ELEC6233
Digital System Synthesis

Introduction to high-level synthesis steps

Coursework 1 briefing
Typical high-level synthesis steps

1. High-level spec in C++/SystemC or other
2. Intermediate representation, e.g. CDFG
3. Optimise CDFG
4. Perform scheduling and determine resources (order in which operations execute)
5. Perform binding (which ops execute on which resources)
6. Generate RTL model

Optimisations are performed at each step
Example – high-level synthesis from C code by hand

```c
// shift-and-add multiplication
// P = M * Q, n –bits
P = 0;
for(i = 0; i<n; i++){
    if (Q[i] == 1)
        P = P+M;
    P = P << 1;
}
```

Step 1: Build CFG for controller (an FSM)

- **P=0, i=0**
- **i<n**
- **Q[i]==1**
  - **P += M**
  - **P <<= 1**
  - **i++**
- **Done**
Example – high-level synthesis from C code by hand

Step 2: Determine datapath resources

// shift-and-add multiplication
// P = M *Q, n –bits
P = 0;
for(i = 0; i<n; i++){  
  if (Q[i] ==1)  
    P = P+M;
  P = P << 1;
}

\[
\begin{align*}
\text{P}=0, \text{i}=0 \\
\text{i}<n \\
\text{Q}[i]==1 \\
\text{P}+=\text{M} \\
\text{P}<<=1 \\
\text{i}++
\end{align*}
\]

\[
\begin{align*}
\text{Q}[i] & \rightarrow \text{MUX 1 of n} \\
\text{P} & \rightarrow \text{shift-and-add multiplication} \\
\text{M} & \rightarrow \text{+}
\end{align*}
\]
Step 3: Determine datapath register inputs

// shift-and-add multiplication
// P = M * Q, n -bits
P = 0;
for(i = 0; i<n; i++){
    if (Q[i] == 1)
        P = P+M;
    P = P << 1;
}

Q[i] MUX 1 of n

Q

// shift-and-add multiplication
// P = M * Q, n -bits
P = 0;
for(i = 0; i<n; i++){
    if (Q[i] == 1)
        P = P+M;
    P = P << 1;
}

// shift-and-add multiplication
// P = M * Q, n -bits
P = 0;
for(i = 0; i<n; i++){
    if (Q[i] == 1)
        P = P+M;
    P = P << 1;
}
Example – high-level synthesis from C code by hand

// shift-and-add multiplication
// P = M *Q, n –bits
P = 0;
for(i = 0; i<n; i++){
    if (Q[i] ==1)
        P = P+M;
    P = P << 1;
}

Step 4: Add control signals

\[
P = 0, \quad i = 0
\]
\[
i < n
\]
\[
Q[i] == 1
\]
\[
P += M
\]
\[
P <= 1
\]
\[
i++
\]
Step 5: Combine control and datapath

// shift-and-add multiplication
// \( P = M \times Q \), \( n \) -bits
\[
P = 0;
\text{for}(i = 0; i < n; i++){
\text{if}(Q[i] == 1)\{
    P = P + M;
    P = P \ll 1;
\}
}\]
Automated high-level synthesis

• Determine when to perform each operation
  – Scheduling
  – Consider alternatives and optimise e.g. for minimum latency

• Allocate resources for each operation
  – Resource allocation
  – Consider alternatives and optimise, for performance, H/W size, etc.

• Map operations onto resources
  – Binding
Scheduling

• Scheduling assigns a start time to each operation in DFG
  – Start times must not violate dependencies in DFG
  – Start times must meet performance constraints
    • Also, resource constraints

• Performed on the DFG at each CFG node (each state)
  – Difficult to process multiple CFG nodes in parallel
Examples
Typical problems in scheduling

- Several types of scheduling issues
  - Usually some trade-off of performance and resource constraints

- Issues:
  - Unconstrained scheduling
    - Not very useful, every schedule is valid
  - Minimum latency scheduling – can be found through optimisation
  - Latency constrained scheduling
  - Resource constrained scheduling
  - Minimum-latency, resource constrained scheduling
    - i.e. find the schedule with the shortest latency, that uses less than a specified # of resources
    - NP-Complete problem!
  - Minimum-resource, latency constrained scheduling
    - i.e. find the schedule that meets a given latency constraint, and uses the minimum # of resources
    - NP-Complete!
Minimum Latency Scheduling

• ASAP (as soon as possible) algorithm
  – Find a candidate node in DFG
    • Candidate is a node whose predecessors have been scheduled (or there are no predecessors)
  – Schedule node one cycle later than max cycle of predecessor
  – Repeat until all nodes scheduled

\[
\begin{array}{ccccccccc}
a & b & c & d & e & f & g & h \\
\hline
\text{Cycle 1} & + & + & - & < \\
\text{Cycle 2} & \ast & \ast & \ast & \ast \\
\text{Cycle 3} & + \\
\text{Cycle 4} & + \\
\end{array}
\]

Minimum latency - 4 cycles
Minimum Latency Scheduling

• ALAP (as late as possible) algorithm
  – Run ASAP, get minimum latency L
  – Find a candidate node in DFG
    • Candidate is node whose successors are scheduled (or there are none)
  – Schedule node one cycle before start cycle of successor
    • Nodes with no successors scheduled to last cycle L
  – Repeat until all nodes scheduled

L = 4 cycles
Minimum Latency Scheduling

- ALAP (as late as possible) algorithm
  - Run ASAP, get minimum latency $L$
  - Find a candidate node in DFG
    - Candidate is node whose successors are scheduled (or there are none)
  - Schedule node one cycle before start cycle of successor
    - Nodes with no successors scheduled to last cycle $L$
  - Repeat until all nodes scheduled

$L = 4$ cycles
Latency-Constrained Scheduling

- Instead of finding the minimum latency, find latency at most that of a given constraint L
  - Solutions:
    - Use ASAP, verify that minimum latency is at most L
    - Use ALAP starting with cycle L instead of minimum latency (no need to run ASAP first, as L is given)
Scheduling with Resource Constraints

• Schedule must use less than specified number of resources

Constraints: 1 ALU (+/-), 1 Multiplier
Scheduling with Resource Constraints

• Schedule must use less than specified number of resources

Constraints: 2 ALU (+/-), 1 Multiplier
Minimum-Latency, Resource-Constrained Scheduling

• Given resource constraints, find a schedule that has the minimum latency
  – Example:

Constraints: 1 ALU (+/-), 1 Multiplier
Minimum-Latency, Resource-Constrained Scheduling

- Given resource constraints, find schedule that has the minimum latency
  
  – Example:

Constraints: 1 ALU (+/-), 1 Multiplier

Different schedules may use same resources, but have different latencies
Minimum-Latency, Resource-Constrained Scheduling

• Hu’s Algorithm

• Assumes that all resources are of one type

• Basic Idea
  – Input: Data Flow Graph G, resource constraint r (number of resources)
  – 1. Label each node in G by the longest path passing through it
  – 2. initialise cycle count: L = 1
  – 3. REPEAT
    3.1. Determine the subset U of unscheduled nodes in G whose predecessors have been scheduled or there are no predecessors
    3.2. Select subset S of U such that |S| <= r and labels in S are largest
    3.3. Schedule the S nodes at cycle L
    3.4. L ++
  UNTIL all nodes in G are scheduled
Hu’s algorithm - example

Assume resource constraint $r = 3$

Longest path nodes scheduled at cycle 1: A, B, C
Hu’s algorithm - example

Nodes scheduled at cycle 2: F, G, D
Hu’s algorithm - example

Nodes scheduled at cycle 3: J, H, E
Hu’s algorithm - example

Nodes scheduled at cycle 4: K, I
Summary

• Scheduling assigns each operation in a DFG a start time
  – Done for each DFG in the CDFG

• Types of scheduling algorithms
  – Minimum Latency
    • ASAP, ALAP
  – Latency-constrained
    • ASAP, ALAP
  – Minimum-latency, resource-constrained
    • Hu’s Algorithm
    • Other algorithms will be considered later
  – Minimum-resource, latency-constrained
    • Will be considered later