Module 3: Behavioral Synthesis
Lecture 3.3: Major Steps – Binding/Assignment
Register Binding

\[ y = a + b + c + d \]

<table>
<thead>
<tr>
<th>Clock cycle</th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>+</td>
<td></td>
<td></td>
</tr>
<tr>
<td>b</td>
<td></td>
<td>+</td>
<td></td>
</tr>
<tr>
<td>c</td>
<td></td>
<td></td>
<td>+</td>
</tr>
<tr>
<td>d</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Registers Implied
Register Binding: Lifetime Analysis

Registers R1 & R2 can be shared

Clock cycle 1 2 3

(R3 needed for output latch)
Functional Unit Binding

\[ y = (a+b+c) \cdot (d+e) \]

Clock cycle 1

- \( a \) to \( + \) (op1)
- \( b \)

Clock cycle 2

- \( c \) to \( + \) (op3)
- \( d \) to \( + \) (op2)

Clock cycle 3

- \( e \)
- \( \cdot \) (op3)
- \( \ast \)
- \( y \)
## Functional Unit Binding Choices

<table>
<thead>
<tr>
<th>Operation</th>
<th>Binding 1</th>
<th>Binding 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Op1</td>
<td>Add1</td>
<td>Add1</td>
</tr>
<tr>
<td>Op2</td>
<td>Add2</td>
<td></td>
</tr>
<tr>
<td>Op3</td>
<td>Add1</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Operation</th>
<th>Binding 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Op1</td>
<td>Add1</td>
</tr>
<tr>
<td>Op2</td>
<td>Add2</td>
</tr>
<tr>
<td>Op3</td>
<td>Add2</td>
</tr>
</tbody>
</table>
Binding 2 Results

<table>
<thead>
<tr>
<th>Operation</th>
<th>Binding</th>
</tr>
</thead>
<tbody>
<tr>
<td>Op1</td>
<td>Add1</td>
</tr>
<tr>
<td>Op2</td>
<td>Add2</td>
</tr>
<tr>
<td>Op3</td>
<td>Add2</td>
</tr>
</tbody>
</table>

Diagram:

- Input variables: a, b, d, c, e
- Operation nodes:
  - Op1: Add1
  - Op2: Add2
  - Op3: Add2
- Intermediate nodes:
  - S1, S2, S3
- Output variable: y
A quick tour through behavioral synthesis

- Advanced techniques
  - Chaining and Multi-cycling
  - Pipelining
    - Using pipelined functional units
    - Pipelining the entire algorithm
- Transformations
  - Loop transformations
  - Arithmetic transformations
  - Control-flow transformations (e.g., speculation)
Multi-Cycling and Chaining

Combinational Logic Block → latch

CL1 → CL2 → latch
Pipelining

Combination Logic Block

latch

Combination Logic Block (1/3) -> latch -> Combination Logic Block (2/3) -> latch -> Combination Logic Block (3/3) -> latch
Loop Unrolling

- Increases the scope of optimization by exposing a larger sub-program to the other steps (scheduling, resource sharing)
- What happens to the control overhead when the loop is unrolled?

```
FOR i IN 0 to 3 LOOP
    result := result + S(i);
END LOOP;
```

\[
\text{result := result + } S(0) + S(1) + S(2) + S(3)
\]

1 adder, 4 cycles

1 adder, 4 cycles
OR
2 adders, 2 cycles
OR
4 adders, 1 cycle
Summary

• Behavioral Synthesis automatically generates an RTL implementation (architecture) from a behavioral specification (algorithm)

• Rich history of research
  – First attempts failed
  – Recent resurgence due to improved technology and higher need

• Basic steps in behavioral synthesis
  – Scheduling, resource sharing
References / Reading Material

The following books are available at POTR