EENG 426/CPSC 459/ENAS 876
Silicon Compilation

Production rule synthesis

Computer Systems Lab

http://csl.yale.edu/~rajit

Fall 2018

Yale AVLSI

Convert HSE to PRS

Transforms a handshaking expansion into a set of production rules.

- state assignment
- guard strengthening
- symmetrization

May have to reshuffle to improve production rules!

Yale AVLSI

Convert HSE to PRS

CHP:

$$*[[\overline{L} \longrightarrow R; L]]$$

Handshaking:
L: passive, since it is probed.
R: active, since matches passive L

$$*[[li]; ro\uparrow; [ri]; ro\downarrow; [\neg ri]; lo\uparrow; [\neg li]; lo\downarrow]$$

Yale AVLSI

Syntactic guards

We begin with a production rule set that is syntactically derived from the original program:

$$\begin{aligned} li &\mapsto ro\uparrow \\ ri &\mapsto ro\downarrow \\ \neg ri &\mapsto lo\uparrow \\ \neg li &\mapsto lo\downarrow \end{aligned}$$

Each action is guarded by the wait immediately before it.

If the handshaking expansion is deadlock-free, it is always possible to execute the syntactic production rules in program order.

Yale AVLSI

Syntactic guards

Other execution orders may be possible!

Example:

$\neg ri \mapsto lo\uparrow$
$\neg li \mapsto lo\downarrow$

can fire in the initial state.

Yale AVLSI

Strengthening

To prevent incorrect firings, we must reduce the number of states in which a production rule can fire by strengthening the guard.

The guard must be strong enough to uniquely identify the state(s) of the handshaking expansion in which the rule must fire.

Where can $ri \mapsto lo\uparrow$ fire?

Yale AVLSI

Systematic approach

State of the circuit as a vector (li, lo, ri, ro):

$*[\{X000\}[li]; \{1000\}ro\uparrow; \{10X1\}[ri]; \{1011\}ro\downarrow;$
$\{10X0\}[\neg ri]; \{1000\}lo\uparrow; \{X100\}[\neg li]; \{0100\}lo\downarrow$
$]$

Environment:

$*[\ li\uparrow; [lo]; li\downarrow; [\neg lo]\ ]$
$\|$
$*[\ [ro]; ri\uparrow; [\neg ro]; ri\downarrow\ ]$

Yale AVLSI

Systematic approach

States in which $\neg ri \mapsto lo\uparrow$ can fire:

$*[\bullet[li]; \bullet ro\uparrow; \bullet[ri]; ro\downarrow; \bullet[\neg ri]; \bullet lo\uparrow; \bullet[\neg li]; \bullet lo\downarrow]$

States in which $\neg ri \mapsto lo\uparrow$ has effective firings:

$*[\bullet[li]; \bullet ro\uparrow; \bullet[ri]; ro\downarrow; \bullet[\neg ri]; \bullet lo\uparrow; [\neg li]; lo\downarrow]$

States in which $\neg ri \mapsto lo\uparrow$ has undesirable effective firings:

$*[\bullet[li]; \bullet ro\uparrow; \bullet[ri]; ro\downarrow; [\neg ri]; lo\uparrow; [\neg li]; lo\downarrow]$

Yale AVLSI

Systematic approach

States in which $\neg ri \mapsto lo\uparrow$ has undesirable effective firings or could pcause interference (conflicting set):

$*[\bullet[li]; \bullet ro\uparrow; \bullet[ri]; ro\downarrow; [\neg ri]; lo\uparrow; [\neg li]; \bullet lo\downarrow]$

States in which $\neg ri \mapsto lo\uparrow$ must fire:

$*[[li]; ro\uparrow; [ri]; ro\downarrow; \bullet[\neg ri]; \bullet lo\uparrow; [\neg li]; lo\downarrow]$

$\Rightarrow$ select guard so that this is the case.

Yale AVLSI

Guard strengthening

Consider $B \mapsto x\uparrow$

HSE:

$\ldots;\ x\uparrow;\ \ldots;\ x\downarrow;\ \ldots;\ x\uparrow;$

firing set: set of states in which the rule could fire. (Determined by B)

disallowed set: set of states in which the production rule firing is not allowed because of interference or violation of HSE

Conflicting set: intersection of firing and disallowed set, should be empty!

Yale AVLSI

State variables

Sometimes it is not possible to identify each state uniquely using the variables we have in the handshaking expansion.

$\ldots.;\ xo\uparrow; [xi]; xo\downarrow; [\neg xi];\ \ldots$

Solution: introduce a new variable that has different values in the two indistinguishable states.

There are several places where the assignment to the state variable can be inserted.

Yale AVLSI

State variables

Handshaking: vector (li, lo, ri, ro)

$*[\{X000\}[li]; \{1000\}ro\uparrow; \{10X1\}[ri]; \{1011\}ro\downarrow;$
$\{10X0\}[\neg ri]; \{1000\}lo\uparrow; \{X100\}[\neg li]; \{0100\}lo\downarrow$
$]$

After state-variable insertion: vector (x, li, lo, ri, ro)

$x\downarrow;$
$*[\{0X000\}[li]; \{01000\}ro\uparrow; \{010X1\}[ri]; \{01011\}x\uparrow;$
$\{11011\}ro\downarrow; \{110X0\}[\neg ri]; \{11000\}lo\uparrow; \{1X100\}[\neg li];$
$\{10100\}x\downarrow; \{00100\}lo\downarrow$
$]$

Yale AVLSI

Production rule generation

HSE:

$$
\begin{array}{l}
x{\downarrow};\\
*[[li];ro{\uparrow};[ri];x{\uparrow};ro{\downarrow};[\neg ri];lo{\uparrow};[\neg li];x{\downarrow};lo{\downarrow}]
\end{array}
$$

PRS:

$$
\begin{array}{rcl}
\neg x \wedge li & \mapsto & ro{\uparrow}\\
ri & \mapsto & x{\uparrow}\\
x & \mapsto & ro{\downarrow}\\
x \wedge \neg ri & \mapsto & lo{\uparrow}\\
\neg li & \mapsto & x{\downarrow}\\
\neg x & \mapsto & lo{\downarrow}
\end{array}
$$

Yale

AVLSI

Symmetrization

$$
\begin{array}{rcl}
x \wedge \neg ri & \mapsto & lo{\uparrow}\\
\neg x & \mapsto & lo{\downarrow}
\end{array}
$$

Turn into combinational logic:

$$
\begin{array}{rcl}
ri \vee \neg x & \mapsto & lo{\downarrow}\\
\neg li \vee x & \mapsto & ro{\downarrow}
\end{array}
$$

Why is this legal?

Yale

AVLSI

Symmetrization

Replacing a state-holding operator with a combinational one:

$$
\begin{array}{rcl}
x \wedge \neg B & \mapsto & z{\uparrow}\\
B & \mapsto & z{\downarrow}
\end{array}
$$

If B holds as a precondition of $\neg x$, we can replace the second rule with:

$$\neg x \vee B \mapsto z{\downarrow}$$

We must ensure that no new effective firings have been introduced, i.e.,

$$x \vee B \vee \neg z$$

is invariant.

Yale

AVLSI

Operator reduction

The last step consists of grouping together production rules into operators, and identifying standard operators in the production rule set.

$$
\begin{array}{rcl}
li \wedge ri & \mapsto & x{\uparrow}\\
\neg ri \wedge \neg li & \mapsto & x{\downarrow}
\end{array}
$$

$$
\begin{array}{rcl}
\neg x \wedge li & \mapsto & ro{\uparrow}\\
\neg li \vee x & \mapsto & ro{\downarrow}
\end{array}
$$

$$
\begin{array}{rcl}
x \wedge \neg ri & \mapsto & lo{\uparrow}\\
ri \vee \neg x & \mapsto & lo{\downarrow}
\end{array}
$$

Yale

AVLSI