Syntax-directed translation

Rajit Manohar

Asynchronous VLSI and Architecture (AVLSI) Group
Computer Systems Lab, Yale University

https://csl.yale.edu/~rajit/
https://avlsi.csl.yale.edu/act
A direct path from CHP to gates

• Goal: to provide a direct path from CHP to gates

• “Syntax directed”
  ❖ Translation uses the syntax of the CHP program to generate the circuit
  ❖ Uses structural induction
    ✷ Induction on the *structure* of the program
    ✷ Translations for
      ‣ Base case: assignment, communication, skip, expression evaluation
      ‣ Induction: selections, loops, sequential composition, parallel composition

• History
  ❖ 1988 : paper at Conference on Advanced Research in VLSI (Caltech)
  ❖ 1991 : Tangram language / Haste @ Handshake Solutions (Philips Research)
  ❖ 1998 : Balsa, based on Tangram with extensions (U. Manchester)
Key idea

• Use a communication channel to select a program for execution

• Given a program “P”, we will implement the following

```c
*[ // infinite loop
    [#C]; // wait for pending C
    P; // execute P
    C? // finish C
]*
```

• We execute “P” by simply executing

```
C!
```

• This is sometimes called “process decomposition” or “process call”
Wire implementation of channels

- Channel “C” that controls the execution of a program
One execution: idle (waiting) state

channel

C.r

C.a

wait

request

acknowledge

time

Yale
One execution: request execution

C.r

C.a

channel

P

wait

request

acknowledge

time

Yale
One execution: running
One execution: done

C.r

C.a

channel

c.r

c.a

P

done

request

acknowledge

time
One execution: respond to requester

C.r

C.a

request

acknowledge

time

channel

C.r

C.a

ack

P

Yale
One execution: reset phase

- C.r: Request
- C.a: Acknowledge
- P: Process

Diagram shows the interaction over time with request and acknowledge signals.
One execution: reset phase

C.r

C.a

request

acknowledge

time

channel

C.r

P

wait

C.a
Variables

• Two operations
  ❖ Write a value to the variable
    \[ W!\text{value} \]
  ❖ Read the current value of the variable
    \[ R?x \]

• The variable itself is “passive”
  ❖ It waits for the environment to either write or read its value
Writing and reading a variable

- **Write** (W):
  - Write request: W.r
  - Write data: W.d[N]
  - Write acknowledge: W.a

- **Read** (R):  
  - Read request: R.r
  - Read data: R.d[N]
  - Read acknowledge: R.a

**Diagram:**
- Variable (var) with input and output connections for writing and reading.
- Data flow illustration.

**Keywords:**
- Write
- Read
- Request
- Acknowledge
- Data

**Institution:**
- Yale
- AVLSI
Writing and reading a variable

write
W.r
W.d[N]
W.a

read
R.r
R.d[N]
R.a

var

Yale

AVLSI
Sending and receiving on a channel

C.r C.a
channel
X.r
X.d[N]
X.a

send
recv

request
acknowledge
data

Send / receive

Yale

AVLSI
Expression evaluation

- Example of expression de-composition

Yale
Assignment
Building blocks

```
    C.r  
  <-   ->  skip
    C.a  

    C.r  
  <-   ->   <<empty box>>
    C.a  
```
Building blocks

- **skip**
  - `c.r` ← `c.a`

- **S₁;S₂**
  - `c.r` ← `c.a`

- **S₁**
  - `c.r` ← `c.a`

- **S₂**
  - `c.r` ← `c.a`
Building blocks

- Skip
- $S_1; S_2$
- $S_1, S_2$

Yale
Selections and loops

\[ [G_1 \rightarrow S_1] \quad [\emptyset \quad G_2 \rightarrow S_2] \]

Selection
Selections and loops

Selection

\[
\begin{align*}
& [ \ G_1 \rightarrow S_1 \\
&[ ] \ G_2 \rightarrow S_2 \\
\end{align*}
\]

Loop

\[
\begin{align*}
& [ \ G_1 \rightarrow S_1 \\
&[ ] \ G_2 \rightarrow S_2 \\
\end{align*}
\]