

CS-472: Design Technologies for Integrated Systems

Exercise Problem Set 3

Date: 26/09/2024

Topics: Control-flow expressions, finite state machines (cf. slide set 3); data flow optimization (cf. slide set 4)

Problem 1

Given the following three pieces of HDL pseudo-code:

```
while (a)
{
    while (b)
    {
        if (c)
            P1
        else
            P2
    }
    P3
}
```

```
wait (d)
{
    while (!e)
        P4
}
```

```
always
{
    if (f)
        P5
}
```

Write the control-flow expression that executes the three codes *in parallel*. Use parentheses properly for nested expressions to avoid ambiguity.

Problem 2

For the following specifications, draw a deterministic finite state machine that implements it. The machines receive a non-empty bit stream $(b_0, b_1, \dots, b_{n-1})$ as input, where $b_i \in \{0, 1\} \forall i$, and output $o \in \{0, 1\}$ which may change upon receiving a new input bit. The length n of the stream can be any finite positive integer and is unknown to the machine. The machines need to be correct for all n seen during the process as if the input bits received so far is the entire bit stream. The output before receiving any input, if any (i.e. if it is a Moore machine), can be ignored.

Let $\#0$ denote the number of 0 bits in the stream, and let $\#1$ denote the number of 1 bits in the stream. We say a FSM *accepts* a stream if it outputs $o = 1$ when the stream is completely fed into the machine.

Recall that a finite state machine must have a *finite* number of states, and the number of states cannot be an unknown number. Also note that for a FSM to be *deterministic*, the transition under every possible input at every reachable state must be specified and unique. Some machines may be impossible to construct; explain why if you think so.

- (a) Construct a FSM which implements the AND operation, i.e., $o = b_0 \wedge b_1 \wedge \dots \wedge b_n$.
- (b) Construct a FSM which implements the XOR operation, i.e., $o = b_0 \oplus b_1 \oplus \dots \oplus b_n$.
- (c) Construct a FSM which accepts a stream where $\#0 = \#1$.
- (d) Construct a FSM which accepts a stream where $(\#0 \bmod 3) = (\#1 \bmod 3)$.

Problem 3

Given the following equations:

$$\begin{aligned}
 x_1 &= (a \times 3); \\
 x_2 &= b; \\
 x_3 &= (5 + c); \\
 x_4 &= (d + e); \\
 x &= (x_1 \times x_2) + 3 + x_3 + (x_4 \times 2);
 \end{aligned}$$

where inputs are a, b, c, d, e , while the output is x .

- (a) Apply variable and constant propagation, and draw the data-flow graph. Assume all additions and multiplications have 2 inputs.
- (b) Apply *operator strength reduction* on the data-flow graph from point (a). Assume that shifting takes no latency. Draw the resulting data-flow graph.

Problem 4

Consider the following equations:

$$\begin{aligned}
 x &= (a \times b \times c + d) \times e \\
 w &= x + f \\
 y &= i + g \times h + (g \times d \times c \times i) \\
 z &= w + y
 \end{aligned}$$

where inputs are $a, b, c, d, e, f, g, h, i$, while the output is z .

- (a) Draw the data-flow graph using the operations as they appear in the expression, without any optimization. Assume additions and multiplications have 2 inputs.
- (b) Apply *tree-height reduction* to the data-flow graph drawn in (a).
- (c) Discuss on the different resources usage between the graph in (a) and the graph in (b).
- (d) Assume that $a = 4$, $b = 2$, $c = 3$, $h = 2$, and $i = 4$ are constant. Optimize the data-flow graph from (a) using the behavioral optimization techniques seen during the lecture. Draw the resulting data-flow graph.