Cornell University
Computer Systems Laboratory

CHP Compiler

Ned Bingham & Nicholas Kramer

Problem Statement

CSL

Asynchronous is awesome

Tool infrastructure lacking

CHP great at describing behavior

PRs great at representing circuitry

Bridge that gap?

How Would You Use This?

CSL

Synthesize new elements from CHP...

...Using syntax we are all familiar with...

...that can be integrated into any toolchain requiring PRs

How might we do this?

CSL

Syntax Directed Translation (Tangram)

Pre-Charge Half Buffer Design (ARCWelder)

Martin Synthesis (Us!)

Problems: CHP to HSE

CSL

Expand expressions

Expand process/operator instantiations

Expand multi bit variables/records/channels

Optimizations

Problems: HSE to State Space

CSL

Information loss in guards

Information loss in parallel branches

Loop invariant

Impossible branches

Conflicting states

Problems: State Space to PRs

CSL

Minterms

Logic Minimization

Optimizations (symmetrization, factoring...)

Supported Syntax: CHP

CSL

Assignment

```
q, r, t := x^y | z^w, a+b, a*b;
```

Conditional

```
[ a | b -> c+ [] ~a & ~b -> c- ]
```

Loop

```
*[ a -> x := x + 1; a := 0
 [] ~a -> x := x*x; a := 1
 ]
```

sequential composition

```
x := x^y; y := x^y; x := x^y
```

assignment

parallel composition

```
q := x^y || r := y^z || s := z^x
```

assignment

Supported Syntax: Instantiation

```
int<32> var;
int var;
mux0b2 MyMux(a, b, c, d);
chan4p1b MyChan;
```

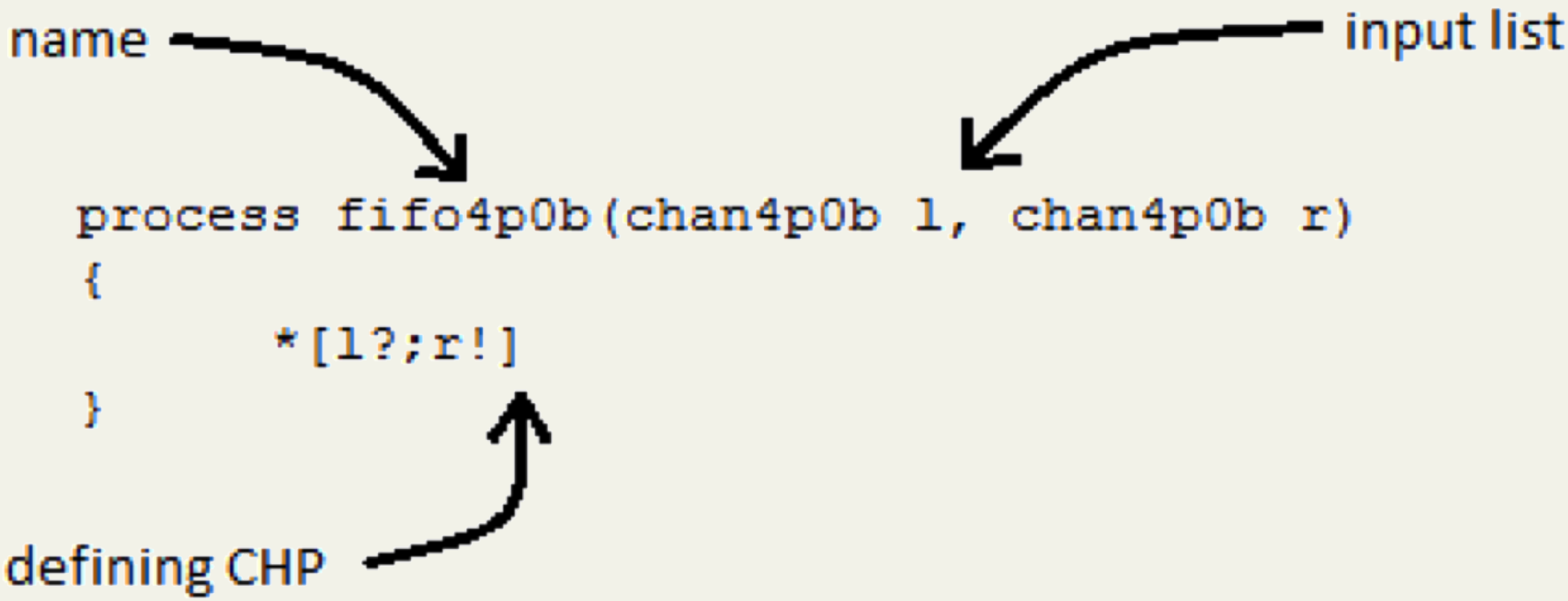
Supported Syntax: Records

```
record dualrail
{
        int<1> t;
        int<1> f;
}
```

name

variable list

Supported Syntax: Processes



Supported Syntax: Operators

name

input list

```
operator&(int<1> s, int<1> a, int<1> b)
{
        [    a &  b -> s+
        [] ~a | ~b -> s-
        ];
}
```

defining CHP

Supported Syntax: Channels

```
channel chan4p0b
{
        int<1> r;
        int<1> a;

        operator!()
        {
                r+; [a]; r-; [~a];
        }

        operator?()
        {
                [r]; a+; [~r]; a-;
        }

        operator@()
        {
                r
        }
}
```

name

variable list

send operator

receive operator

probe operator

Implementation: Preprocessing

CSL

#include means libraries!

//comments are useful

/*So are block comments
that span multiple lines!*/

nee d to remove unne
eded whit e spa
ce t o make pa rsi ng eas ier

Implementation: Shortcuts/Parsing

CSL

Easier to parse if syntax 1to1 with meaning

So, make 'shortcut' syntax fully explicit!

e.g. a.r+ means a.r:=1 , [a.a] means [a.a->skip], *[S] means *[1->S]

Recursively break CHP into lists of blocks/instructions

Implementation: Instantiation

CSL

Record Instantiation

Process Instantiation

Copy code + search and replace

Expression Instantiation

Parse Expression + Process Instantiation

Implementation: Guard/Assign Merging

CSL

[G1][G2] turns into [G1&G2]

[G1->[Ga->Sa[]Gb->Sb]]
turns into
[G1&Ga->Sa[]G1&Gb->Sb]

x+; y-; z:=x; turns into x,y:=1,0; z:=x;

Implementation: State Space Generation

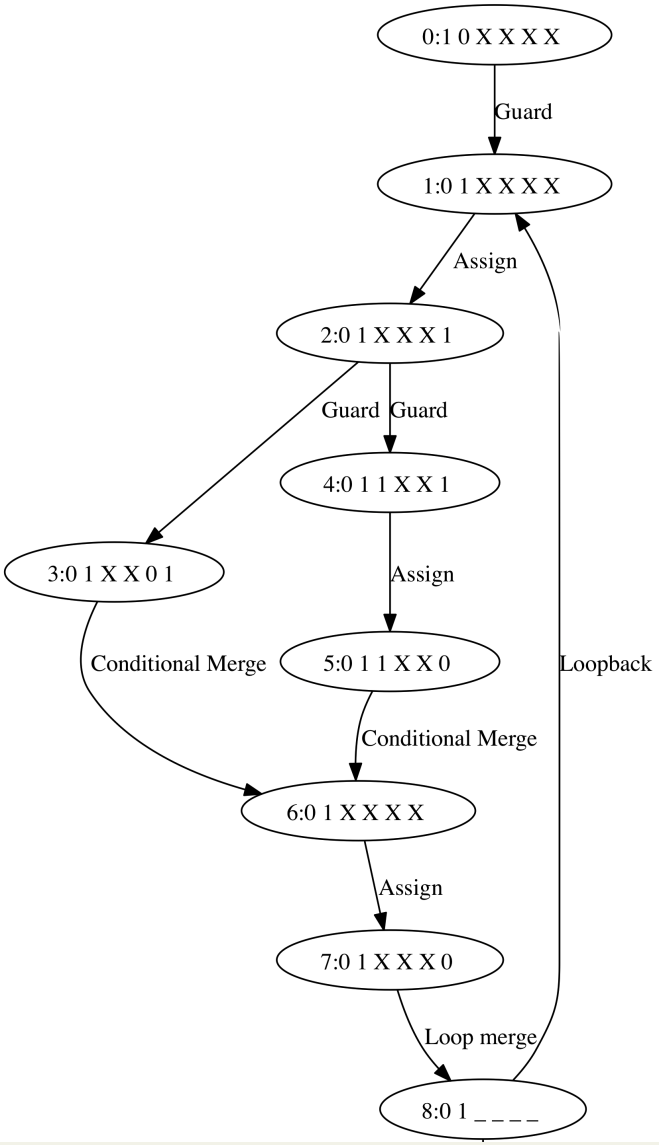
CSL

What gains state information?

- Assignments
- Guards
- Mutual Exclusion (future)

What loses information?

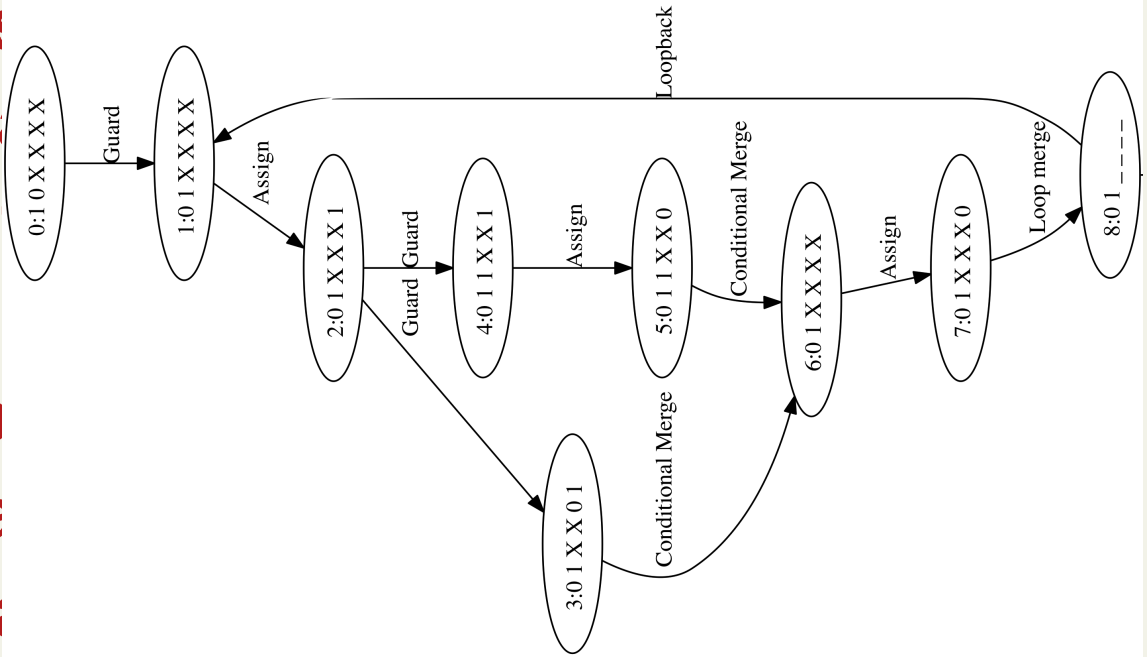
- Branch Merges
- Channel Assigns
- Across Parallel Branches



Implementation: Trace Space Generation

What gains state information?

- Ass
- Gu
- Mu



What lo

- Bra
- Cha
- Across Parallel Branches

Implementation: Scribe Variables

CSL

Find guard with OR

Create variable whose name is the guard

Add it to the state space

Implementation: State Variable Insertion

CSL

Maximize the number of conflicts removed

Brute Force

Implementation: PR Generation

CSL

For each variable/direction pair:

Find all implicant states that cause change

Build weakest 'sufficient' PR guard

Implementation: Logic Minimization

CSL

Quine McCluskey reduction:

- Reduce to prime implicants (think prime factors)
- Find 'essential' implicants

Factor implicants:

- Too long to implement?
- Lots of common terms?

Future Directions:

CSL

Handshake Reshuffling

Process Decomposition/Projection

Impossible Branch Removal

Isochronic Fork Detection

Future Directions:

CSL

Power Analysis

Timing Analysis

Design Space Exploration

CHP to CHP Optimizations

Verification

Toolchain Integration

CSL

Full integration into ACT

CHP Block

Interchangeable ACT and CHP libraries

Requests/Suggestions/Questions??

CSL

With all the time we have already and plan to spend on this project, let's try to make it as useful as possible!

Tech demo until Out of time!!