Multi-Level Programmable Array

20023179 Kim, Hyungock
Introduction

- Regular Structures
  - Why? Easy to P&R (almost no need to P&R)
  - Examples
    - PLA – like
    - Binary Tree base
    - Lattice Diagram
      - Better solution than UAA
      - UAA is treated as attempt to combine PLA-like and tree-like
This Presentation is composed as following

- Intro to Lattice Diagram
- MOPS for multiple-out Lattice Diagram
- Generalized architecture for MOPS
1. Intro to Lattice Diagram

- Characteristics
  - Like Tree and similar to BDD.
  - BDD has combined predecessors if and only if predecessors in the same level is equal.
  - But Lattice Diagram has always combine neighbor predecessors by some Rule. It occurs repetition of control variables.
  - Although BDD grows horizontally, Lattice grows vertically by the repetition of variables...
  - BDD and Lattice Diagram is made of MUX.
1. Intro to Lattice Diagram

- Combining Rule
  - Basic rule is the combining of two predecessors by XOR

(a AND P1-2) XOR (a' AND P2-1)

- More rule and method are introduced in “LATTICE DIAGRAMS USING REED-MULLER LOGIC” by Perkowski
Some problems in Lattice Diagram

- Repetition of control variable
  - It increases vertical depth.
  - This problem controlled by variable ordering.

- In the case of multi-output func
  - Ordering is not easy to be performed
  - There is quite waste for one block
  - And Partitions generate big empty subareas
    - Not good method, it leads to horizontal growth.
2. MOPS for multiple-out Lattice

- Functional Decomposition
  - Basic conception is to divide function to sub-functions
  - There are some decomposition methods
    - AND Decomposition, OR ~, Decomposition with Mux
  - Multi-output func can be decomposed by symmetric func
    - Multi-output func can be composed of Boolean operation (AND, OR, EXOR) of symmetric funcs.
    - Because of no repetition of variable in symmetric func, this method is very nice to reduce vertical depth.
2. MOPS for multiple-out Lattice

- What is symmetric func?
  - All minterms that have same number of ones in their binary number have same value (zero, or one).
  - Eg

\[
F(a,b,c,d) \quad \text{: It polarity 1111} = S3,4
\]

<table>
<thead>
<tr>
<th>a, b</th>
<th>00</th>
<th>01</th>
<th>11</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>01</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>11</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>10</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>
2. MOPS for multiple-out Lattice

- **MOPS for 4-variables**
  - MOPS is one diagram but it can express all symmetric func which has same polarity
  - So that reason, it reduces horizontal width compare to partition-method.

```
a------------------
b
  
c
  
d
  
S1 S2 S3 S4
```
2. MOPS for Multiple-out Lattice

Examples of using MOPS

- \[ F = (\sim b \text{ XOR } \sim d) \text{ OR } (a \text{ XOR } c) \text{ OR } (abcd) \]
  - It is decomposed to two symmetric functions \( S_3, S_4 \) that have same polarity
Every multi-output Boolean func can be decomposed to vector-OR of symmetric func of variable polarity

- Each MOPS has same control variable but different polarity
- Outputs of two MOPSes are combined in OR plane
3. Generalized architecture for MOPS

Every multi-output function with subset $SV_i$, $i = 1 \sim k$ of mutual symmetric variables can be decomposed to serial composition of $K$ MOPS arrays followed by AND/OR plane.

- $F(SV) = f_1(SV_1) \text{ OR } f_2(SV_2) \ldots \text{ OR } f_k(SV_k)$
- Each $f_i(SV_i)$ is symmetric, it can be expressed by one MOPS