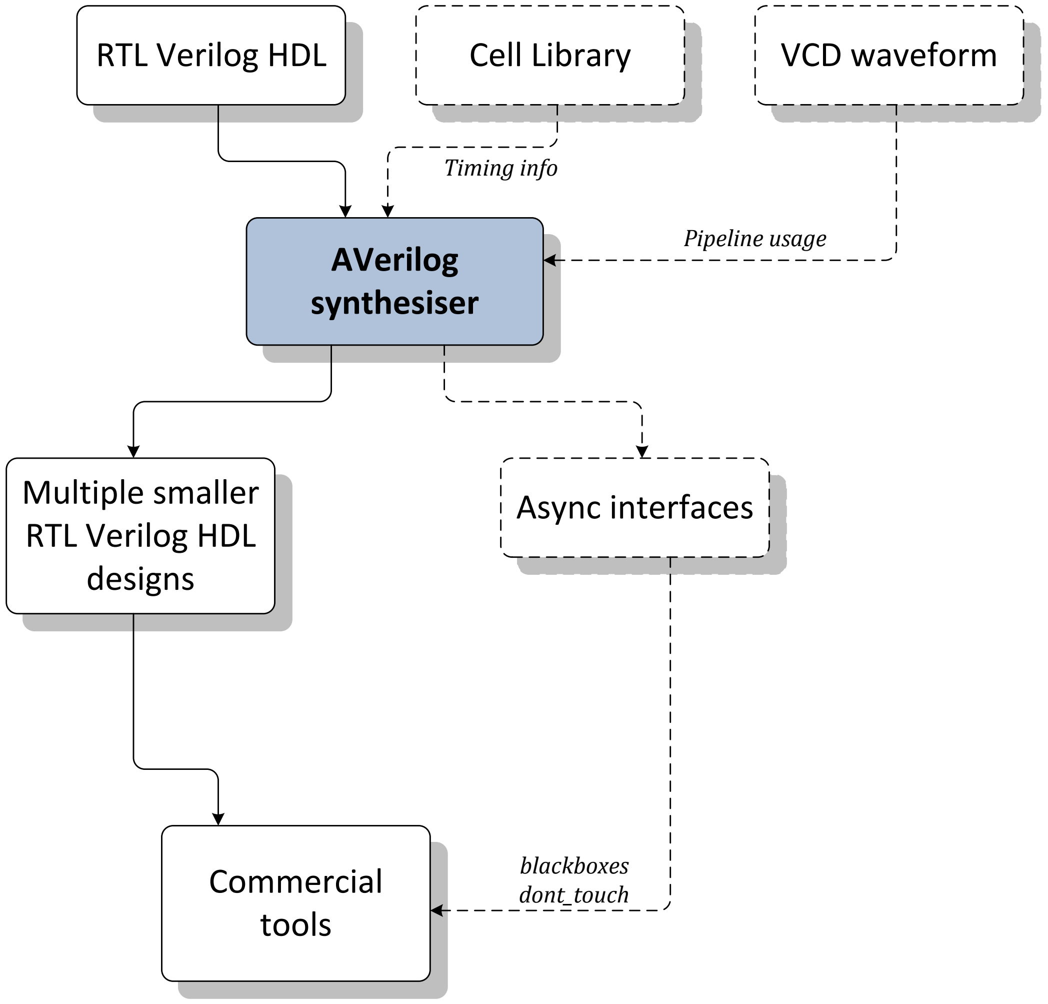
GAELS Progress

Wei Song

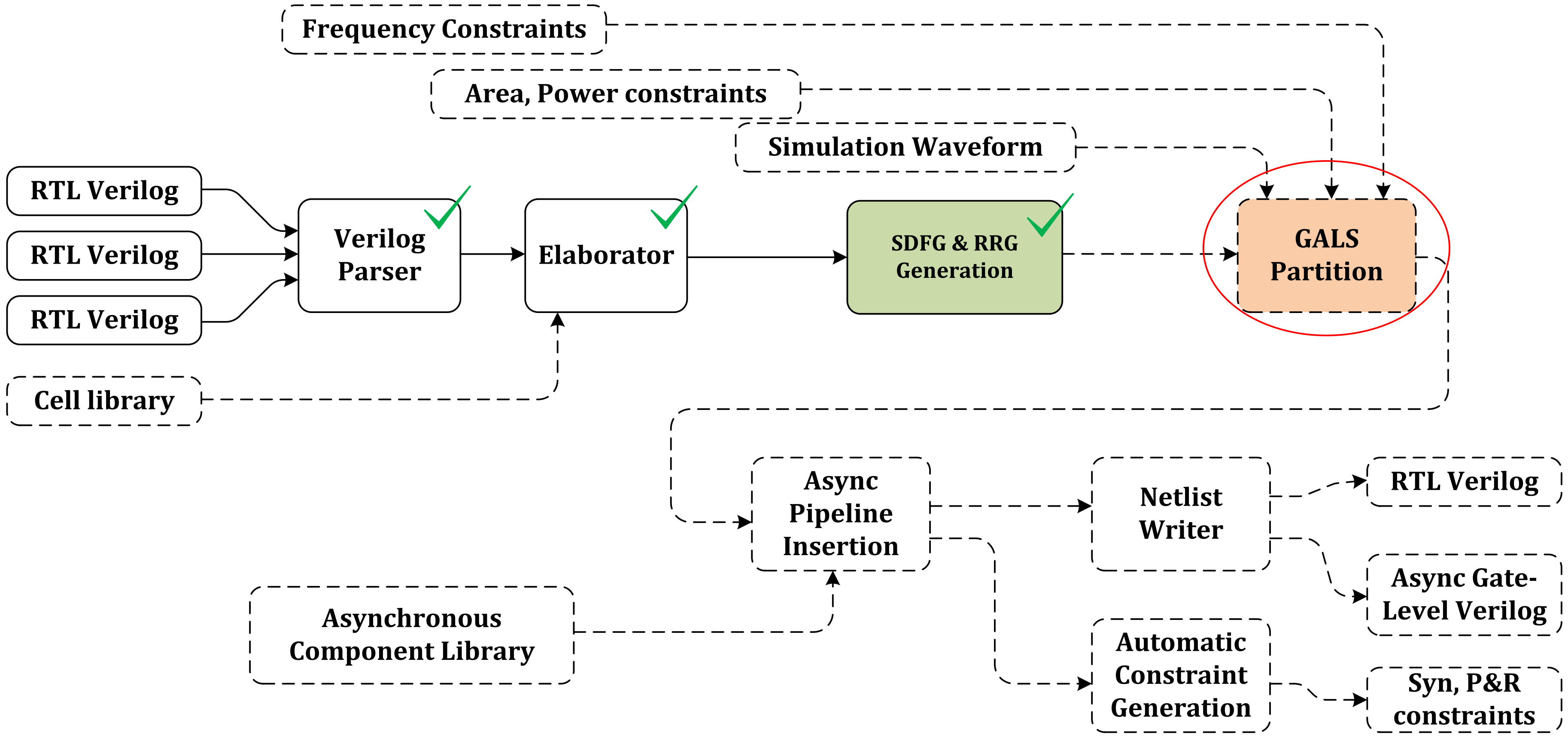
13/03/2013

Content

Tool flow

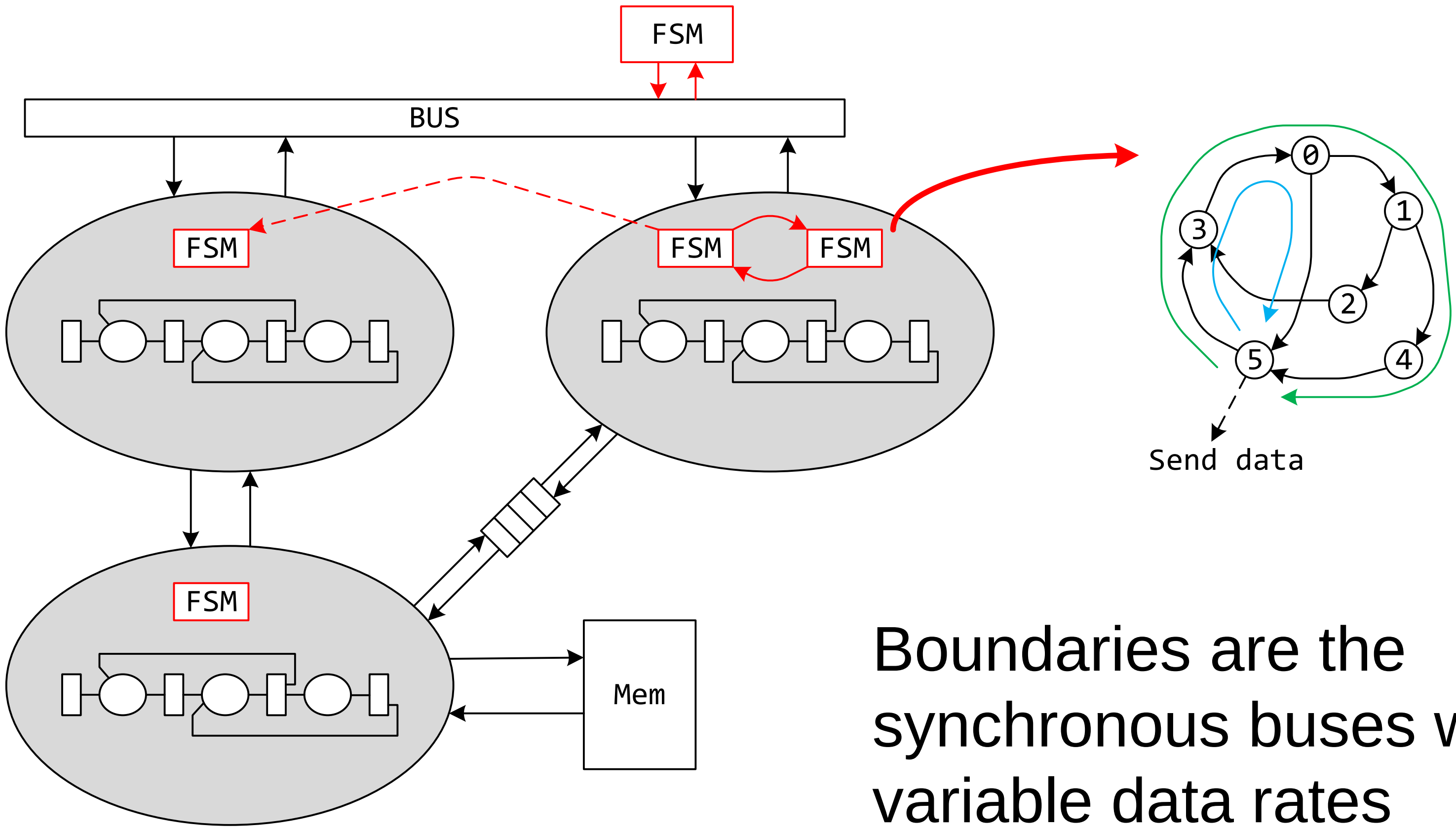

RTL Verilog HDL
Cell Library
VCD waveform
Timing info
Pipeline usage
AVerilog synthesiser
Multiple smaller RTL Verilog HDL designs
Async interfaces
blackboxes
dont_touch
Commercial tools

Flow inside Synthesizer



The University of Manchester

Hypotheses in Partition Detection



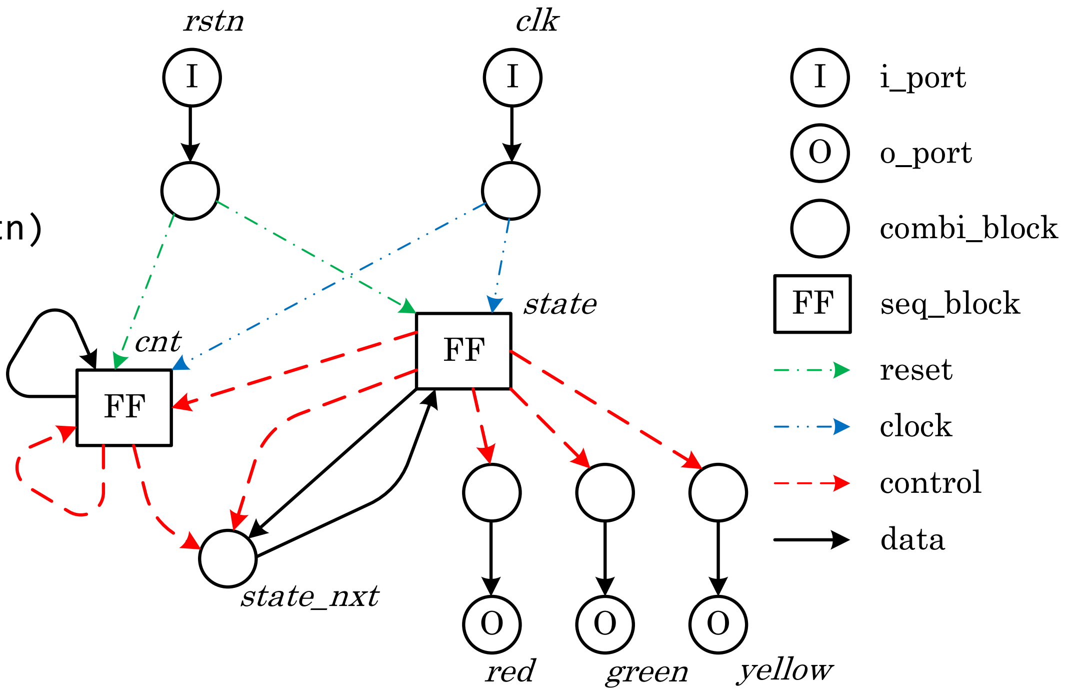
Boundaries are the synchronous buses with variable data rates controlled by FSMs.

Progress from Last Meeting

- Signal-level Data Flow Graph (DFG)
  - Parse Verilog to AST
  - AST to DFG conversion
  - Arc type detection
- Register Relation Graph
- Automatic FSM detection
  - Detect all FSMs, counters and flags with finite state spaces

Signal-level DFG

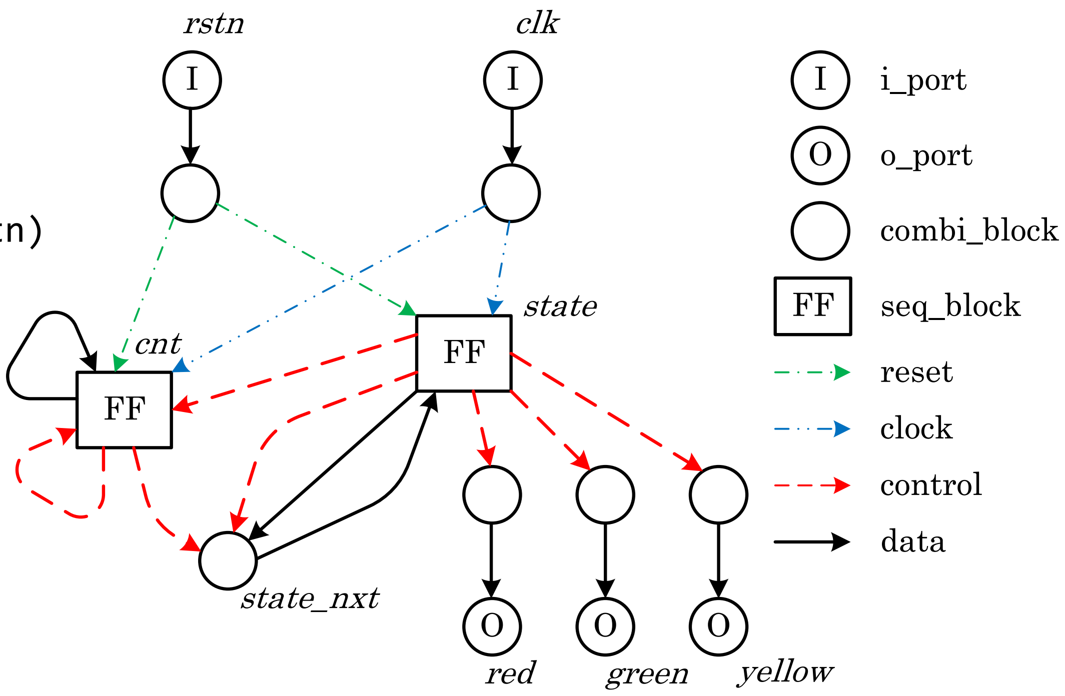
```
always @(posedge clk or negedge rstn)
  if(~rstn)
    state <= R;
  else
    state <= state_nxt;

always @(state or cnt) // next state
  if(cnt == 0)
    case(state)
    R:  state_nxt = YR;
    YR: state_nxt = G;
    G:  state_nxt = YG;
    default: state_nxt = R;
    endcase // case (state)
  else
    state_nxt = state;

always @(posedge clk or negedge rstn)
  if(~rstn)
    cnt <= 0;
  else if(cnt == 0)
    case(state)
    R:  cnt <= 2;
    YR: cnt <= 49;
    G:  cnt <= 4;
    default: cnt <= 49;
    endcase // case (state)
  else
    cnt <= cnt - 1;

assign red = state == R ? 1 : 0;
assign green = state == G ? 1 : 0;
assign yellow =
  (state == YR || state == YG) ? 1 : 0;
```



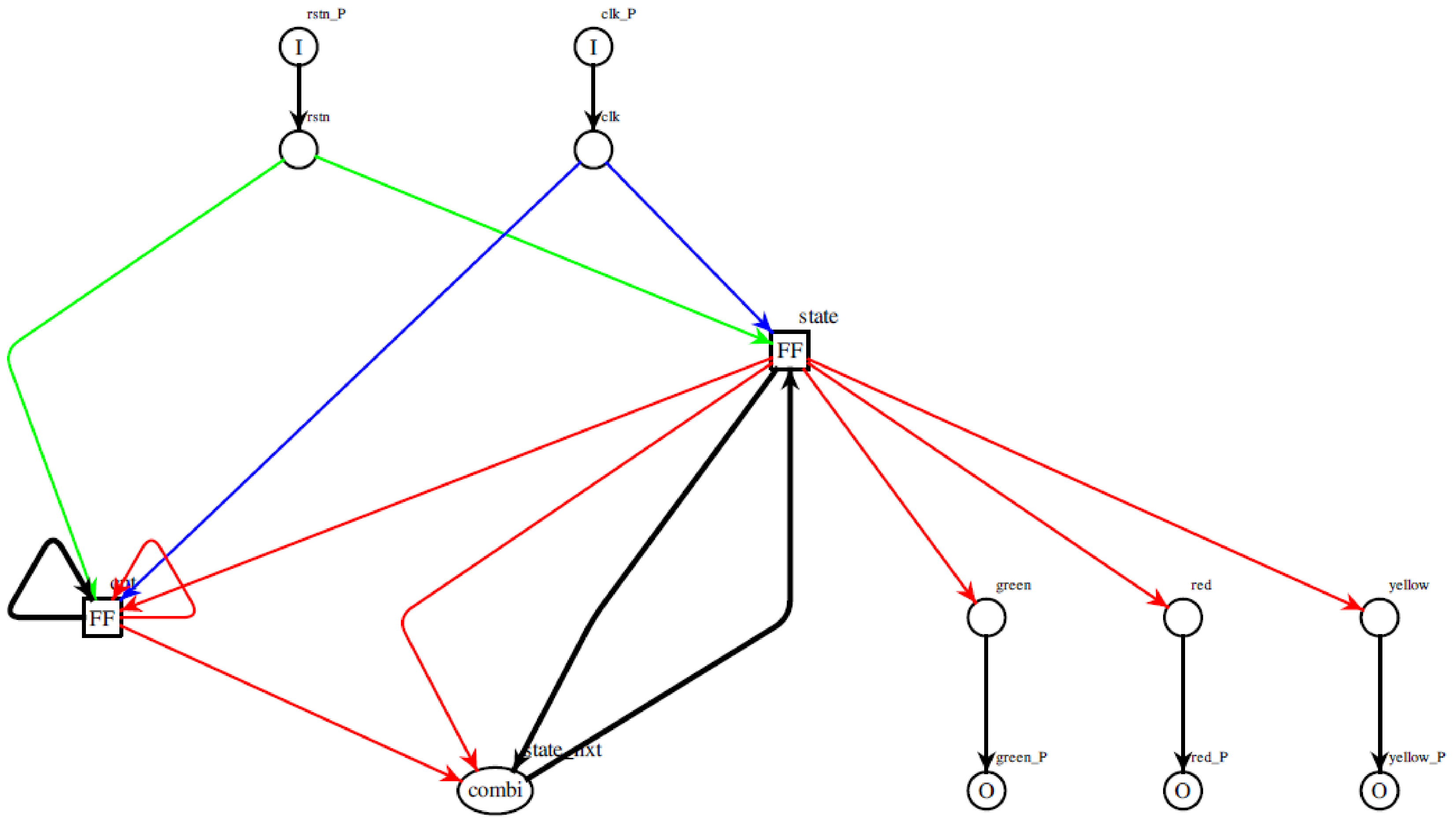
Signal-level DFG

```
always @(posedge clk or negedge rstn)
  if(~rstn)
    state <= R;
  else
    state <= state_nxt;

always @(state or cnt) // next state
  if(cnt == 0)
    case(state)
    R:   state_nxt = YR;
    YR:  state_nxt = G;
    G:   state_nxt = YG;
    default: state_nxt = R;
    endcase // case (state)
  else
    state_nxt = state;

always @(posedge clk or negedge rstn)
  if(~rstn)
    cnt <= 0;
  else if(cnt == 0)
    case(state)
    R:   cnt <= 2;
    YR:  cnt <= 49;
    G:   cnt <= 4;
    default: cnt <= 49;
    endcase // case (state)
  else
    cnt <= cnt - 1;

assign red = state == R ? 1 : 0;
assign green = state == G ? 1 : 0;
assign yellow =
  (state == YR || state == YG) ? 1 : 0;
```

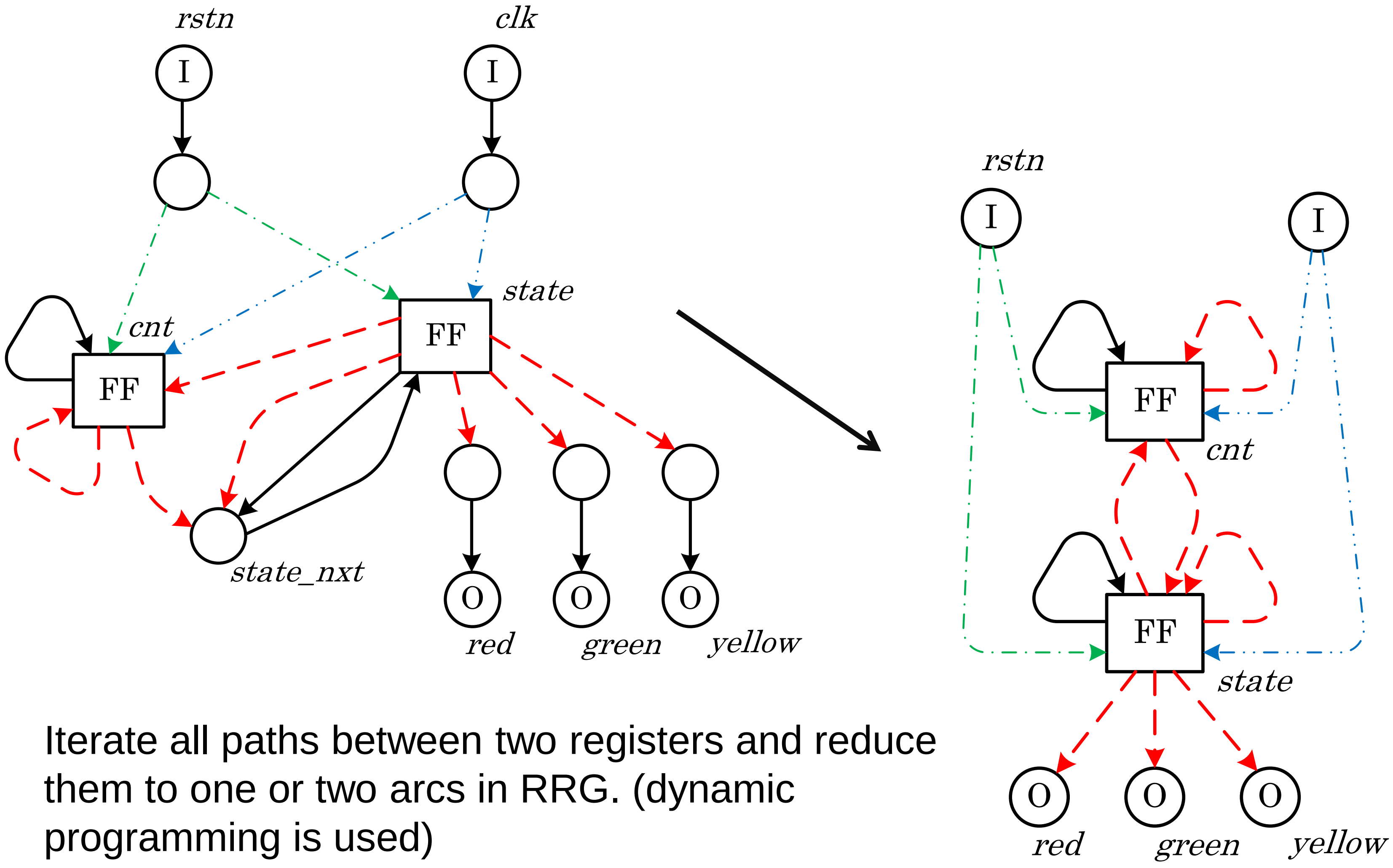
Arc types are extracted from the use of signals in statements.



Signal-level DFG



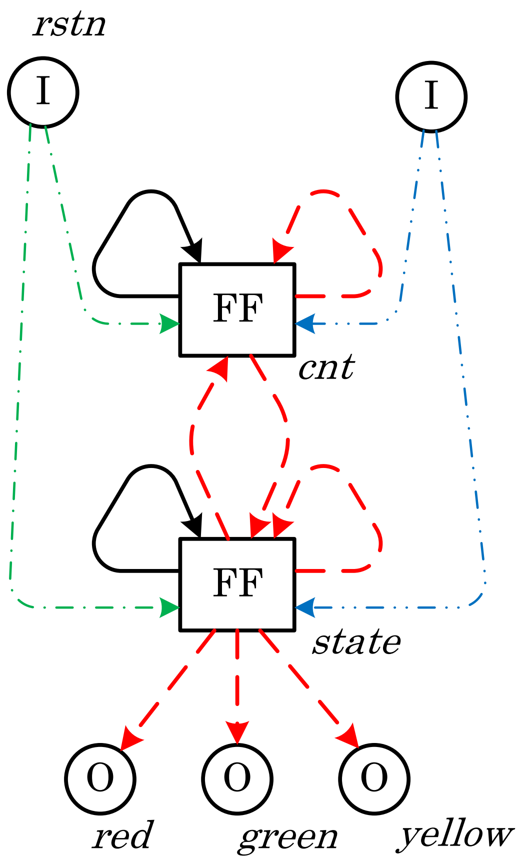
Register Relation Graph



Iterate all paths between two registers and reduce them to one or two arcs in RRG. (dynamic programming is used)

FSM detection

- At least one of an FSM's output paths is a self-loop path.
- At least one of an FSM's output paths is a control path towards another register.
- All input data of an FSM comes from self-loop paths or constant numbers.



Test Cases

Design	DFG Nodes	Registers	Time	FSMs		Rate	Types			
				Reported	Verified		FSM	Counter	Bit	Fake
OR1200	2074	124	$< 1s$	19	17	89%	7	5	5	2
Reed-Solomon	1063	325	$2.0s$	55	53	96%	6	35	12	2
H.264/AVC	7043	855	$7.1s$	55	49	89%	13	30	6	6

OR1200: micro processor, combinational loop, program counter.
RS decoder: ad hoc coding style, multiple signals in one always block, use range as control
H.264: large design with large fanouts (a global counter with 400 fanouts, 280K unfolded output paths).

False negative error: not found
False positive error: around 10%

Causes of error: combinational loop, range expression, sequential assigns.

Compare with Others

- Coding style
  - Synthesis tools like DC
  - Only recognise FSMs written in standard one or two always blocks
- Pattern recognition
  - [Liu2000] recognise FSMs described in an always block (block level granularity).
  - No support for explicit type detection.

Future works

- Boundary detection
  - Pattern of the buses with variable data rate
  - Pattern of on-chip buses
  - Pattern of FIFOs
  - Relations between controllers

Conclusions

- Large scale Verilog designs have been parsed and converted into signal level DFGs
- FSMs and controlling counters have been automatically detected
- Need a method to detect available system boundaries