Case Study 9
Decomposition in VLSI Design

Design a vending machine controller
Fang Qin

Vending machine controller model

Function analysis
Coin counting

Return money

Excess money

Item selection

Note: Item_select includes itemA_select, itemB_select, itemC_select.
Item_out includes itemA_out, itemB_out, itemC_out.
Function analysis

**Change calculation**

**Change out**

Function analysis

**BCD indicator**

**Approach 1**

Structural Decomposition

Function-input/output matrix after branch-and-bound algorithm

Function-input/output matrix after introducing additional inputs/outputs

Approach 1

Structural Decomposition

**M1**

**M2**

**M3**

**M4**
**Approach I Structural Decomposition**

**Module design**

**Module 1: Coin Handler**

1. **Coin handler**
   - Nickel_in
   - Dime_in
   - Quarter_in
   - Return_money_in
   - Total_amount
   - Return_money_out
   - Excess_money_out

2. **Count coins**
   - IF nickel_in = '1'
     THEN
     total_amount := total_amount + 05;
   END IF;
   - IF dime_in = '1'
     THEN
     total_amount := total_amount + 10;
   END IF;
   - IF quarter_in = '1'
     THEN
     total_amount := total_amount + 25;
   END IF;

3. **Return money**
   - IF return_money = '1'
     THEN
     return_money_out = '1';
   END IF;

4. **Handle excess money**
   - IF total_amount > 95
     THEN
     excess_money_out = '1';
   END IF;

**Approach I Structural Decomposition**

**Module design**

**Module 2: Item Processor**

5. **Select item A**
   - IF ItemA_select = '1'
     THEN
     IF total_amount >= 55
     THEN
     ItemA_out = '1';
     Change := total_amount – 55;
     END IF;
   END IF;

6. **Select item B**
   - IF ItemB_select = '1'
     THEN
     IF total_amount >= 65
     THEN
     ItemB_out = '1';
     Change := total_amount – 65;
     END IF;
   END IF;

7. **Select item C**
   - IF ItemC_select = '1'
     THEN
     IF total_amount >= 75
     THEN
     ItemC_out = '1';
     Change := total_amount – 75;
     END IF;
   END IF;

**Approach I Structural Decomposition**

**Module design**

**Module 3: Change maker**

8. **Change (cents)**
   - Nickel_out
   - Dime_out
   - Dimes_out
   - Quarter_out

<table>
<thead>
<tr>
<th>Change (cents)</th>
<th>Nickel_out</th>
<th>Dime_out</th>
<th>Dimes_out</th>
<th>Quarter_out</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>10</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>15</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>20</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>25</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>30</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>35</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>40</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>45</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>50</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>55</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>60</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>65</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>70</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>75</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>80</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>85</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>90</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>95</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

9. **Suppose the change is 35 cents.**

   Following VHDL is used:
   - IF change = 35
     THEN
     Nickel_out = '1';
     Dime_out = '1';
     Dimes_out = '1';
     Quarter_out = '0';
     Total_amount := 0;
     Change := 0;
     END IF;

**Approach I Structural Decomposition**

**Module design**

**Module 4: BCD indicator**

10. **Change**
    - BCD_high
    - BCD_low

    | Amount (cents) | BCD_high | BCD_low |
    |---------------|----------|---------|
    | 0             | 0000     | 0000    |
    | 5             | 0000     | 0101    |
    | 10            | 0001     | 0000    |
    | 15            | 0001     | 0101    |
    | 20            | 0010     | 0000    |
    | 25            | 0010     | 0101    |
    | 30            | 0011     | 0000    |
    | 35            | 0011     | 0101    |
    | 40            | 0100     | 0000    |
    | 45            | 0100     | 0101    |
    | 50            | 0101     | 0000    |
    | 55            | 0101     | 0101    |
    | 60            | 0110     | 0000    |
    | 65            | 0110     | 0101    |
    | 70            | 0111     | 0000    |
    | 75            | 0111     | 0101    |
    | 80            | 1000     | 0000    |
    | 85            | 1000     | 0101    |
    | 90            | 1001     | 0000    |
    | 95            | 1001     | 0101    |

11. **Suppose the amount is 35.**

   Following VHDL is used:
   - IF amount = 35
     THEN
     BCD_low = "0101";
     BCD_high = "0011";
     END IF;

**Approach I Structural Decomposition**

**Module design**

**System design**

**Module interaction study**

12. **Coin handler**
    - Nickel_in
    - Dime_in
    - Quarter_in
    - Return_money_in
    - Total_amount
    - Return_money_out
    - Excess_money_out

13. **Item processor**
    - Item_select
    - Total_amount
    - Item_out
    - Change

    | ITEM PRICE(CENTS) | A 5 | B 6 | C 7 |
    |-------------------|-----|-----|-----|
    | 5                 | 00  | 01  | 02  |
    | 10                | 02  | 03  | 04  |
    | 15                | 03  | 04  | 05  |
    | 20                | 04  | 05  | 06  |
    | 25                | 05  | 06  | 07  |
    | 30                | 06  | 07  | 08  |
    | 35                | 07  | 08  | 09  |
    | 40                | 08  | 09  | 10  |
    | 45                | 09  | 10  | 11  |
    | 50                | 10  | 11  | 12  |
    | 55                | 11  | 12  | 13  |
    | 60                | 12  | 13  | 14  |
    | 65                | 13  | 14  | 15  |
    | 70                | 14  | 15  | 16  |
    | 75                | 15  | 16  | 17  |
    | 80                | 16  | 17  | 18  |
    | 85                | 17  | 18  | 19  |
    | 90                | 18  | 19  | 20  |
    | 95                | 19  | 20  | 21  |

14. **Suppose the amount is 35.**

   Following VHDL is used:
   - IF amount = 35
     THEN
     BCD_low = "0101";
     BCD_high = "0011";
     END IF;

**Approach I Structural Decomposition**

**System design**

**Vending machine configuration**

15. **Coin handler**
    - Nickel_in
    - Dime_in
    - Quarter_in
    - Return_money_in
    - Total_amount
    - Return_money_out
    - Excess_money

16. **Item processor**
    - Item_select
    - Total_amount
    - Item_out
    - Change

17. **Change maker**
    - Nickel_out
    - Dime_out
    - Dimes_out
    - Quarter_out

18. **BCD indicator**
    - Total_amount
    - Change
    - BCD_high
    - BCD_low

19. **Clock**
    - Reset

20. **System interaction**

   ![System interaction diagram]

21. **Vending machine configuration**

   ![Vending machine configuration diagram]
Approach II
Process Decomposition

1: Idle state

IF nickel_in = '1' THEN
   total_amount := total_amount + 05;
   next_state = coin_collect_state;
END IF;

IF dime_in = '1' THEN
   total_amount := total_amount + 10;
   next_state = coin_collect_state;
END IF;

IF quarter_in = '1' THEN
   total_amount := total_amount + 25;
   next_state = coin_collect_state;
END IF;

2: Coin collect state

IF nickel_in = '1' THEN
   total_amount := total_amount + 05;
   next_state = coin_collect_state;
END IF;

IF dime_in = '1' THEN
   total_amount := total_amount + 10;
   next_state = coin_collect_state;
END IF;

IF quarter_in = '1' THEN
   total_amount := total_amount + 25;
   next_state = coin_collect_state;
END IF;

IF return_money = '1' THEN
   next_state = return_money_state;
END IF;

IF total_amount > 95 THEN
   next_state = excess_money_state;
END IF;

IF (itemA_select = '1' OR itemB_select = '1' OR itemC_select = '1') THEN
   next_state = item_select_state;
END IF;

3: Item select state

1. item A is selected
IF ItemA_select = '1' THEN
   IF total_amount >= 55 THEN
      ItemA_out = '1';
      Change := total_amount – 55;
      Next_state = change_out_state;
   ELSE
      Next_state = coin_collect_state;
   END IF;
END IF;

2. item B is selected
IF ItemB_select = '1' THEN
   IF total_amount >= 65 THEN
      ItemB_out = '1';
      Change := total_amount – 65;
      Next_state = change_out_state;
   ELSE
      Next_state = coin_collect_state;
   END IF;
END IF;

3. item C is selected
IF ItemC_select = '1' THEN
   IF total_amount >= 75 THEN
      ItemC_out = '1';
      Change := total_amount – 75;
      Next_state = change_out_state;
   ELSE
      Next_state = coin_collect_state;
   END IF;
END IF;

4: Change out state

Suppose the change is 35 cents.
Following VHDL is used:

IF change = 35 THEN
   Nickel_out = '1';
   Dime_out = '1';
   Dimes_out = '1';
   Quarter_out = '0';
   total_amount := 0;
   Change := 0;
   Next_state = idle_state;
END IF;

5: Return money state

Return_money_out = '1';
Next_state = idle_state;

6: Excess money state

Excess_money_out = '1';
Next_state = coin_collect_state;

Process optimize
Approach II
Process Decomposition

Process optimize

1: idle state
2': coin/item state
4: return money state
5: excess money state
6: change maker state

CASE current_state IS
WHEN idle_state =>
  [idle_state implementation]
WHEN coin/item_state =>
  [coin/item_state implementation]
WHEN return_money_state =>
  [return_money_state implementation]
WHEN excess_money_state =>
  [excess_money_state implementation]
WHEN change_maker_state =>
  [change_maker_state implementation]
END CASE;

System Design and Clock design

Conclusions

- Both structural decomposition and process decomposition can be used in VLSI product design based on function analysis.
- Structural decomposition makes it easier for a designer to concentrate on solving problems at a subsystem level rather than considering the entire system. The function-input/output matrix helps us find the inner relationship of subsystems.
- Process decomposition provides a unique view of the process of the system. It allows one to identify the crucial path and bottlenecks of the process quickly. It is a key tool for restructuring processes. The working states of vending machine are clearly identified, so I use process decomposition method to design vending machine controller.