when $Ucmd = 1$, the output is 0.



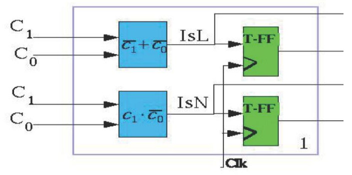


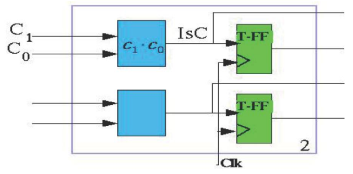
-
  ```
  entity CharComp
  port (Cin,ESC: in std_logic_vector(7 downto 0);
          isESC: out std_logic);
  end entity;
  architecture structural of CharComp is

  begin
  if(Cin=ESC) then
  isESC <= ’1’;
  else
  isESC <= ’0’;
  end if;
  end architecture;
  ```
- Let us consider the codes of the four letters that may appear in an escape sequence:
  $L = 0x4c = 01001100 = 74$
  $U = 0x55 = 01010101 = 85$
  $N = 0x4E = 01001110 = 78$
  $C = 0x43 = 01001110 = 67$
  Notice that the two least significant bits are sufficient to distinguish them. Furthermore, from the discussion of the transform block, we know that we do not need to produce $Ucmd$. Therefore we can implement the command decoder as shown in the following:



```
entity Comm_Interpret
port (Cin: in std_logic_vector(7 downto 0);
         Lcmd,Ucmd,Ncmd,Ccmd: out std_logic);
end entity;
architecture structural of Comm_Interpret is

begin
 case Cin(1 downto 0) is
    when ‘‘00’’ =>
        Lcmd <= ’1’;
        Ucmd <= ’0’;
        Ncmd <= ’0’;
        Ccmd <= ’0’;
    when ‘‘01’’ =>
```

```
        Lcmd <= ’0’;
        Ucmd <= ’1’;
        Ncmd <= ’0’;
        Ccmd <= ’0’;
    when ‘‘10’’ =>
        Lcmd <= ’0’;
        Ucmd <= ’0’;
        Ncmd <= ’1’;
        Ccmd <= ’0’;
    when ‘‘11’’ =>
        Lcmd <= ’0’;
        Ucmd <= ’0’;
        Ncmd <= ’0’;
        Ccmd <= ’1’;
    when others =>
  end case;
end architecture;
```





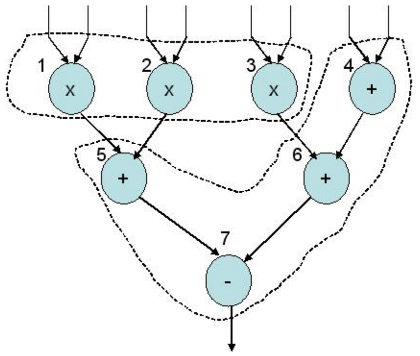
Friedrich-Alexander-Universität
Erlangen-Nürnberg

Informatik 12
Am Weichselgarten 3
91058 Erlangen

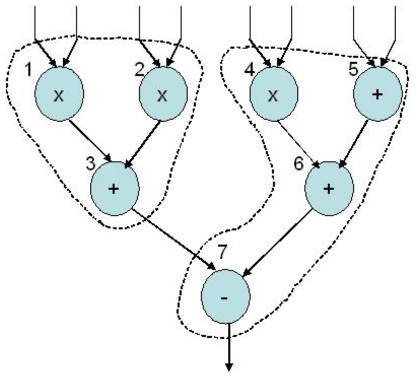
Reconfigurable Computing
Solution of Exercise 5

Answer 1

- Connectivity : $con(G) = 2 * |E|/(|V|^2 - |V|) = 2 * 15/(7^2 - 7) = 5/7 = 0.714$
- Straight Forward Partitioning



- List Scheduling Partitioning



- Quality of partitioning : Average Connectivity over partitions
  Straight-Forward :
  $con(P1) = 0$
  $con(P2) = 2 * 3/(16 - 4) = 0.5$
  $Q(SF) = 0.25$

  List-Scheduling :
  $con(P1) = 2 * 2/(9 - 3) = 0.66$
  $con(P2) = 2 * 3/(16 - 4) = 6/12 = 0.5$
  $Q(LS) = 0.58$

Answer 2

Connection matrix C:

$$C = \begin{bmatrix} 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 \end{bmatrix}$$

Degree matrix D:

$$D = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 3 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 2 \end{bmatrix}$$

Laplacian matrix $B = D - C$:

$$B = \begin{bmatrix} 1 & 0 & 0 & 0 & -1 & 0 & 0 \\ 0 & 1 & 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 & 0 & -1 & 0 \\ -1 & -1 & 0 & 0 & 3 & 0 & -1 \\ 0 & 0 & -1 & -1 & 0 & 3 & -1 \\ 0 & 0 & 0 & 0 & -1 & -1 & 2 \end{bmatrix}$$

Three smallest non-zero eigenvalues:
0.267949, 1 , 1, corresponding respectively to the 4-th, 5-th and 6-th eigenvectors.

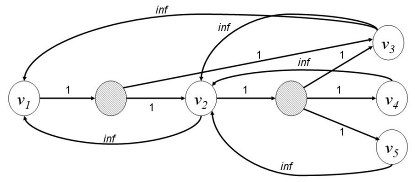
$$x = \begin{bmatrix} +0.444024 \\ +0.444024 \\ -0.444035 \\ -0.444035 \\ +0.325108 \\ -0.325047 \\ 0.000000 \end{bmatrix}$$

$$y = \begin{bmatrix} 0.707107 \\ -0.707107 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

$$t = \begin{bmatrix} 0 \\ 0 \\ -0.707107 \\ +0.707107 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

Temporal partitions:
The eigenvectors describe the placement of the 7 nodes in the 3-dimensional space (x, y, t). E.g. the coordinates of node $n_1 = (0.444024, 0.707107, 0)$, of node $n_2 = (0.444024, -0.707107, 0)$, etc. From the eigenvector t follows, that there will be three temporal partitions: one for $t = -0.707107$, the second for $t = 0$ and the third for $t = 0.707107$.

Answer 3



Answer 4

Area constraints:
$P_1 : S(1) + S(12) + S(7) + S(4) \le S(device)$, i.e for P1 of the first partition :$10 + 80 + 50 + 25 = 165 \le 150 \rightarrow$ not fulfilled
For P_2, P_3, P_4 of both partitions these constraints must be computed in the same way.

Terminal constraints:
$P_1 : T(1,9) + T(12,8) + T(4,5) + T(7,3) + T(PI,1) + T(PI,7) \le T(device)$, i.e for P1 of the first partition : $8 \times 6 = 48 \le 30 \rightarrow$ not fulfilled
For P_2, P_3, P_4 of both partitions these constraints must be computed in the same way.

Unique assignment constraints:
For vertex v_4 of the second Partition: $v_4 : y_{41} = y_{42} = 0, y_{43} = y_{44} = 1$,

$v_4 : y_{41} + y_{42} + y_{43} + y_{44} = 2 \rightarrow$ not fulfilled

All other vertices must be computed in the same way.

Result: Both partitions are illegal, as none of them fulfills all constraints.

Friedrich-Alexander-Universität
Erlangen-Nürnberg

Informatik 12
Am Weichselgarten 3
91058 Erlangen

Reconfigurable Computing
Solution of Exercise 6

Answer 1

- **Component Graph (see Figure 1 and 2):**
  - For the X-Axis: connect two nodes with an edge, if they are placed in the same column
  - For the Y-Axis: connect two nodes with an edge, if they are placed in the same row
- **Complement Graph (see Figure 1 and 2):**
  - Invert the edges of the component graph
  - For the X-Axis: This graph shows you the possible left/right neighbours of a node v_i
  - For the Y-Axis: This graph shows you the possible upper/lower neighbours of a node v_i

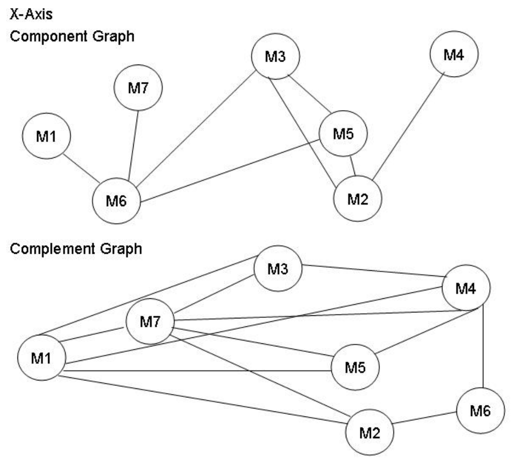


Figure 1: X-Axis: Component and Complement Graph

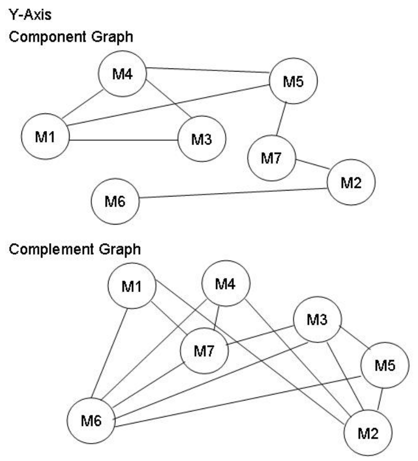


Figure 2: Y-Axis: Component and Complement Graph

- **Transitive Orientation (see Figure 3):**
  - Several possible Solutions
  - Has to be cycle-free
  - For the X-Axis: Gives the order in x-direction between the different nodes
  - For the Y-Axis: Gives the order in y-direction between the different nodes

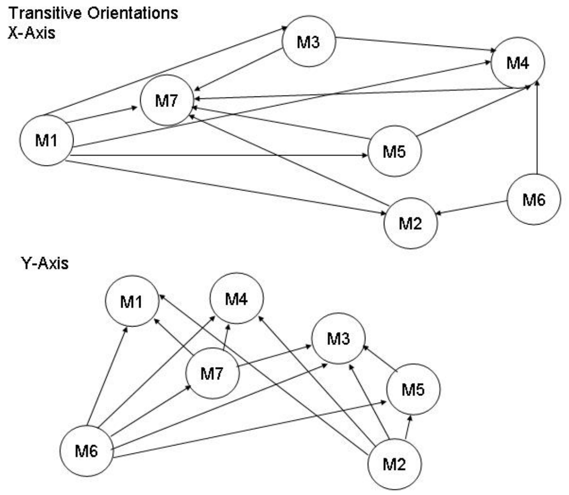


Figure 3: Transitive Orientations for X- and Y-Axis

- **The corresponding placement for the two Transitive Orientations is illustrated in Figure 4.**

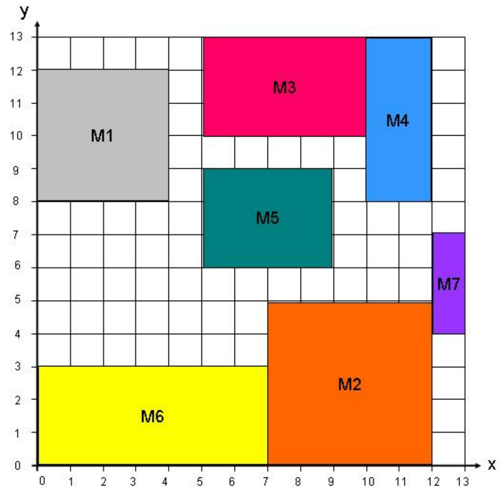


Figure 4: Corresponding Placement according to the two previously computed transitive orientations

- **The given partial ordering does not affect the two dimensional placement.**
  Neither a transitive implication nor a path implication conflict exists.
  In the third dimension, component M8 will replace component M1 and component M9 will replace component M6.

Answer 2

- Figure 5 shows the computed 8 maximal empty rectangles: A,B,C,D,E,F,G,H

- Figure 6 shows the computed 7 non overlapping empty rectangles:

- Figure 7 shows the computed 11 maximal empty rectangles:

- According to communications, it will be placed as shown in Figure 8: Following the RC lecture the communication aware placement according to the Ahmadinia-Bobda approach can be found by solving the following equation for x_n and y_n:

$$\min\left\{\sum_{i=1}^{n-1}\left(\left(x_n+\frac{w_n}{2}-x_i-\frac{w_i}{2}\right)^2+\left(y_n+\frac{h_n}{2}-y_i-\frac{h_i}{2}\right)^2 * w_{in}\right)\right\} \tag{1}$$

As x_n and y_n are independent from each other the above equation can be split into 2 equations:

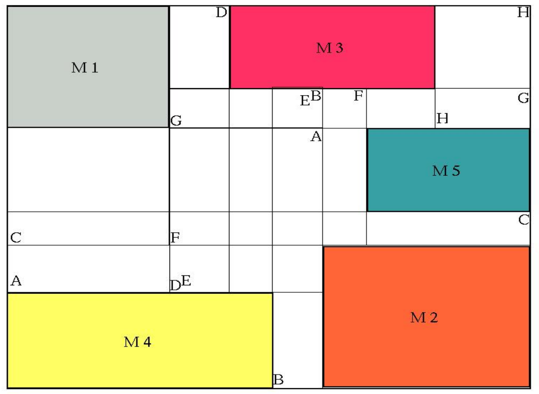


Figure 5: 8 maximal empty rectangles

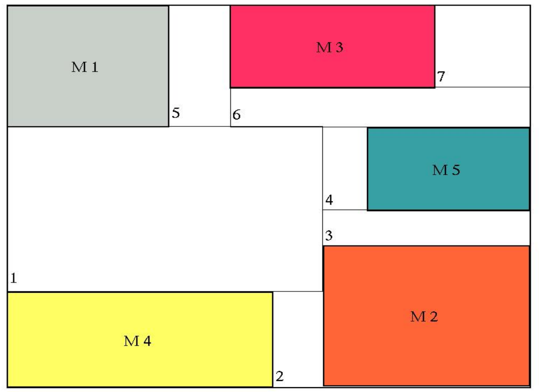


Figure 6: 7 non overlapping empty rectangles

$$\min\left\{\sum_{i=1}^{n-1}\left(\left(x_n+\frac{w_n}{2}-x_i-\frac{w_i}{2}\right)^2 * w_{in}\right)\right\} \tag{2}$$

$$\min\left\{\sum_{i=1}^{n-1}\left(\left(y_n+\frac{h_n}{2}-y_i-\frac{h_i}{2}\right)^2 * w_{in}\right)\right\} \tag{3}$$

To find the minima, we have to derivate both functions and set them equal to 0:

$$0=\frac{\partial\left\{\sum_{i=1}^{n-1}\left(\left(x_n+\frac{w_n}{2}-x_i-\frac{w_i}{2}\right)^2 * w_{in}\right)\right\}}{\partial x_n} \tag{4}$$

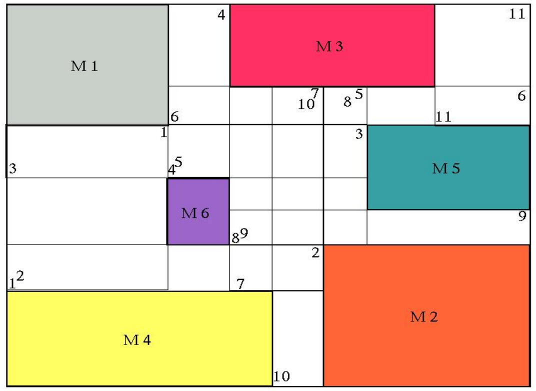


Figure 7: 11 maximal empty rectangles

$$0 = \frac{\partial \left\{ \sum_{i=1}^{n-1} \left(\left(y_n + \frac{h_n}{2} - y_i - \frac{h_i}{2} \right)^2 * w_{in} \right) \right\}}{\partial y_n} \tag{5}$$

The resulting equations, which now can be solved for x_n and y_n by inserting the known values for the different x_i, y_i, w_i, h_i, w_n, h_n and w_{in} are:

$$x_n = \frac{\sum_{i=1}^{n-1} w_{in} * \left(\left(x_i + \frac{w_i}{2} \right) - \frac{w_n}{2} \right)}{\sum_{i=1}^{n-1} w_{in}} \tag{6}$$

$$y_n = \frac{\sum_{i=1}^{n-1} w_{in} * \left(\left(y_i + \frac{h_i}{2} \right) - \frac{h_n}{2} \right)}{\sum_{i=1}^{n-1} w_{in}} \tag{7}$$

You can solve these equations by inserting the following values:

Center of M3: $x_3 + \frac{w_3}{2} = 8$, $y_3 + \frac{h_3}{2} = 11$
Center of M5: $x_5 + \frac{w_5}{2} = 11$, $y_5 + \frac{h_5}{2} = 7$
IO-Pins on the right border: $x_{a1} = 13$, $y_{a1} = 10$
IO-Pins on the upper border: $x_{a2} = 11$, $y_{a2} = 13$

The resulting placement coordinates (bottom left corner of M6) for M6 are:

$$x_n = 10, y_n = 9{,}25$$

This coordinate is within M3 and therefore it is an illegal placement. To find the next best placement, we go from this point into the four cardinal points. In the directions north, west and south, no legal placements can be found. However in direction east a legal placement is found on which we can place M6.

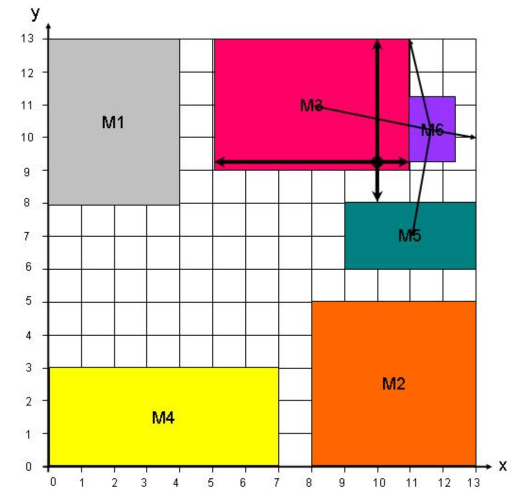


Figure 8: Placement of M6 according to the Ahmadinia-Bobda-approach