Key Takeaways from Lecture

Register Transfer Level

List Processor

Scheduling
Register Transfer Level

\[
\text{in1} \rightarrow \begin{array}{c}
\text{CL} \\
\text{in2}
\end{array} \rightarrow \begin{array}{c}
\text{reg} \\
\text{in2}
\end{array} \rightarrow \begin{array}{c}
\text{CL} \\
\text{out}
\end{array} \\
\rightarrow \begin{array}{c}
\text{reg} \\
\text{out}
\end{array}
\]

- verilog
- VHDL
Register Transfer Level vs HLS (high level synthesis)

HLS: C, C++, VHDL

HLS

Pros: quick implementation, less error prone
Cons: less efficient/optimized

Pros: lower level design choices to optimize design
Cons: development time is longer
List Processor

\[
\begin{align*}
\text{IF} & \ (\text{START} == 1) \ \text{NEXT} \gets 0, \ \text{SUM} \gets 0, \ \text{DONE} \gets 0; \\
\text{REPEAT} & \ \{ \ \text{SUM} \gets \text{SUM} + \text{MEM}[\text{NEXT} + 1]; \\
& \ \ \ \text{NEXT} \gets \text{MEM}[\text{NEXT}]; \\
& \ \ \ \text{UNTIL} \ (\text{NEXT} == 0); \\
\& \ \text{R} \gets \text{SUM}, \ \text{DONE} \gets 1; \\
\end{align*}
\]
List Processor

\[
\text{numA - addr of num to add to next\numA,}\text{sum to next memory}\]

\[
\text{if (status = 1) next = 0, sum to next memory}\]

\[
\text{numA+1, sum = sum + memory[next+1, next = memory[next+1]}\]

\[
\text{ stripping Done, Done = 1}\]

\[
p \gets \text{sum, Done = 1}\]
Scheduling

unoptimized

memory
adder 1
adder 2

optimized

memory
adder

1
load x
t
sum

2
load next
t
num a

3
fetch
next +1
sum

4
load next
t
num a

x ← memory [NEXT +1]
Scheduling

(Module Scheduling)

1) calculate minimum length of characteristic section
   (max # of cycles that any one resource is used during
   one iteration)

2) schedule one iteration of the computation of characteristic
   section

3) if iteration doesn’t fit in minimal length section,
   increase section by 1 and try again
Scheduling

(Example)

read

2 memory ports

1 adder (3 input)

mem 1 load A | load C
mem 2 load B | load C
add

$E = B$

$E = A + B$

store $E$

$E = E$

$E = (A + B) + (C + D)$
Homework Questions