# EENG 426/CPSC 459/ENAS 876
# Silicon Compilation

**Non-deterministic selections**

Computer Systems Lab

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

Fall 2018

Yale AVLSI



## Arbitration

Consider the following CHP program:

$$
\begin{array}{l}
*[[\overline{X} \longrightarrow X?x \\
\;\; | \overline{Y} \longrightarrow Y?x \\
\;\; ]; \\
\;\; Z!x \\
]
\end{array}
$$

When $\overline{X}$ and $\overline{Y}$ are both **true**, we have to pick one of them and execute the appropriate branch of the selection statement.

Arbitration is the mechanism that picks one of two alternatives, deciding which alternative came "first."

Yale AVLSI



## Arbiters

An **arbiter** is the following process:

$$
\begin{array}{ll}
Arb(a,b,u,v) \equiv & *[[a \longrightarrow u\uparrow; [\neg a]; u\downarrow \\
& \;\; | b \longrightarrow v\uparrow; [\neg b]; v\downarrow \\
& ]]
\end{array}
$$

The process does a handshake on $(a,u)$ and $(b,v)$.
Suppose we try and write production rules:

$$
\begin{array}{lcl}
a \wedge \neg v & \mapsto & u\uparrow \\
\neg a \vee v & \mapsto & u\downarrow
\end{array}
$$

$$
\begin{array}{lcl}
b \wedge \neg u & \mapsto & v\uparrow \\
\neg b \vee u & \mapsto & v\downarrow
\end{array}
$$

Yale AVLSI



## Arbiters

To make the circuit directly implementable, we flip the sense of variables $u$ and $v$.

$$
\begin{array}{rcl}
a \wedge \_v & \mapsto & \_u\downarrow \\
\neg a \vee \neg\_v & \mapsto & \_u\uparrow
\end{array}
$$

$$
\begin{array}{rcl}
b \wedge \_u & \mapsto & \_v\downarrow \\
\neg b \vee \neg\_u & \mapsto & \_v\uparrow
\end{array}
$$

$\Rightarrow$ cross-coupled NAND gates.

What happens if both $a$ and $b$ go up at the same time?

Yale AVLSI

## Arbiters

The signals will separate eventually; however, we don't know how long it will take. It is impossible to have a circuit that decides which input switched first in bounded time.

$$\Pr[time \geq t] = Ae^{-t/\tau_0}$$

Note: the average time taken for signals to separate is bounded.

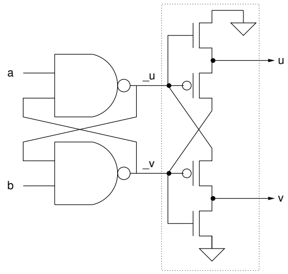
Since our circuits are asynchronous, we can wait until the signals separate.

Yale AVLSI



## Arbiters

The output of the cross-coupled NAND gate is connected to a filter circuit that waits for the signals to be separated by a threshold voltage.




(Note that the CMOS circuit is indeed weakly fair!)

Yale AVLSI



## Arbitration

Simple example:

$$
\begin{array}{l}
*[[\overline{A} \longrightarrow X; A \\
\;\;|\,\overline{B} \longrightarrow Y; B \\
]]
\end{array}
$$

Handshaking:

$$
\begin{array}{l}
*[[ai \longrightarrow xo\uparrow; [xi]; ao\uparrow; [\neg ai]; xo\downarrow; [\neg xi]; ao\downarrow \\
\;\;|\,bi \longrightarrow yo\uparrow; [yi]; bo\uparrow; [\neg bi]; yo\downarrow; [\neg yi]; bo\downarrow \\
]]
\end{array}
$$

Yale AVLSI



## Arbitration

Introduce new variables $u$ and $v$:

$$
\begin{array}{l}
*[[ai \longrightarrow u\uparrow; [u]; xo\uparrow; [xi]; ao\uparrow; \\
\qquad [\neg ai]; u\downarrow; [\neg u]; xo\downarrow; [\neg xi]; ao\downarrow \\
\\
\;\;|\,bi \longrightarrow v\uparrow; [v]; yo\uparrow; [yi]; bo\uparrow; \\
\qquad [\neg bi]; v\downarrow; [\neg v]; yo\downarrow; [\neg yi]; bo\downarrow \\
]]
\end{array}
$$

The idea is to introduce the output of the arbiter into the handshaking expansion. The next step is to decompose the arbiter out of the handshaking expansion.

Yale AVLSI

## Process factorization

Idea: "factor out" an arbiter!

After process factorization:

$$
\begin{array}{l}
*[[ai \longrightarrow u\uparrow; [\neg ai]; u\downarrow \\
\quad | \, bi \longrightarrow v\uparrow; [\neg bi]; v\downarrow \\
]] \\
\\
\| \\
\\
*[[u \longrightarrow xo\uparrow; [xi]; ao\uparrow; [\neg u]; xo\downarrow; [\neg xi]; ao\downarrow \\
\quad [\!] \, v \longrightarrow yo\uparrow; [yi]; bo\uparrow; [\neg v]; yo\downarrow; [\neg yi]; bo\downarrow \\
]]
\end{array}
$$



## Process factorization

Production rules:

$$
\begin{array}{rcl}
\neg bo \wedge u & \mapsto & xo\uparrow \\
xi & \mapsto & ao\uparrow \\
(bo \vee) \neg u & \mapsto & xo\downarrow \\
\neg xi & \mapsto & ao\downarrow \\
\\
\neg ao \wedge v & \mapsto & yo\uparrow \\
yi & \mapsto & bo\uparrow \\
(ao \vee) \neg v & \mapsto & yo\downarrow \\
\neg yi & \mapsto & bo\downarrow
\end{array}
$$



## Arbitration with multiplexing

CHP Program:

$$
\begin{array}{l}
*[[\overline{A} \longrightarrow S; A \\
\quad | \, \overline{B} \longrightarrow S; B \\
]]
\end{array}
$$

Decomposition:

$$
\begin{array}{l}
*[[\overline{A} \longrightarrow P; A \\
\quad | \, \overline{B} \longrightarrow Q; B \\
]] \\
\| \\
*[[\overline{P} \longrightarrow S; P \\
\quad [\!] \, \overline{Q} \longrightarrow S; Q \\
]]
\end{array}
$$



## Arbitration with multiplexing

Handshaking:

$$
\begin{array}{l}
*[[pi \longrightarrow so\uparrow; [si]; po\uparrow; [\neg pi]; so\downarrow; [\neg si]; po\downarrow \\
\quad [\!] \, qi \longrightarrow so\uparrow; [si]; qo\uparrow; [\neg qi]; so\downarrow; [\neg si]; qo\downarrow \\
]]
\end{array}
$$

Production rules:

$$
\begin{array}{rcl}
pi \vee qi & \mapsto & so\uparrow \\
\neg pi \wedge \neg qi & \mapsto & so\downarrow
\end{array}
\qquad
\begin{array}{rcl}
si \wedge qi & \mapsto & qo\uparrow \\
(\neg qi \wedge) \neg si & \mapsto & qo\downarrow
\end{array}
$$

$$
\begin{array}{rcl}
si \wedge pi & \mapsto & po\uparrow \\
(\neg pi \wedge) \neg si & \mapsto & po\downarrow
\end{array}
$$

## Negated probes

Consider the following CHP program:

$$
\begin{array}{l}
*[\,[\overline{X} \wedge \overline{S} \longrightarrow S!\mathbf{true}, X \\
\quad |\, \neg\overline{X} \wedge \overline{S} \longrightarrow S!\mathbf{false} \\
\quad ]\,]
\end{array}
$$

This program determines the current value of the probe.
$S$ determines when the probe is evaluated.

- Why a thin bar?!

Yale AVLSI



## Negated probes

Assuming the channels are passive, we get the following handshaking expansion:

$$
\begin{array}{l}
*[\,[Xi \wedge Si \longrightarrow Sto\uparrow;\ [\neg Si];\ Sto\downarrow;\ Xo\uparrow;\ [\neg Xi];\ Xo\downarrow \\
\quad |\, \neg Xi \wedge Si \longrightarrow Sfo\uparrow;\ [\neg Si];\ Sfo\downarrow \\
\quad ]\,]
\end{array}
$$

Since the CMOS implementation of a two-way arbiter is weakly fair, we can implement this HSE with the following:

$$
\begin{array}{l}
*[\,[Xi \longrightarrow [Si];\ Sto\uparrow;\ [\neg Si];\ Sto\downarrow;\ Xo\uparrow;\ [\neg Xi];\ Xo\downarrow \\
\quad |\, Si \longrightarrow Sfo\uparrow;\ [\neg Si];\ Sfo\downarrow \\
\quad ]\,]
\end{array}
$$

Yale AVLSI



## Negated probes

Introduce arbiter variables:

$$
\begin{array}{l}
*[\,[Xi \longrightarrow u\uparrow;\ [u];\ [Si];\ Sto\uparrow;\ [\neg Si];\ Sto\downarrow; \\
\qquad\qquad Xo\uparrow;\ [\neg Xi];\ u\downarrow;\ [\neg u];\ Xo\downarrow \\
\quad |\, Si \longrightarrow v\uparrow;\ [v];\ Sfo\uparrow;\ [\neg Si];\ v\downarrow;\ [\neg v];\ Sfo\downarrow \\
]\,]
\end{array}
$$

Apply process factorization:

$$
\begin{array}{l}
*[\,[u \longrightarrow [Si];\ Sto\uparrow;\ [\neg Si];\ Sto\downarrow;\ Xo\uparrow;\ [\neg u];\ Xo\downarrow \\
\quad [\!]\, v \longrightarrow Sfo\uparrow;\ [\neg v];\ Sfo\downarrow \\
\quad ]\,] \\
\| \\
\quad Arb(Xi, Si, u, v)
\end{array}
$$

Yale AVLSI



## Negated probes

Reshuffled HSE:

$$
\begin{array}{l}
*[\,[u \longrightarrow [Ei];\ Eto\uparrow;\ [\neg Ei];\ Xo\uparrow;\ Eto\downarrow;\ [\neg u];\ Xo\downarrow \\
\quad [\!]\, v \longrightarrow Efo\uparrow;\ [\neg v];\ Efo\downarrow \\
\quad ]\,]
\end{array}
$$

Production rules:

$$
\begin{array}{rcl@{\qquad\qquad}rcl}
u \wedge \_Xo \wedge Ei & \mapsto & \_Eto\downarrow & Xo & \mapsto & \_Xo\downarrow \\
\neg\_Xo & \mapsto & \_Eto\uparrow & \neg Xo & \mapsto & \_Xo\uparrow \\
\\
\_Xo \wedge v & \mapsto & Efo\downarrow & u & \mapsto & \_u\downarrow \\
\neg v & \mapsto & Efo\uparrow & \neg u & \mapsto & \_u\uparrow \\
\\
\neg\_u \wedge \neg\_Eto \wedge \neg Ei & \mapsto & Xo\uparrow \\
\_u \wedge \_Eto & \mapsto & Xo\downarrow
\end{array}
$$

Yale AVLSI