ECE 720 – ESL & Physical Design

Lecture 13: Behavioral Synthesis

Spring 2013
W. Rhett Davis
NC State University
Announcements

- Homework #5 Due Today
- Project Advising Monday (no lecture)
  » Meet in regular classroom
- Class will not meet on Wed. next week
- Project 1 Milestone 1 Due in 1 week
- Project 1 Milestone 2 Due in 3 weeks
- Changes to RTL Testbench planned
Today’s Lecture

- Pareto Optimality
- Behavioral Synthesis with Catapult
  » Introduction
  » Loop Constraints
Example Design

How many ways are there to implement this function? How do they differ?
Architectural Parameters

- **Basic**
  - Delay: $NT_{cyc}$
  - Throughput: $1/T_{cyc}$

- **Pipelining**
  - Delay: $(N+P-1)T_{cyc}$
  - Throughput: $1/T_{cyc}$

- **Parallelism**
  - Delay: $(N-1)T_{cyc}/R + T_{cyc}$
  - Throughput: $R/T_{cyc}$

- **Hybrid**
  - Delay: $(N-1)T_{cyc}/R + PT_{cyc}$
  - Throughput: $R/T_{cyc}$

- Need to choose no. of operations N to define Delay
Pareto Optimality

- Multi-constraint Optimization is defined as "Pareto Sets" that are optimal according to one set of constraints.
Pipelining/Parallelism Variation

- Lines show the pareto optimal set with one parameter fixed
- To achieve smaller delay with minimum power, it's better to try pipelining first
- Parallelism helps once pipelining reaches the point of diminishing returns
Need for High-Level Synthesis

- RTL descriptions can be "Retimed" (see Leiserson & Saxe, *Algorithmica* 1991)
  - see Design Compiler `optimize_registers` command, which will move operators between pipeline stages in order to minimize cycle time
- But RTL synthesis cannot change cycle-to-cycle behavior. So, many optimizations are not possible with RTL synthesis
Retiming Algorithm Illustrated

- Original correlator function

\[ \delta(P,Q) = \begin{cases} 0 & \text{if } P \neq Q \\ 1 & \text{if } P = Q \end{cases} \]

- Retimed correlator has shorter critical path delay than the original

Today’s Lecture

- Pareto Optimality
- Behavioral Synthesis with Catapult
  » Introduction
  » Loop Constraints
Behavioral Synthesis

- Also called "High Level Synthesis"
- Logic Synthesis from behavioral descriptions, generally assumed to be higher level than RTL
- Developed in the 80's, around the same time as RTL synthesis
  - See papers by D. E. Thomas in IEEE Trans. CAD
  - Grew out of the RISC compilers community
- Commercial products have been available since the mid-90's
- Slow to catch on, because they make manual circuit optimization more difficult.
- Gaining popularity, because the need for increased design productivity is so great.
High-Level Synthesis Flow

- Set Architectural Constraints
- Allocate Resources
- Schedule
- Generate RTL
A Very Simple Design

- **Code**
  
  ```c
  #pragma design top
  void test(int a, int b, int &retval) {
    retval = a + b;
  }
  ```

- **Schedule**

- **Schematic**
"Gantt Chart" Schedule Explained

- Grey areas show non-time related events, such as data being registered.
- Clock period. Shaded area is Percent Sharing Allocation (user configurable).
- Width of real operations (rectangles) is proportional to their clock delay.
- Red bar shows the slack for the operation.

"Gantt Chart" Schedule Explained

Waveforms for this Schedule
Catapult Terminology

- **Latency** – Time (in cycles) from the first input to the first output
- **Throughput** – How often (in cycles) a function call can complete (most important quality measure)
- **Initiation Interval (II)** – How many clock cycles are taken before starting the next loop iteration (differs from common definition of II)
- **Source**: High Level Synthesis Blue Book © 2010 Mentor Graphics Corporation
### Synthesis Results

- **Constraints**
  - Clock Frequency = 333 MHz
- The metrics we care most about are Throughput and Area

<table>
<thead>
<tr>
<th>Process</th>
<th>Real Operation(s) count</th>
<th>Latency</th>
<th>Throughput</th>
<th>Reset Length</th>
<th>II</th>
</tr>
</thead>
<tbody>
<tr>
<td>/test/core</td>
<td>4</td>
<td>1</td>
<td>2</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Component Name</th>
<th>Area Score</th>
<th>Delay</th>
<th>Post Alloc</th>
<th>Post Assign</th>
</tr>
</thead>
<tbody>
<tr>
<td>mgc_add(32,0,32,0,32,4)</td>
<td>267</td>
<td>1.792</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>mgc_mux(32,1,2,3)</td>
<td>128</td>
<td>0.081</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>mgc_reg_pos(32,0,0,1,1,0,0,2)</td>
<td>636</td>
<td>0.121</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Total (After Assignment)</td>
<td>1030</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Inputs & Outputs

- I/O inferred from arguments

```c
#pragma design top
void test(int a, int b, int &retval) {
    retval=a+b;
}
```

- Pass-by-value considered input
- Pass-by reference
  - considered output if never assigned
  - considered inout if assigned
Today’s Lecture

- Pareto Optimality

- Behavioral Synthesis with Catapult
  » Introduction
  » Loop Constraints
#pragma design top

```c
void dotprod(int a_re[4], int a_im[4], int b_re[4],
             int b_im[4], int &z_re, int &z_im) {
    int sum_re = 0;
    int sum_im = 0;
    mult_acc: for (int i=0; i<4; i++) {
        sum_re = sum_re + a_re[i]*b_re[i]-a_im[i]*b_im[i];
        sum_im = sum_im + a_re[i]*b_im[i]+a_im[i]*b_re[i];
    }
    z_re = sum_re;
    z_im = sum_im;
}
```
Example Synthesis Results 1

- Constraints
  - Pipelined
  - Clock Frequency = 333 MHz

<table>
<thead>
<tr>
<th>Process</th>
<th>Real Operation(s) count</th>
<th>Latency</th>
<th>Throughput</th>
<th>Reset Length</th>
<th>II</th>
</tr>
</thead>
<tbody>
<tr>
<td>/dotprod/core</td>
<td>26</td>
<td>5</td>
<td>7</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Component Name</th>
<th>Area Score</th>
<th>Delay</th>
<th>Post Alloc</th>
<th>Post Assign</th>
</tr>
</thead>
<tbody>
<tr>
<td>mgc_add(32,0,32,0,32,3)</td>
<td>440</td>
<td>0.915</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>mgc_add(33,0,33,0,33,3)</td>
<td>454</td>
<td>0.945</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>mgc_mul(32,0,32,0,32,4)</td>
<td>4261</td>
<td>2.191</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>mgc_reg_pos(32,0,0,1,1,0,0,2)</td>
<td>636</td>
<td>0.121</td>
<td>0</td>
<td>8</td>
</tr>
<tr>
<td>Total (After Assignment)</td>
<td>26186</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
## Schedule

### Loop Hierarchy

```plaintext
main (core)
- mult_acc
  - mult_accio_read(a_re:rscl)
  - mult_accmux
  - mult_accio_read(b_im:rscl)
  - mult_accmux#1
  - mult_accio_read(a_im:rscl)
  - mult_accmux#2
  - mult_accio_read(b_re:rscl)
  - mult_accmux#3
  - mult_accmul
  - mult_accmul#1
  - mult_accmul#2
  - mult_accmul#3
  - mult_accacc#13
  - mult_accacc
  - mult_accacc#14
  - mult_accacc#5
  - mult_accacc#11
- io_write(z_re:rscl.d)
- io_write(z_im:rscl.d)
- io_sync(a_re:triosy)
- io_sync(b_re:triosy)
- io_sync(a_im:triosy)
- io_sync(b_im:triosy)
```

### Scheduled Operations

<table>
<thead>
<tr>
<th>C0</th>
<th>C1</th>
<th>C2</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

![Diagram showing scheduled operations]
Timing Diagram

- Multiply and Add are pipelined
- Cycles are needed to get into loop and exit the procedure
RTL Schematic

- 4 multipliers
- 4 pipeline registers
- 2 adders
- 2 pipeline registers
- 2 adders
### Example Synthesis Results 2

#### Constraints

- Pipelined
- Clock Frequency = 667 MHz

<table>
<thead>
<tr>
<th>Process</th>
<th>Real Operation(s) count</th>
<th>Latency</th>
<th>Throughput</th>
<th>Reset Length</th>
<th>II</th>
</tr>
</thead>
<tbody>
<tr>
<td>/dotprod/core</td>
<td>26</td>
<td>6</td>
<td>8</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Component Name</th>
<th>Area Score</th>
<th>Delay</th>
<th>Post Alloc</th>
<th>Post Assign</th>
</tr>
</thead>
<tbody>
<tr>
<td>mgc_add(32,0,32,0,32,3)</td>
<td>440</td>
<td>0.915</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>mgc_add(33,0,33,0,33,3)</td>
<td>454</td>
<td>0.945</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>mgc_mul(32,0,32,0,32,4)</td>
<td>4764</td>
<td>1.059</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>mgc_reg_pos(32,0,0,1,1,0,0,2)</td>
<td>636</td>
<td>0.121</td>
<td>0</td>
<td>10</td>
</tr>
<tr>
<td>Total (After Assignment)</td>
<td>29477</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Timing Diagram

cycle 1 2 3 4 5 6 7 8 1 2 3

a,b

i=0 i=1 i=2 i=3

a_0,b_0 a_1,b_1

start new op

mult

a_0[0],b_0[0] sampled

add0,1

i=0 i=1 i=2 i=3

a_0[1],b_0[1] sampled

add2,3

z

i=0 i=1 i=2 i=3

z_0 available

start new op

a_1[0],b_1[0] sampled
Loop Unrolling

- By default, each iteration of a loop takes at least 1 cycle

- In order to execute multiple loop iterations per cycle (i.e. parallelism), need to specify that loops should be "unrolled"
Example Synthesis Results 3

- **Constraints**
  - Pipeline, Partially Unroll by 2 (execute iterations in parallel)
  - Clock Frequency = 667 MHz

<table>
<thead>
<tr>
<th>Process</th>
<th>Real Operation(s)</th>
<th>Latency</th>
<th>Throughput</th>
<th>Reset Length</th>
<th>II</th>
</tr>
</thead>
<tbody>
<tr>
<td>/dotprod/core</td>
<td>count</td>
<td>39</td>
<td>4</td>
<td>6</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Component Name</th>
<th>Area Score</th>
<th>Delay</th>
<th>Post Alloc</th>
<th>Post Assign</th>
</tr>
</thead>
<tbody>
<tr>
<td>mgc_add(32,0,32,0,32,1)</td>
<td>1146</td>
<td>0.200</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>mgc_add(32,0,32,0,32,2)</td>
<td>495</td>
<td>0.509</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>mgc_add(33,0,33,0,33,2)</td>
<td>510</td>
<td>0.525</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>mgc_add(33,0,33,0,33,3)</td>
<td>454</td>
<td>0.945</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>mgc_mul(32,0,32,0,32,4)</td>
<td>4764</td>
<td>1.059</td>
<td>8</td>
<td>8</td>
</tr>
<tr>
<td>mgc_reg_pos(32,0,0,1,1,0,0,2)</td>
<td>636</td>
<td>0.121</td>
<td>0</td>
<td>16</td>
</tr>
<tr>
<td><strong>Total (After Assignment)</strong></td>
<td><strong>55761</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Schedule

- Note: main cycle C2 was cut off from this image
Waveforms

cycle | 1 | 2 | 3 | 4 | 5 | 6 | 1 | 2 | 3
--- | --- | --- | --- | --- | --- | --- | --- | --- | ---

| a,b | a₀,b₀ | a₁,b₁ |
--- | --- | --- |

| mult | i=0,1 | i=2,3 |
--- | --- | --- |

| add0,1 | i=0,1 | i=2,3 |
--- | --- | --- |

| add2,3 | i=0,1 | i=2,3 |
--- | --- | --- |

| z | | |
--- | --- | --- |

start new op

a₀[0],b₀[0] sampled

a₀[1],b₀[1] sampled

z₀ available

start new op

a₁[0],b₁[0] sampled
Manual Loop Unrolling

- Can get the exact same result as Ex. 3 by altering the code as shown below and using same constraints from Ex. 2

```c
#pragma design top
void dotprod(int a_re[4], int a_im[4], int b_re[4],
             int b_im[4], int &z_re, int &z_im) {
    int sum_re = 0;
    int sum_im = 0;
    mult_acc: for (int i=0; i<4; i+=2) {
        sum_re = sum_re + a_re[i]*b_re[i]-a_im[i]*b_im[i];
        sum_re = sum_re + a_re[i+1]*b_re[i+1]-a_im[i+1]*b_im[i+1];
        sum_im = sum_im + a_re[i]*b_im[i]+a_im[i]*b_re[i];
        sum_im = sum_im + a_re[i+1]*b_im[i+1]+a_im[i+1]*b_re[i+1];
    }
    z_re = sum_re;
    z_im = sum_im;
}
```
### Example Synthesis Results 4

- **Constraints**
  - Unroll Fully
  - Clock Frequency = 1000 MHz

<table>
<thead>
<tr>
<th>Process</th>
<th>Real Operation(s)</th>
<th>Latency</th>
<th>Throughput</th>
<th>Reset Length</th>
<th>II</th>
</tr>
</thead>
<tbody>
<tr>
<td>/dotprod/core</td>
<td>count</td>
<td>46</td>
<td>19</td>
<td>20</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Component Name</th>
<th>Area Score</th>
<th>Delay</th>
<th>Post Alloc</th>
<th>Post Assign</th>
</tr>
</thead>
<tbody>
<tr>
<td>mgc_add(32,0,32,0,32,3)</td>
<td>495</td>
<td>0.509</td>
<td>1</td>
<td>5</td>
</tr>
<tr>
<td>mgc_add(33,0,33,0,33,3)</td>
<td>510</td>
<td>0.525</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>mgc_mul(32,0,32,0,32,4)</td>
<td>6154</td>
<td>0.719</td>
<td>1</td>
<td>10</td>
</tr>
<tr>
<td>mgc_reg_pos(32,0,0,1,1,0,0,2)</td>
<td>636</td>
<td>0.121</td>
<td>0</td>
<td>29</td>
</tr>
<tr>
<td>mgc_reg_pos(128,0,0,1,1,0,0,2)</td>
<td>2543</td>
<td>0.121</td>
<td>0</td>
<td>4</td>
</tr>
<tr>
<td>Total (After Assignment)</td>
<td>100028</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Schedule

- Not sure what's going on here… doesn't seem to be successfully unrolling. Perhaps I'm doing something wrong?
Example Synthesis Results 5

- **Constraints**
  - Not Pipelined
  - Clock Frequency = 667 MHz

<table>
<thead>
<tr>
<th>Process</th>
<th>Real Operation(s) count</th>
<th>Latency</th>
<th>Throughput</th>
<th>Reset Length</th>
<th>II</th>
</tr>
</thead>
<tbody>
<tr>
<td>/dotprod/core</td>
<td>26</td>
<td>28</td>
<td>30</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Component Name</th>
<th>Area Score</th>
<th>Delay</th>
<th>Post Alloc</th>
<th>Post Assign</th>
</tr>
</thead>
<tbody>
<tr>
<td>mgc_add(32,0,32,0,32,3)</td>
<td>440</td>
<td>0.915</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>mgc_add(33,0,33,0,33,3)</td>
<td>454</td>
<td>0.945</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>mgc_mul(32,0,32,0,32,4)</td>
<td>4764</td>
<td>1.059</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>mgc_reg_pos(32,0,0,1,1,0,0,2)</td>
<td>636</td>
<td>0.121</td>
<td>0</td>
<td>15</td>
</tr>
<tr>
<td>Total (After Assignment)</td>
<td>28264</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
What are we doing here?

- By turning off pipelining, we target an architecture much more like a processor, with single function units.

- Because the clock frequency is so high, however, we are unable to achieve the desired frequency without additional function units.
### Example Synthesis Results 6

- **Constraints**
  - Not Pipelined
  - Clock Frequency = 333 MHz

<table>
<thead>
<tr>
<th>Process</th>
<th>Real Operation(s)</th>
<th>Latency</th>
<th>Throughput</th>
<th>Reset Length</th>
<th>II</th>
</tr>
</thead>
<tbody>
<tr>
<td>/dotprod/core</td>
<td>26</td>
<td>20</td>
<td>22</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Component Name</th>
<th>Area Score</th>
<th>Delay</th>
<th>Post Alloc</th>
<th>Post Assign</th>
</tr>
</thead>
<tbody>
<tr>
<td>mgc_add(32,0,32,0,32,3)</td>
<td>494</td>
<td>0.509</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>mgc_add(33,0,33,0,33,3)</td>
<td>1183</td>
<td>0.201</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>mgc_mul(32,0,32,0,32,4)</td>
<td>4710</td>
<td>1.419</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>mgc_reg_pos(32,0,0,1,1,0,0,2)</td>
<td>636</td>
<td>0.121</td>
<td>0</td>
<td>12</td>
</tr>
<tr>
<td>Total (After Assignment)</td>
<td>17741</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Schedule
Pareto-Optimal Designs

- Delay (ns)
- Area