Translating SMALL Programs to FPGA Configurations

Ying Shen & Theodore S. Norvell
Memorial University of Newfoundland

NECEC 1999

SMALL

SMALL is ...

- An imperative programming language
- *Synchronous* and *parallel*
- A low level language — compared to C
- A high level language — compared to VHDL
- Intended for hardware design

The SMALL implementation

- From program to hardware at a keypress.
- Short design cycles
- Reasonable efficiency
- Thesis:
  - hardware is becoming very cheap
  - hardware is programmed more than manufactured
  - programmed hardware is getting faster faster

  Design costs will dominate.

The elements of SMALL

Types and Expressions

- Booleans — Logic expressions
- N-Dimensional Arrays — Arithmetic & APLish expressions

Entities

- Registers — Record values
- Signals — Transfer values to parallel statements.

Statements

Assign Register:	$r \leftarrow$ *Exp*
Assert Signal:	s ! *Exp*
Wait for clock:	*tick*
Parallel composition:	$\mathbf{par}\ S \parallel T \parallel U\ \mathbf{rap}$
Sequential composition:	$S\ T\ U$
Choice:	$\mathbf{if}$ *Exp* $\mathbf{then}\ S\ \mathbf{else}\ T\ \mathbf{fi}$
Loops:	$\mathbf{while}\ Exp\ \mathbf{do}\ S\ \mathbf{od}$
Loops:	$\mathbf{repeat}\ S\ \mathbf{until}$ *Exp*

Timing semantics

Time passes only

- at *tick* statements
- after loop iterations
- for parallel process waiting for another to terminate.

$$s \mathbin{!} 10 \text{ as } 4 \text{ bits}$$
$$t \mathbin{!} s - u$$

is identical to

$$t \mathbin{!} s - u$$
$$s \mathbin{!} 10 \text{ as } 4 \text{ bits}$$

But

$$s \mathbin{!} 10 \text{ as } 4 \text{ bits}$$
$$tick$$
$$t \mathbin{!} s - u$$

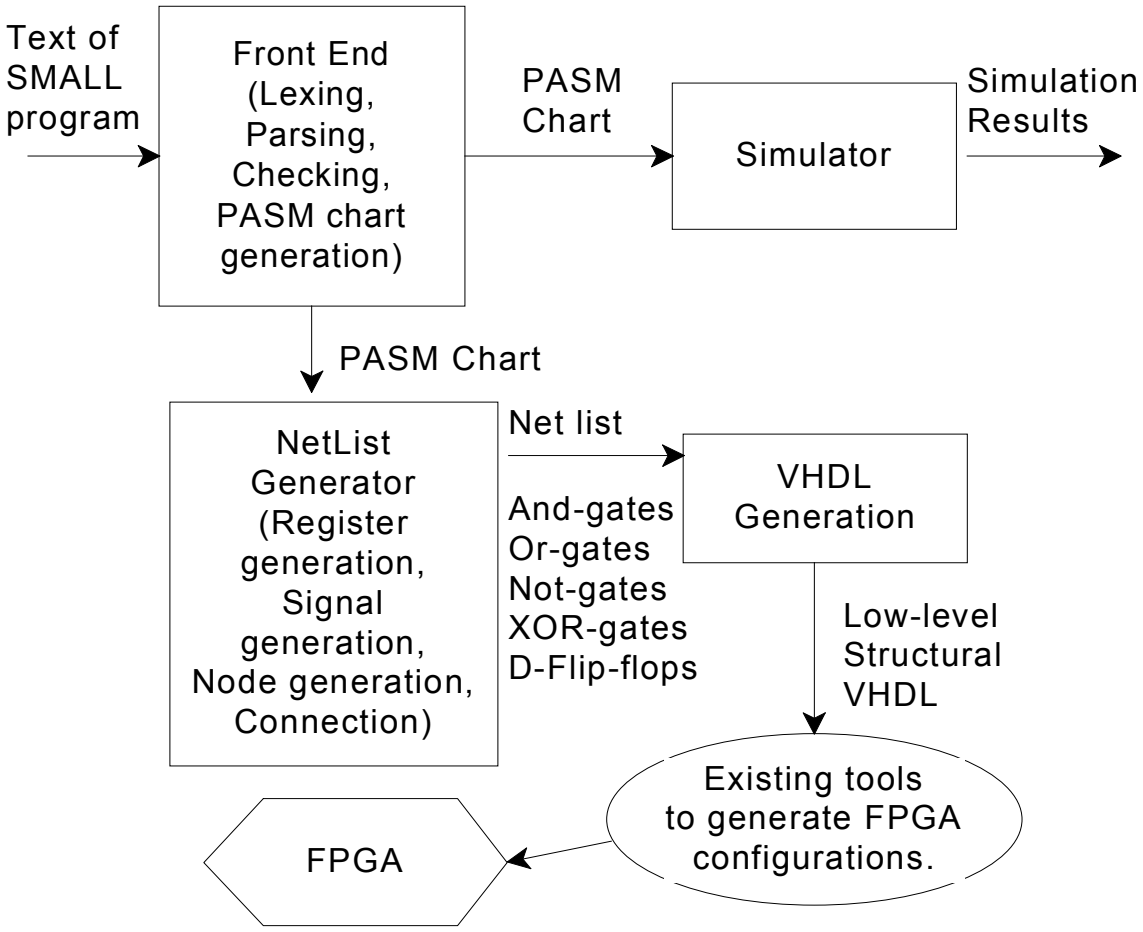
is NOT identical to

$$t \mathbin{!} s - u$$
$$tick$$
$$s \mathbin{!} 10 \text{ as } 4 \text{ bits}$$

Implementations

- Simulator — for debugging of designs
- Compiler — for hardware implementation

Implementation block diagram



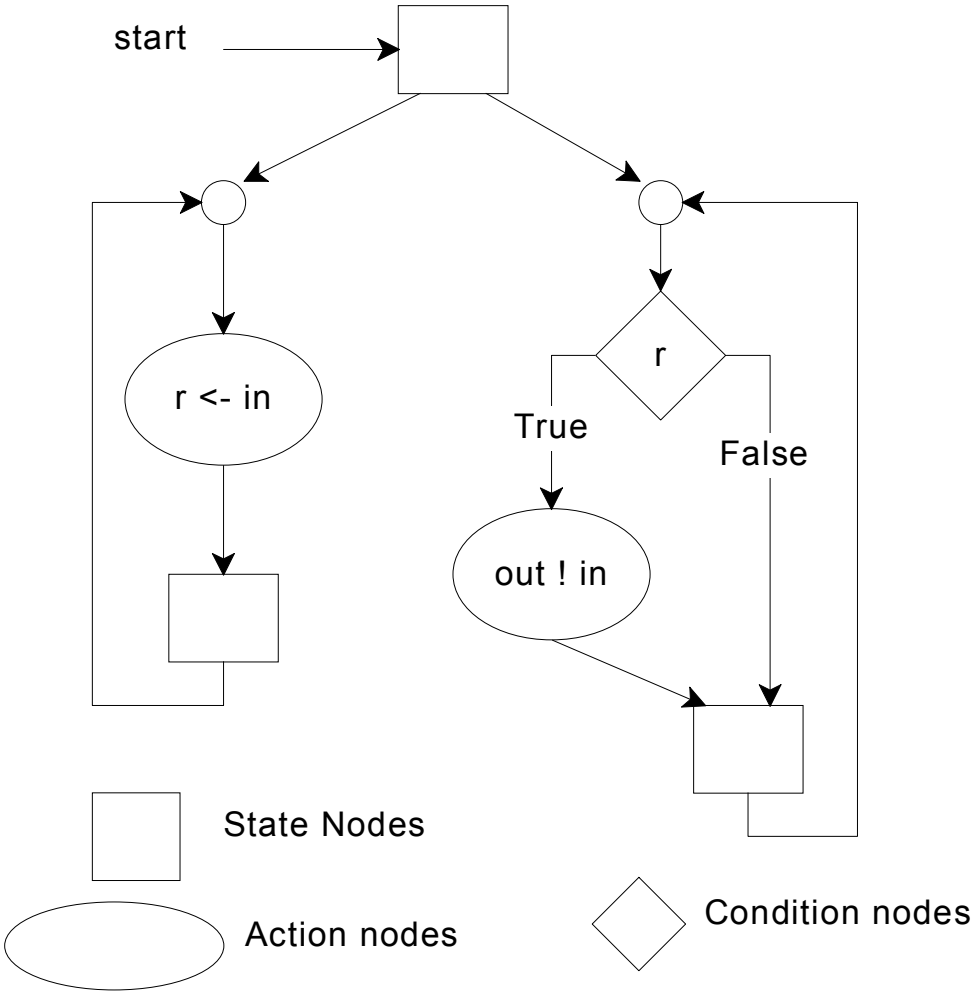
Parallel Algorithmic State Machine Charts (PASM Charts)

A state-machine representation for control flow.

Example SMALL program

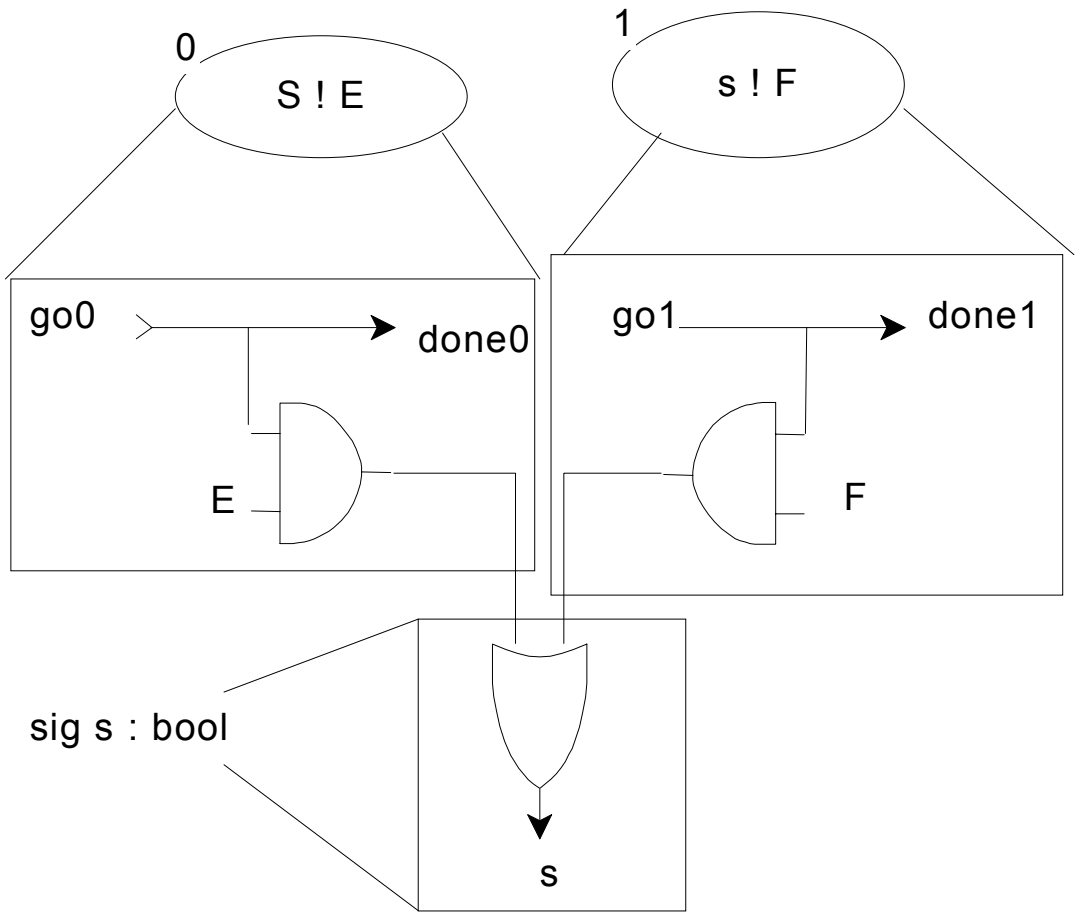
```
global sig in : bool
global sig out : bool
reg r : bool
 par
        while true
        do r ← in
        od
  ||
        while true
        do if r
           then out ! in
           fi
        od
 rap
```

Example PASM Chart

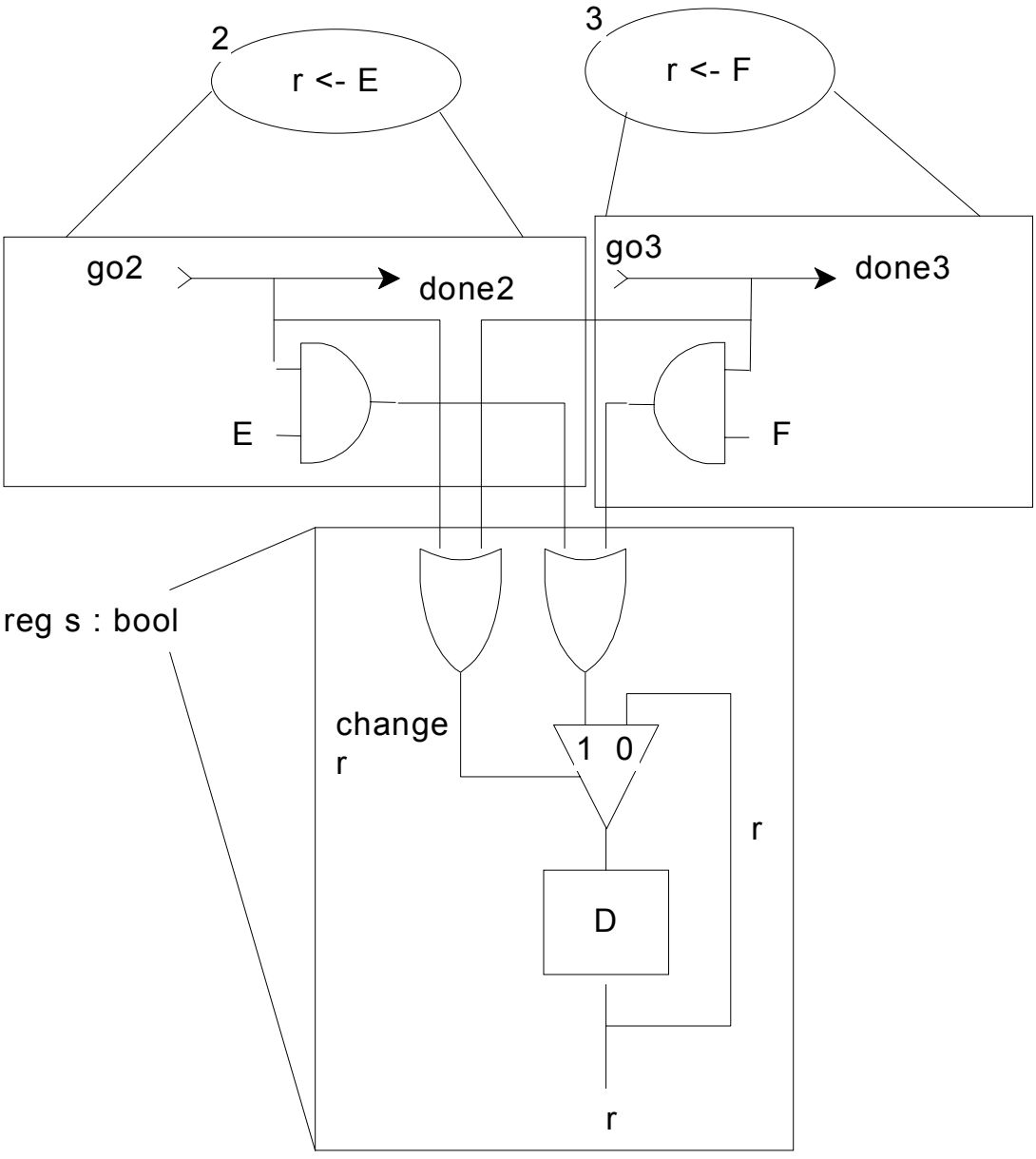


The state is represented by a set of active state-nodes. Nodes reachable from an active state-node are executed. Then all state nodes reachable from an active state-node become active.

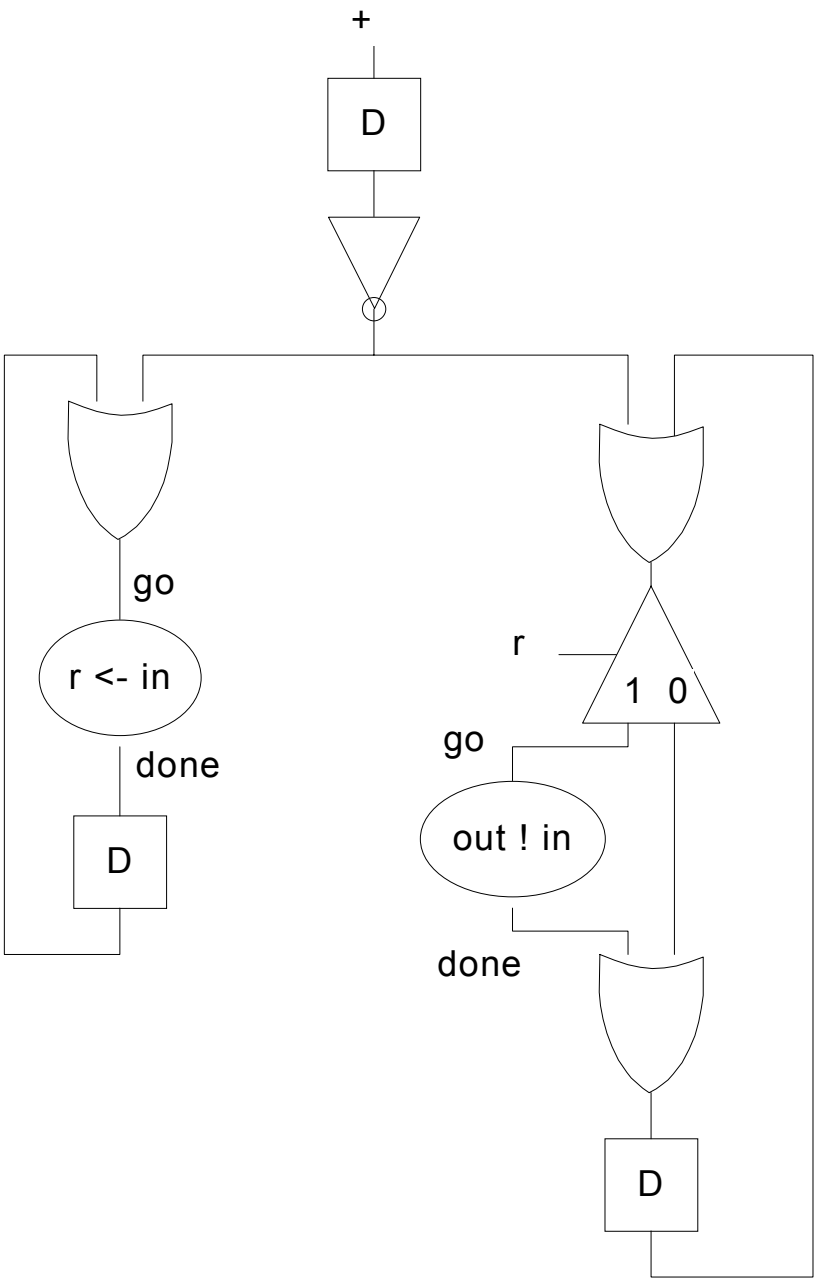
Translation of signals and asserts



Registers and assignments



Dealing with Control State



Optimization

- Dead device and wire elimination
  - Deadwood is removed
- Constant folding and propagation
  - Ground and power inputs eliminated
- Common subcircuit elimination
  - Two devices with the same inputs
- PASM chart level — not yet done
  - Sharing of nodes and expressions

Future Directions

- Language improvements
  - Asynchronous communication.
  - Module system and separate compilation
  - Higher level types
  - User defined types
- Compiler improvements
  - Speed and Space
  - More optimization — resource sharing.
- Industry involvement