Automatic Data Path Extraction in Large-Scale Register-Transfer Level Designs

Wei Song, Jim Garside and Doug Edwards

School of Computer Science, the University of Manchester

Manchester M13 9PL United Kingdom

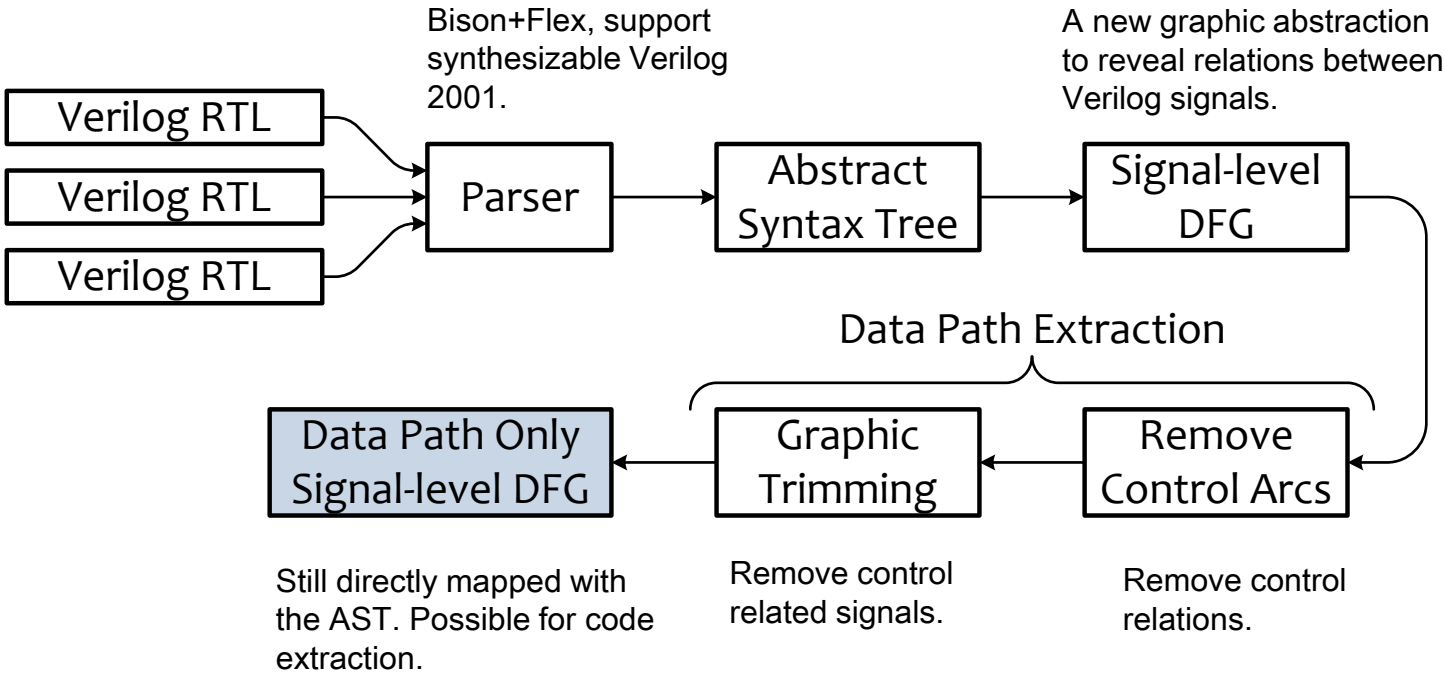
02/06/2014

Advanced Processor Technologies Group
The School of Computer Science

Motivation

- Data path extraction is important
  - Hardware verification
  - Synthesis for asynchronous circuits
  - System partition
- Methodology
  - State-space analysis
    - actuate but slow and not scalable
  - Pattern matching
    - Fast, scalable but inaccurate
    - Accuracy depends on accurate type recognition

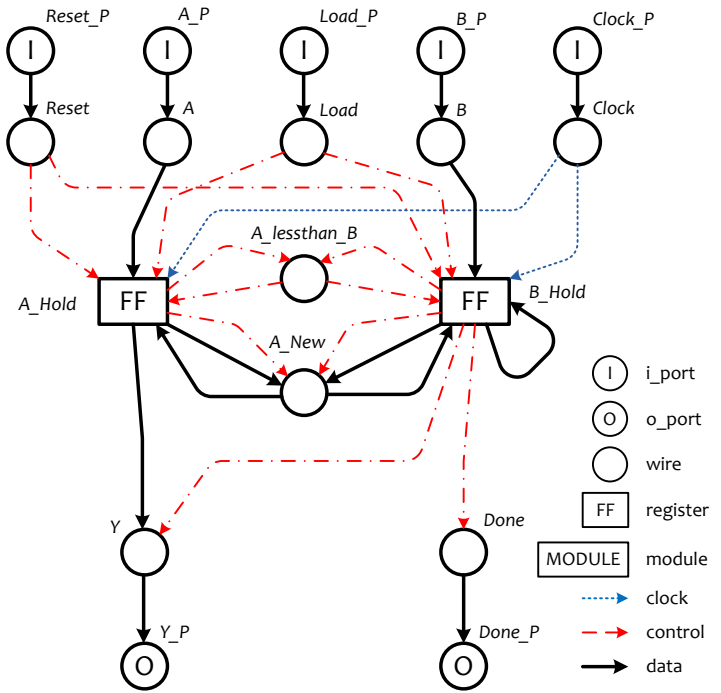
Tool Flow


Bison+Flex, support synthesizable Verilog 2001.
A new graphic abstraction to reveal relations between Verilog signals.
Verilog RTL
Verilog RTL
Verilog RTL
Parser
Abstract Syntax Tree
Signal-level DFG
Data Path Extraction
Data Path Only Signal-level DFG
Graphic Trimming
Remove Control Arcs
Still directly mapped with the AST. Possible for code extraction.
Remove control related signals.
Remove control relations.

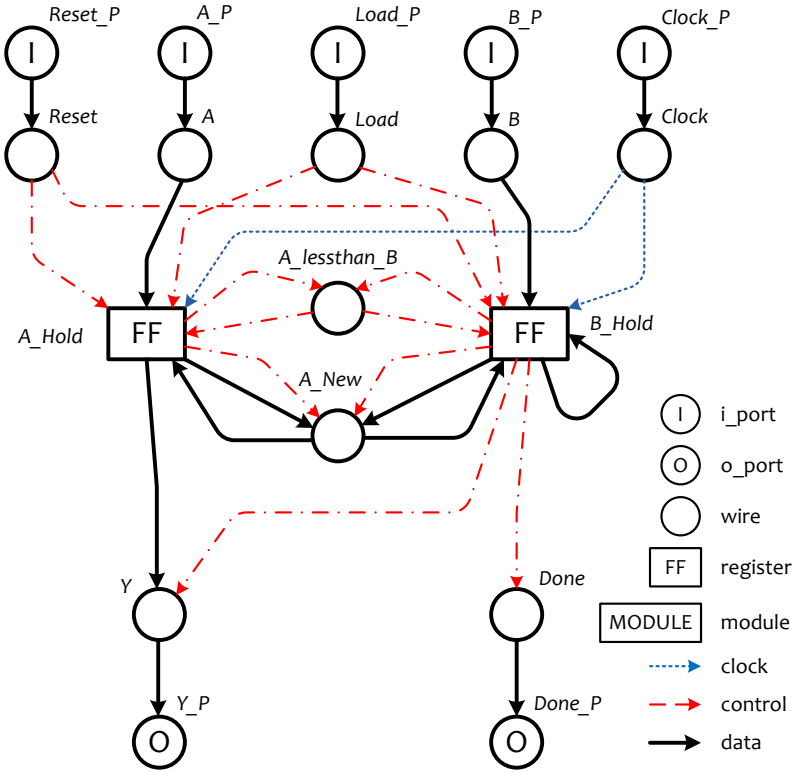
Signal-Level Data Flow Graph (DFG)

```
module GCD (Clock,Reset,Load,A,B,Done,Y);
  input  Clock,Reset,Load;
  input  [7:0] A,B;
  output Done;
  output [7:0] Y;
  reg A_lessthan_B, Done;
  reg [7:0] A_New, A_Hold, B_Hold, Y;

  always @(posedge Clock)
    if(Reset) begin
      A_Hold = 0; B_Hold = 0;
    end else if(Load) begin
      A_Hold = A; B_Hold = B;
    end else if(A_lessthan_B) begin
      A_Hold = B_Hold;
      B_Hold = A_New;
    end else
      A_Hold = A_New;

  always @(A_Hold or B_Hold)
    if(A_Hold >= B_Hold) begin
      A_lessthan_B = 0;
      A_New = A_Hold - B_Hold;
    end else begin
      A_lessthan_B = 1;
      A_New = A_Hold;
    end

  always @(A_Hold or B_Hold)
    if(B_Hold == 0) begin
      Done = 1; Y = A_Hold;
    end else begin
      Done = 0; Y = 0;
    end
endmodule
```

A Greatest Common Divider (GCD) Calculator



Definition of a Signal-Level DFG



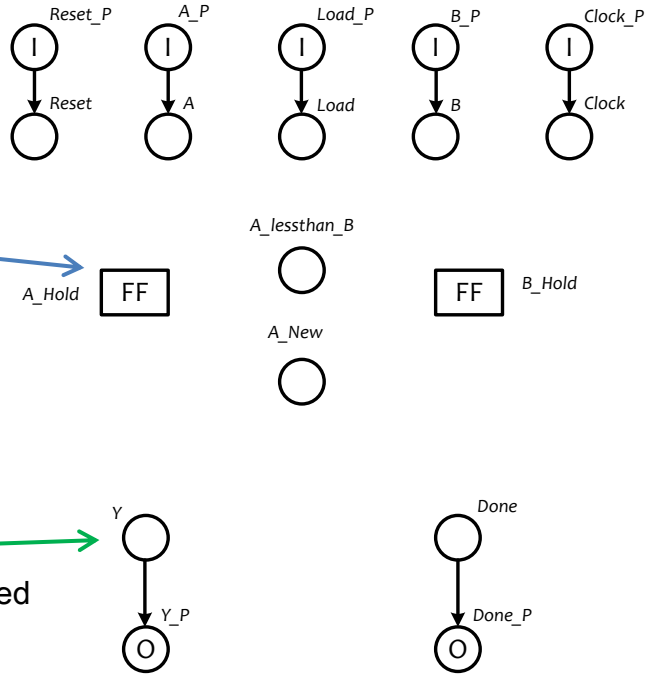
A signal-level DFG is a directed multi-graph denoted by a six-tuple:

$DFG = (V, A, T_V, F_V, T_A, F_A)$

V	a finite set of nodes representing Verilog components (signal and module)
$A \subseteq V \times V$	a finite set of arcs connecting nodes
T_V	a set of node types $\{reg, wire, in, out, module\}$
$F_V : V \rightarrow T_V$	map types to nodes
T_A	a set of arc types $\{control, data, clock, reset\}$
$F_A : A \rightarrow T_A$	map types to arcs

Node insertion

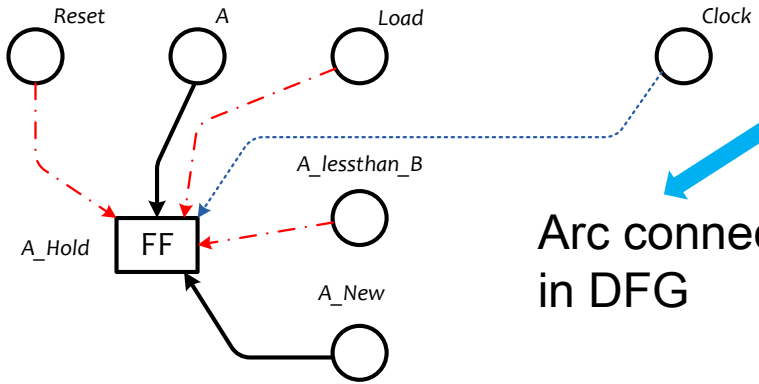
```
input  Clock,Reset,Load;
input  [7:0] A,B;
output Done;
output [7:0] Y;
reg A_lessthan_B, Done;
reg [7:0] A_New, A_Hold, B_Hold, Y;

always @(posedge Clock)
  if(Reset) begin
    A_Hold = 0; B_Hold = 0;
  end else if(Load) begin
    A_Hold = A; B_Hold = B;
  end else if(A_lessthan_B) begin
    A_Hold = B_Hold;
    B_Hold = A_New;
  end else
    A_Hold = A_New;

always @(A_Hold or B_Hold)
  if(B_Hold == 0) begin
    Done = 1; Y = A_Hold;
  end else begin
    Done = 0; Y = 0;
  end
```



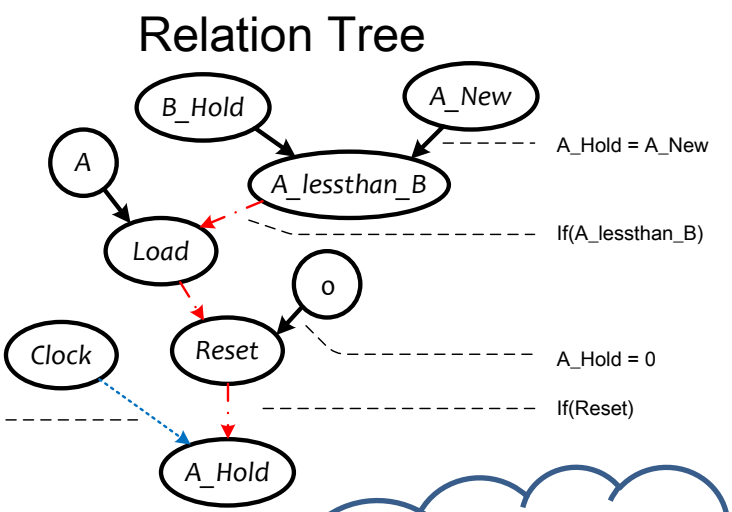
Latch will be detected by case analysis.

Type Estimation of Arcs

```
always @(posedge Clock)
    if(Reset) begin
      A_Hold = 0; B_Hold = 0;
    end else if(Load) begin
      A_Hold = A; B_Hold = B;
    end else if(A_lessthan_B) begin
      A_Hold = B_Hold;
      B_Hold = A_New;
    end else
      A_Hold = A_New;
```

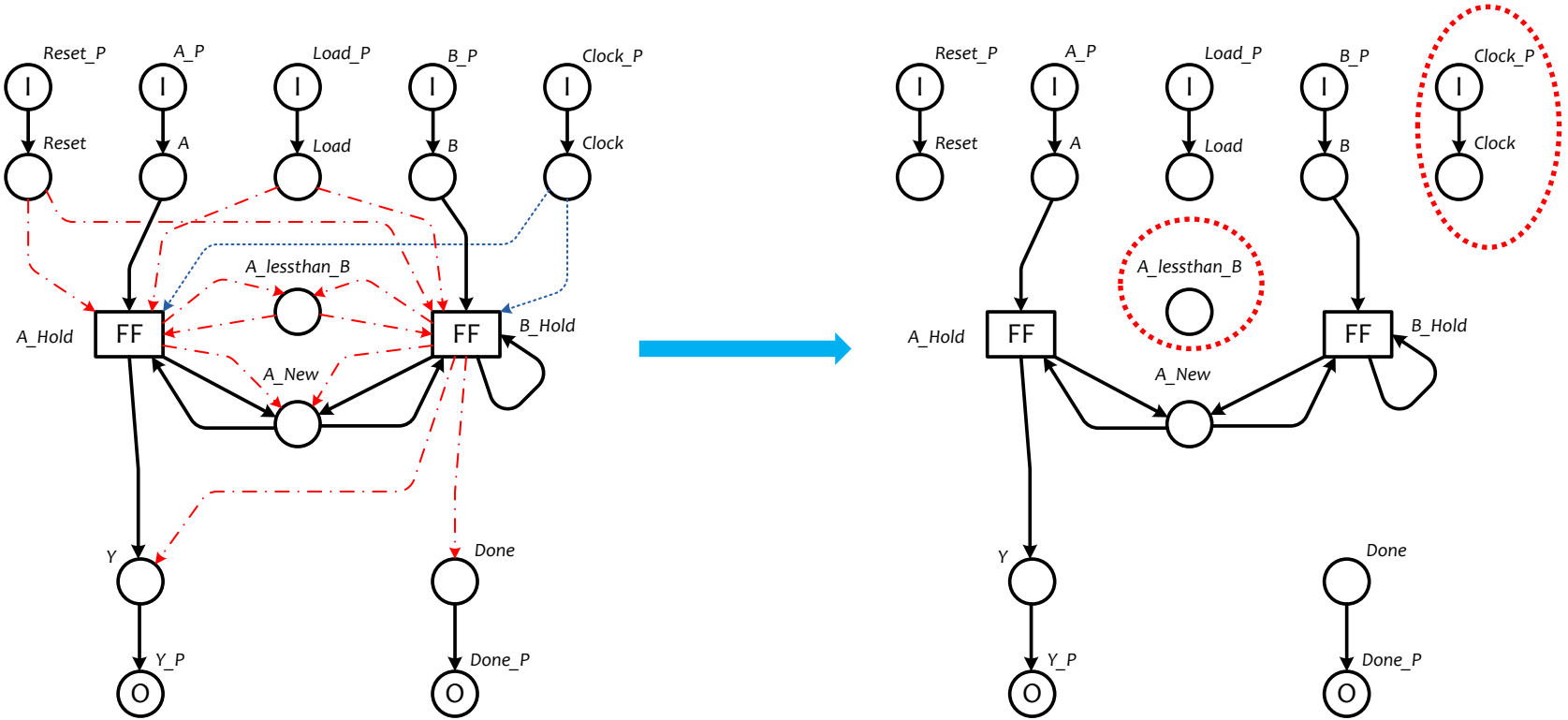


Arc connection in DFG

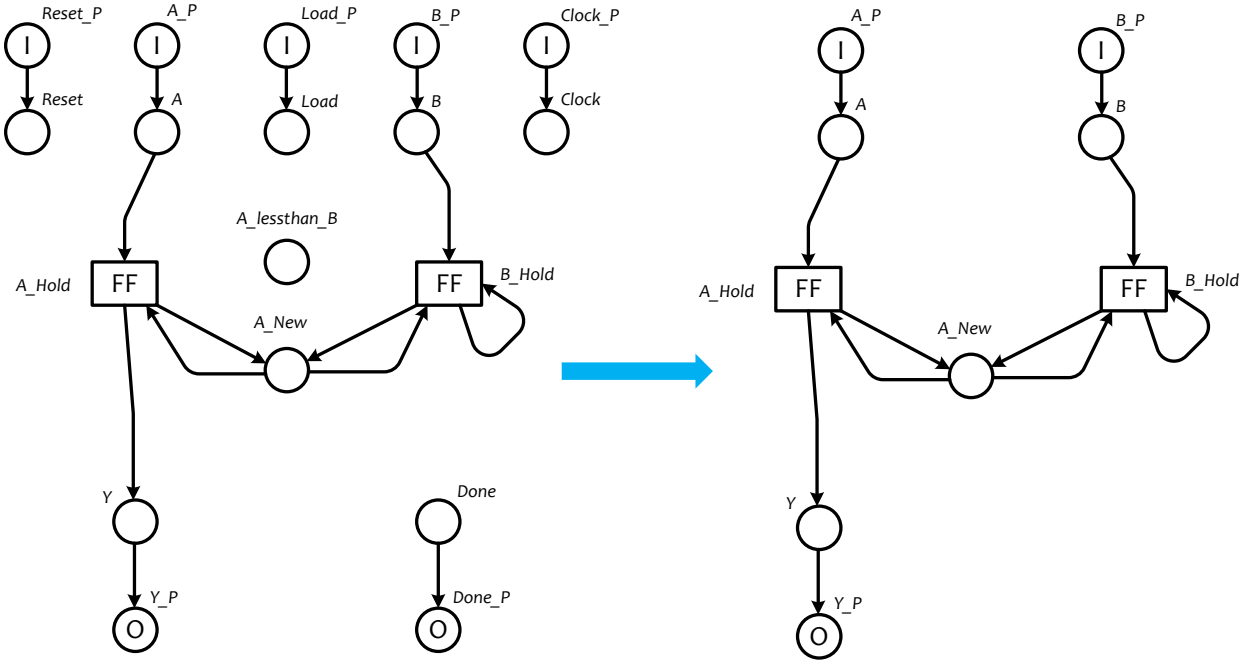


control recognition:
if(sig)
case(sig)
C = sig?A:B
D = mem[addr]

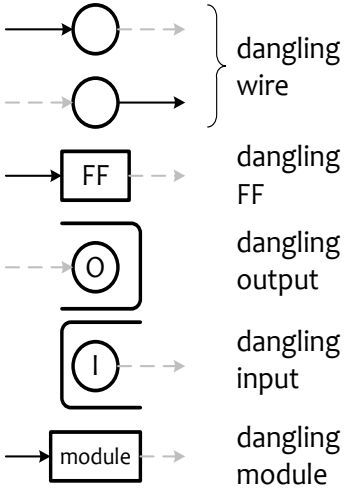
Remove Control Arcs



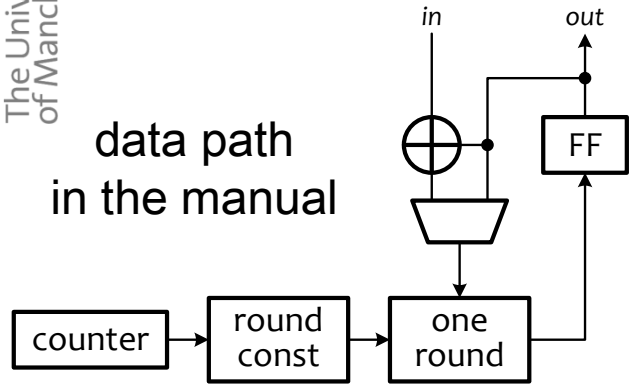
Recursively Remove Dangling Components



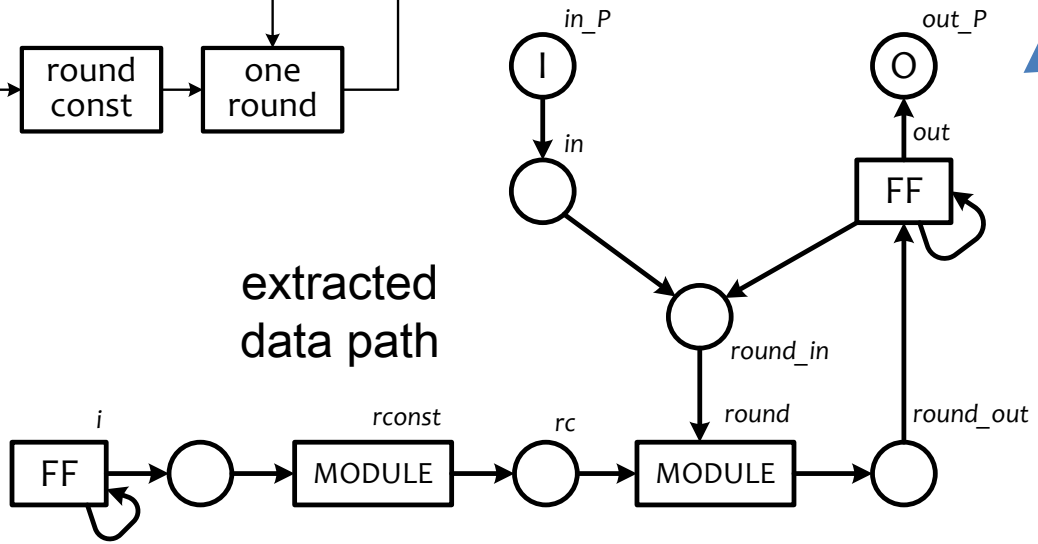
Nodes to be recursively removed



Hierarchical Designs (SHA-3 Encoder)



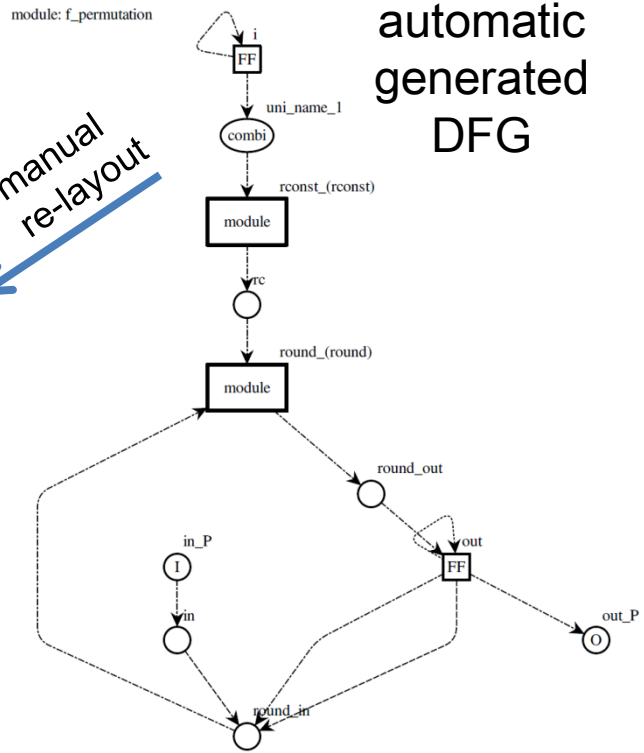
data path in the manual



extracted data path

manual re-layout

automatic generated DFG



Large-Scale Test Cases

Designs	Signal-level DFG			Data path DFG			Time (s)
	I/O	Module	Signal	I/O	Module	Signal	
OR1200	52	37	2074	40	33	1142	1
RSD	7	24	1063	3	23	659	<1
NOVA	19	140	7043	9	103	4279	10

OR1200: A 5-stage OpenRISC processor.
RSD: An industrial standard Reed-Solomon decoder.
NOVA: An FPGA proven H.264/AVC baseline decoder.

Intel Core™2 Due 3.00 GHz with 2GB memory

Possible Usages

- Signal-level DFG
  - Code extraction
  - Controller detection*
  - Interface recognition (Memory, bus, handshake)
- Data path extraction
  - Data-flow analysis by switching back-annotation
  - System partition

* Wei Song and Jim Garside. Automatic controller detection for large scale RTL designs. In Proc. of EUROMICRO Conference on Digital System Design (DSD), Santander, Spain, pp. 884–851, September 2013

Conclusion

- Signal-level DFG
  - A graphic representation of signal relations in RTL.
  - A typed and hierarchical multi-graph.
- Data path extraction
  - Trim all control related arcs and nodes.
  - Automatic control/data recognition.
  - Able to process large-scale designs.

Implementation and Test Environment

- Verilog parser
  - Bison, Flex and Vpreprocessor (Verilog Perl tool)[1]
  - C++ STL based abstract syntax tree
- Signal-level DFG
  - Boost Graphic Library (data structure)
  - Open Graph Drawing Framework (automatic layout)[2]
  - PugiXML (file format)
  - Qt (diagram drawer)
- User interface
  - Asynchronous Verilog Synthesiser[3]
  - Command line UI: embedded Tcl with self-defined commands

[1] http://www.veripool.org/verilog-perl
[2] http://www.ogdf.net
[3] https://github.com/wsong83/Asynchronous-Verilog-Synthesiser