

Predictive Synchronisation in the new Generation of GALS

Alex Bystrov, Delong Shang

School of Electrical Electronic and Computer Engineering
University of Newcastle upon Tyne

Rationale (1)

- ITRS-05 predicts:
 - 22% of global signalling will be asynchronous in 2013
 - GALS infrastructures
 - Reconfigurability for fault tolerance
 - 42% of functionality in 2013 will be implemented as reconfigurable logic (this does not include memory)
 - Design levels will be tightly coupled and jointly optimised

Rationale (2)

- GALS first appeared long ago, but their use is still very limited. Why???
- Usual difficulties with self-timed design
- Slow synchronisers
- Aperiodic clock (pausable, stretchable)
- Design must be latency insensitive
- Performance analysis and verification complications due to cyclic dependencies

Rationale (3)

- There is a hole in GALS classification
 - Synchronisation mechanics
 - fixed clock + synchronisers in Req/Ack circuits
 - pausable, stretchable clock, etc
 - Types of system clocking
 - synchronous
 - mesochronous, etc
 - Types of block interfaces
 - latency insensitive
 - weakly endochronous, etc.
 - Timeliness of synchronisation
 - aposteriority
 - **THE HOLE (space for Predictive Synch.)**

Our expectations

- Obvious features:
 - Synchronous inter-block communication
 - Improved latency and throughput
 - Improved power through the power-performance trade-off
- Not-so-obvious features:
 - Switching between FIXED clocks, no stretching
 - Dynamically formed synchronous clusters
 - Clock is treated as a resource
 - “Intelligent” clock allocation
 - Utilisation of communication info. of all system levels
 - Phase adjustment for the long interconnect, etc.

Difficulties

The main problem is to extract the prediction information from the given level of system specification. How to do it?

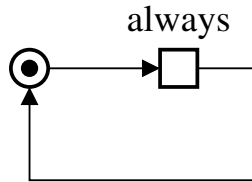
- Language → Formal model → Event prediction
 - Verilog
 - Petri nets
 - Prediction model
 - HW/SW predictor
- We have a simple example based on Verilog, but
- Are VHDL, Verilog, System-C, etc. are expressive enough?

Example of prediction – GCD

```
01 module gcd(x, y, z);
02     input [7:0] x, y;
03     output reg [7:0] z;
04     reg [7:0] x_reg, y_reg;
05     always
06     begin
07         x_reg = x; y_reg = y;
08         while (x_reg != y_reg)
09         begin
10             if (x_reg < y_reg)
11                 y_reg = y_reg - x_reg;
12             else
13                 x_reg = x_reg - y_reg;
14         end
15         z <= x_reg;
16     end
17 endmodule
```

Global PN for GCD

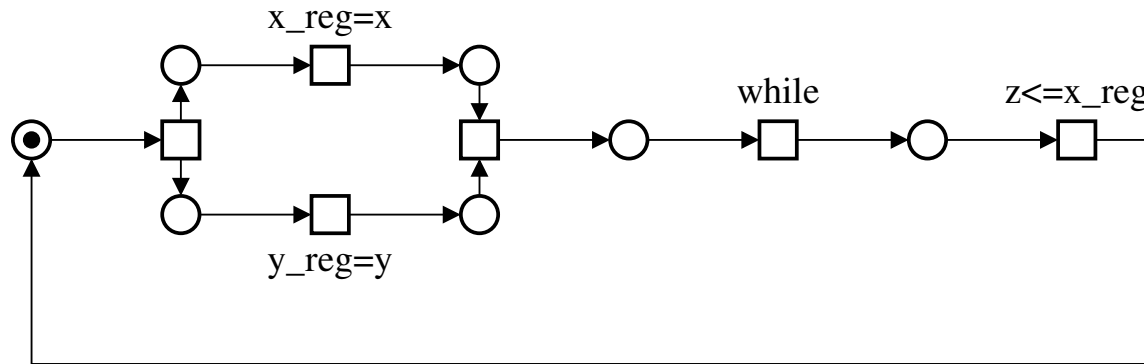
Initial PN for GCD module



```
x_reg = x; y_reg = y;
while (x_reg != y_reg)
begin
    if (x_reg < y_reg)
        y_reg = y_reg - x_reg;
    else
        x_reg = x_reg - y_reg;
end
z <= x_reg;
```

Global PN for GCD

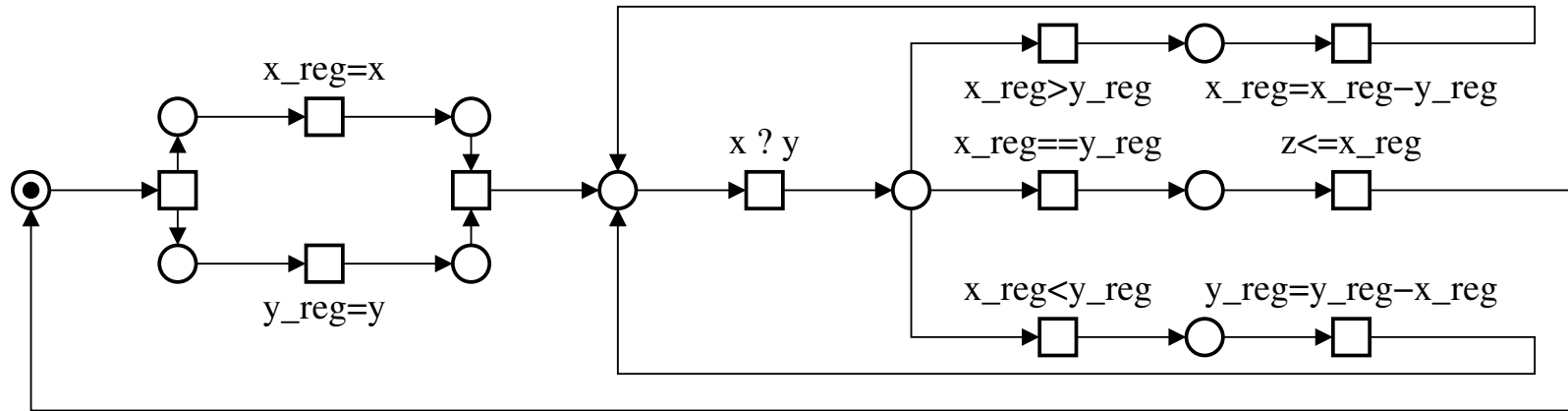
Always-statement refinement



```
x_reg = x; y_reg = y;
while (x_reg != y_reg)
begin
  if (x_reg < y_reg)
    y_reg = y_reg - x_reg;
  else
    x_reg = x_reg - y_reg;
end
z <= x_reg;
```


Global PN for GCD

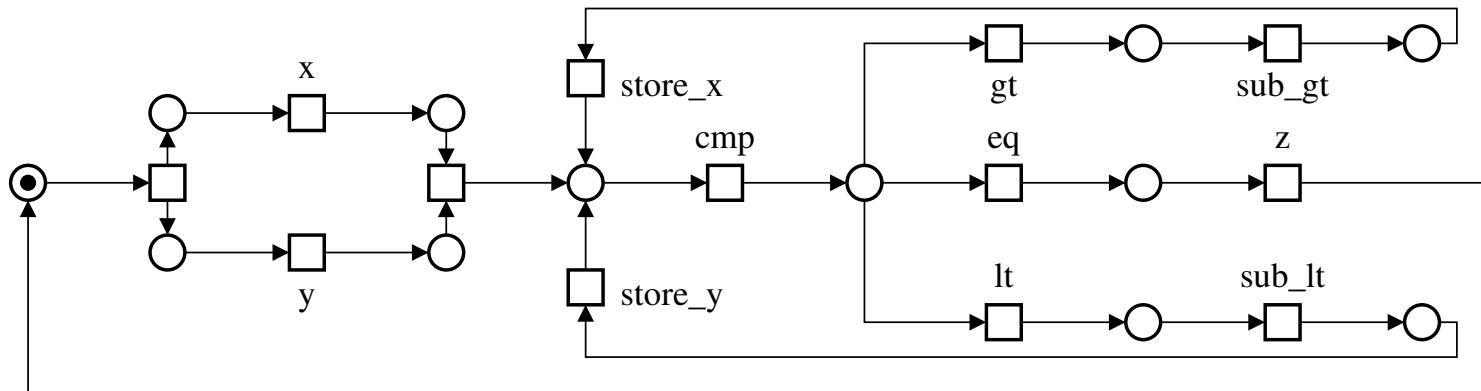
● If-statement refinement



```
x_reg = x; y_reg = y;
while (x_reg != y_reg)
begin
  if (x_reg < y_reg)
    y_reg = y_reg - x_reg;
  else
    x_reg = x_reg - y_reg;
end
z <= x_reg;
```

Global PN for GCD

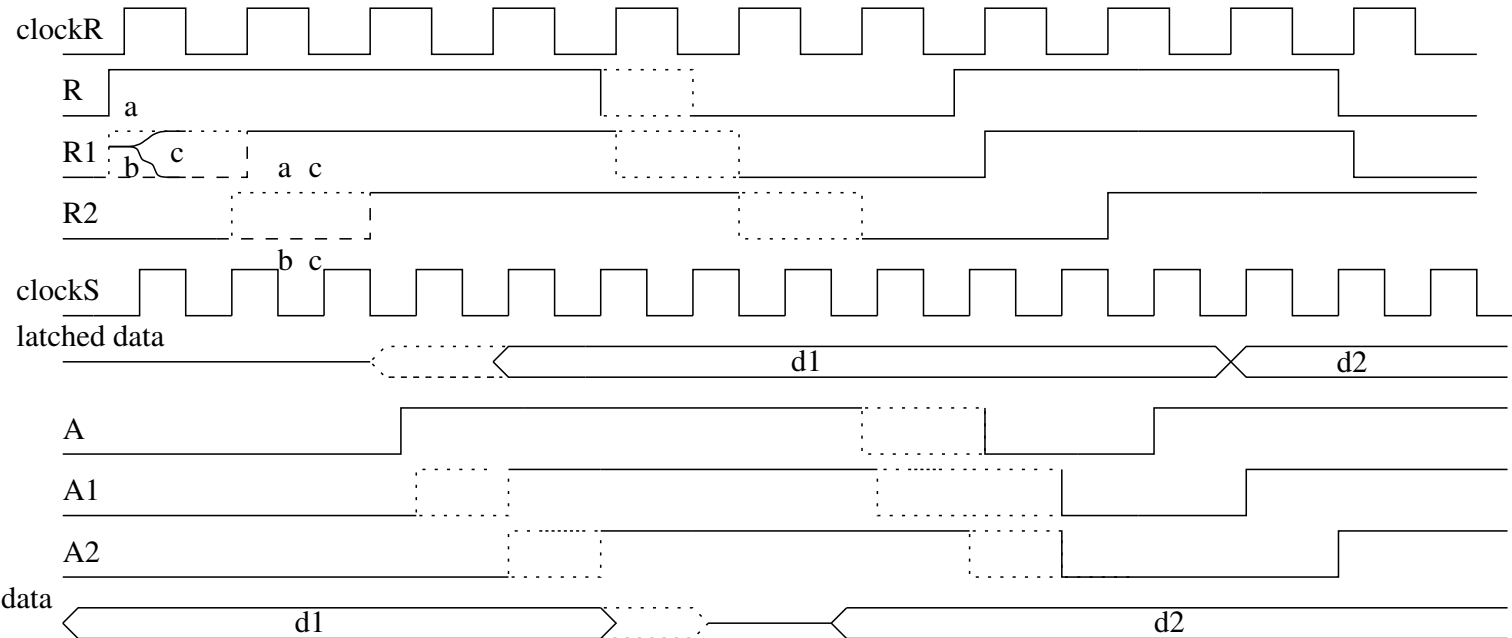
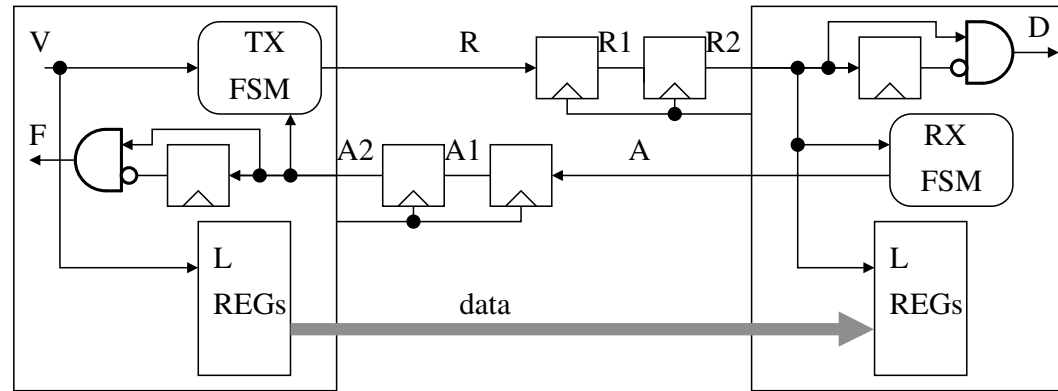
Assignment-operation refinement



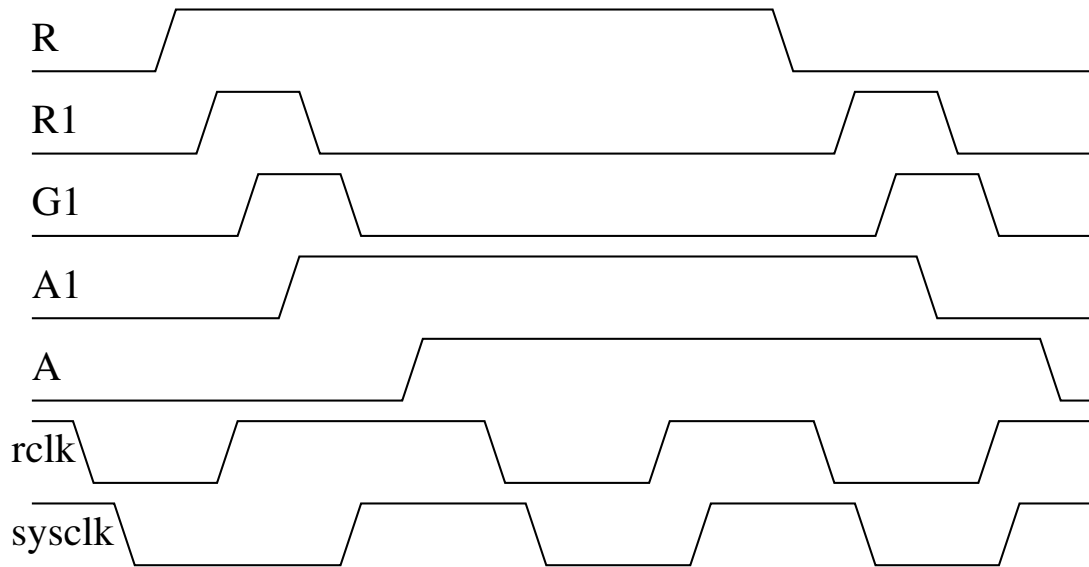
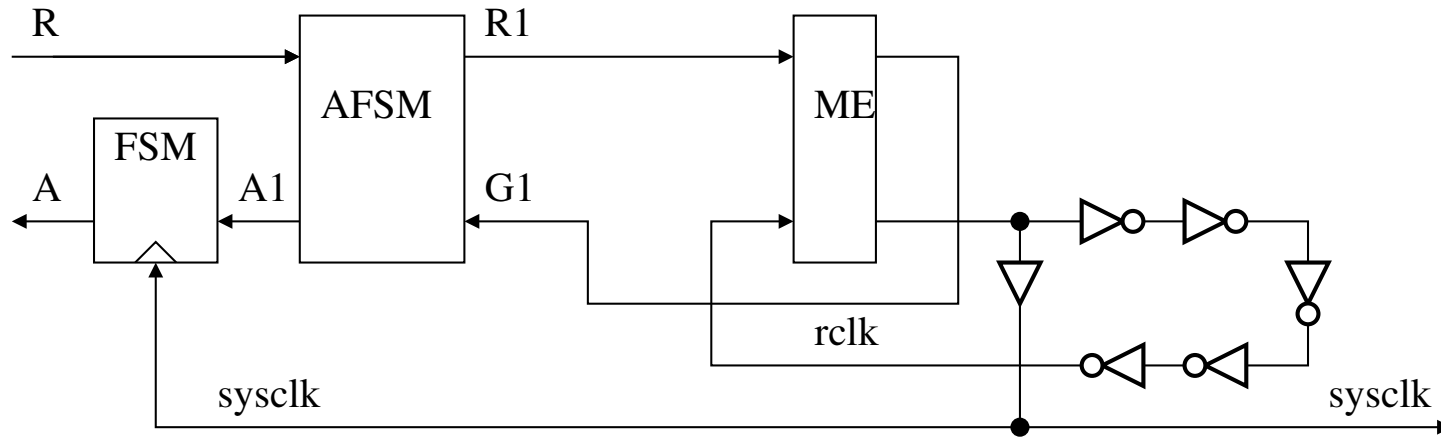
```
x_reg = x; y_reg = y;
while (x_reg != y_reg)
begin
  if (x_reg < y_reg)
    y_reg = y_reg - x_reg;
  else
    x_reg = x_reg - y_reg;
end
z <= x_reg;
```


Background: GALs with fixed clocks

From Ran's paper "Fourteen Ways to Fool Your Synchronizer"

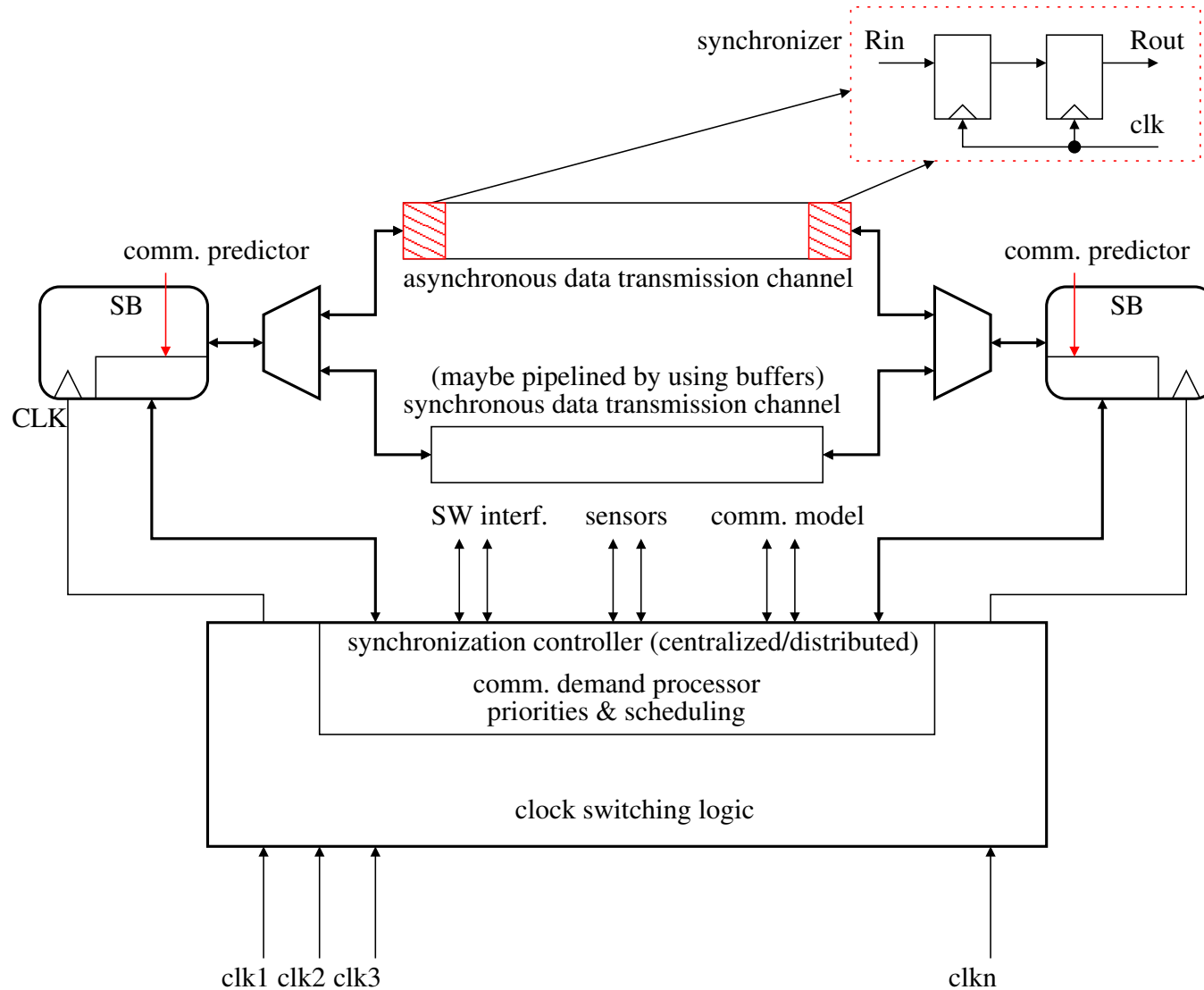


Background: pausable clocking

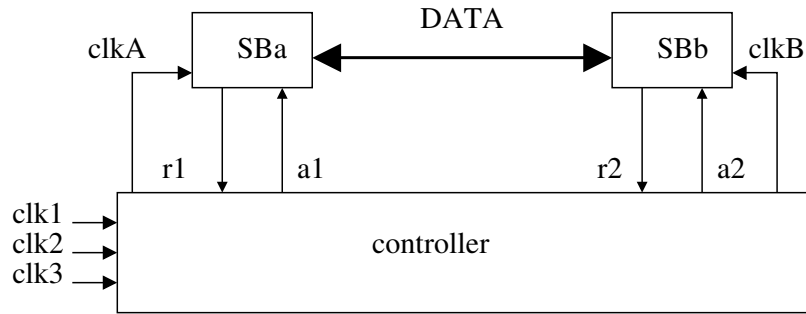


from Kenneth's paper "Pausible Clocking: A First Step Toward Heterogeneous Systems"

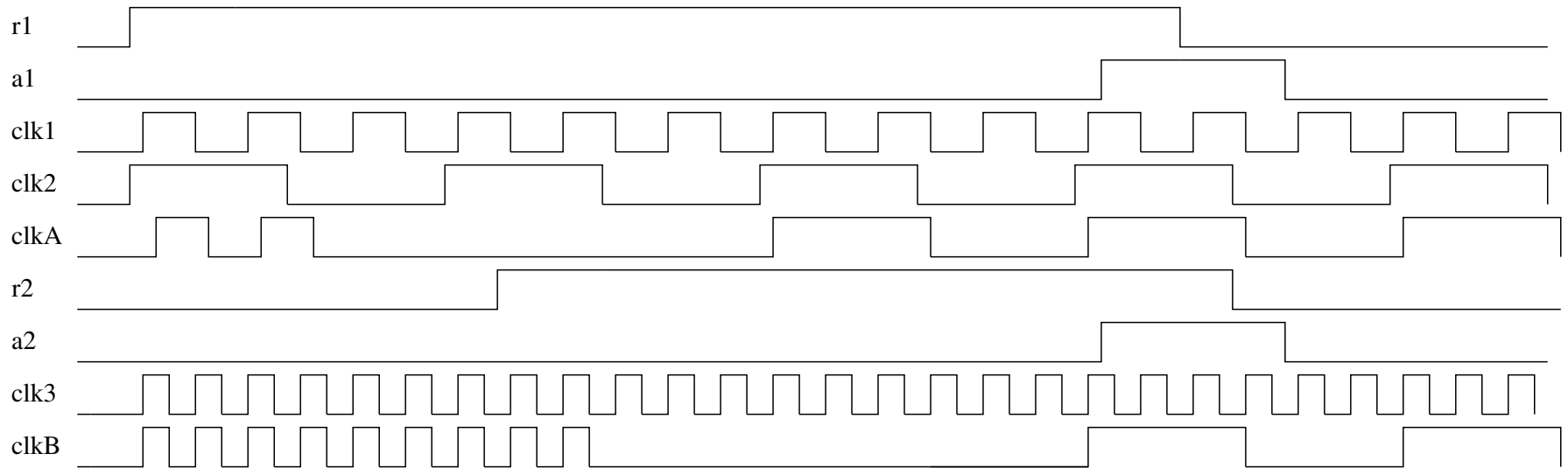
Proposed structure



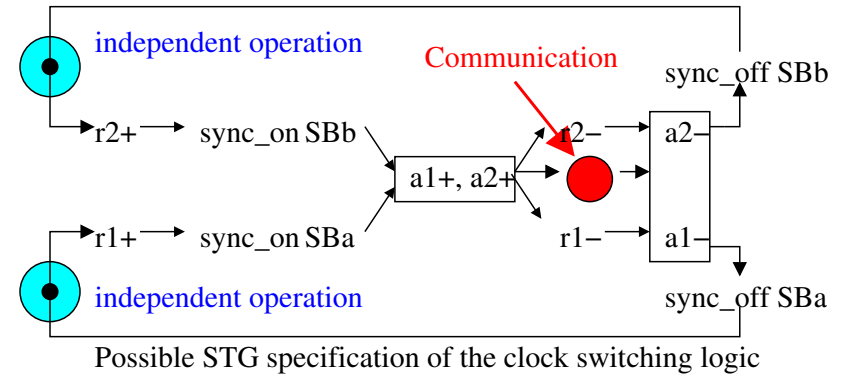
Block-level protocol



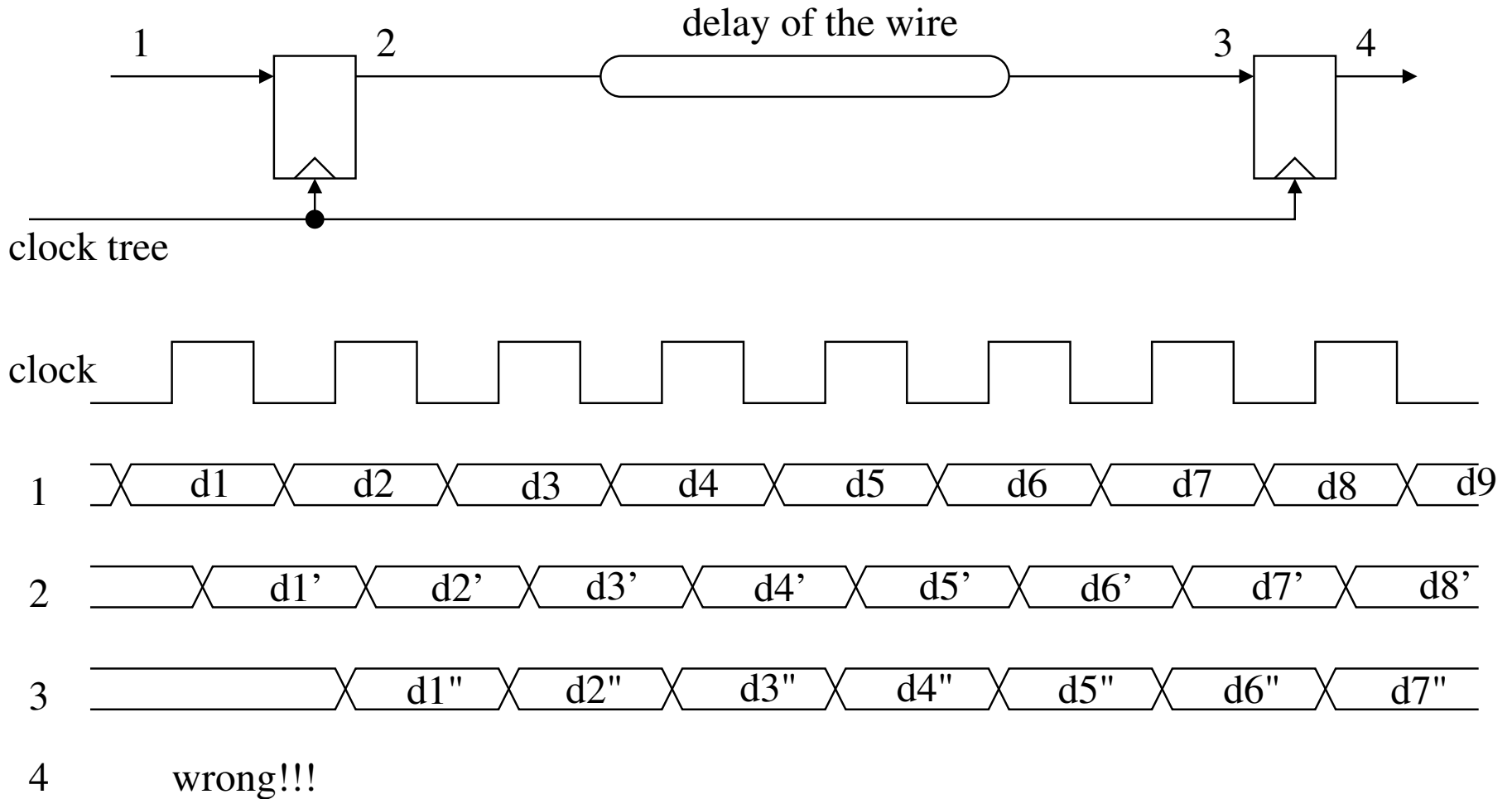
The waveforms of requiring switching



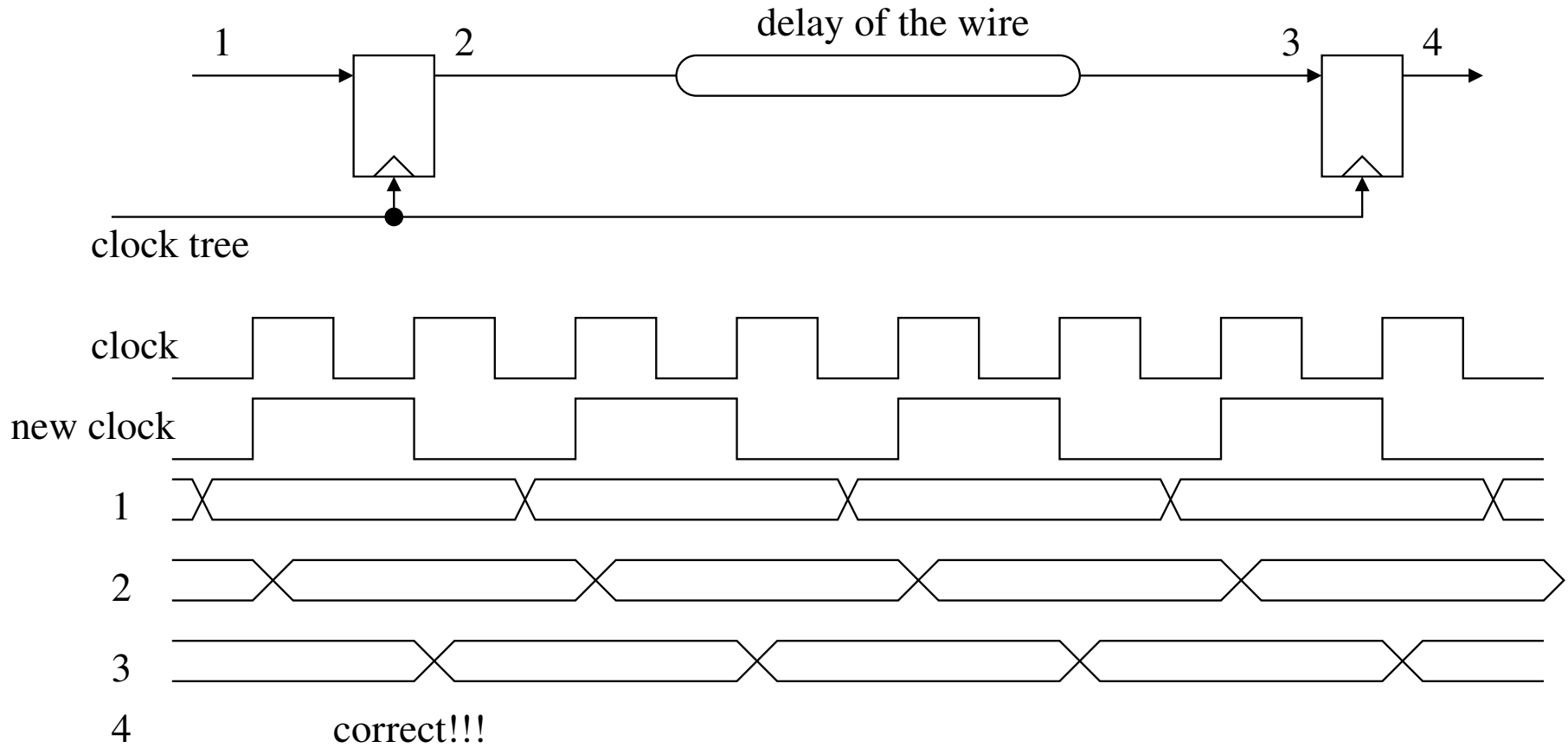
Two synchronisation aspects: clock switching and synchronic distance correction



Faster communication: the problem

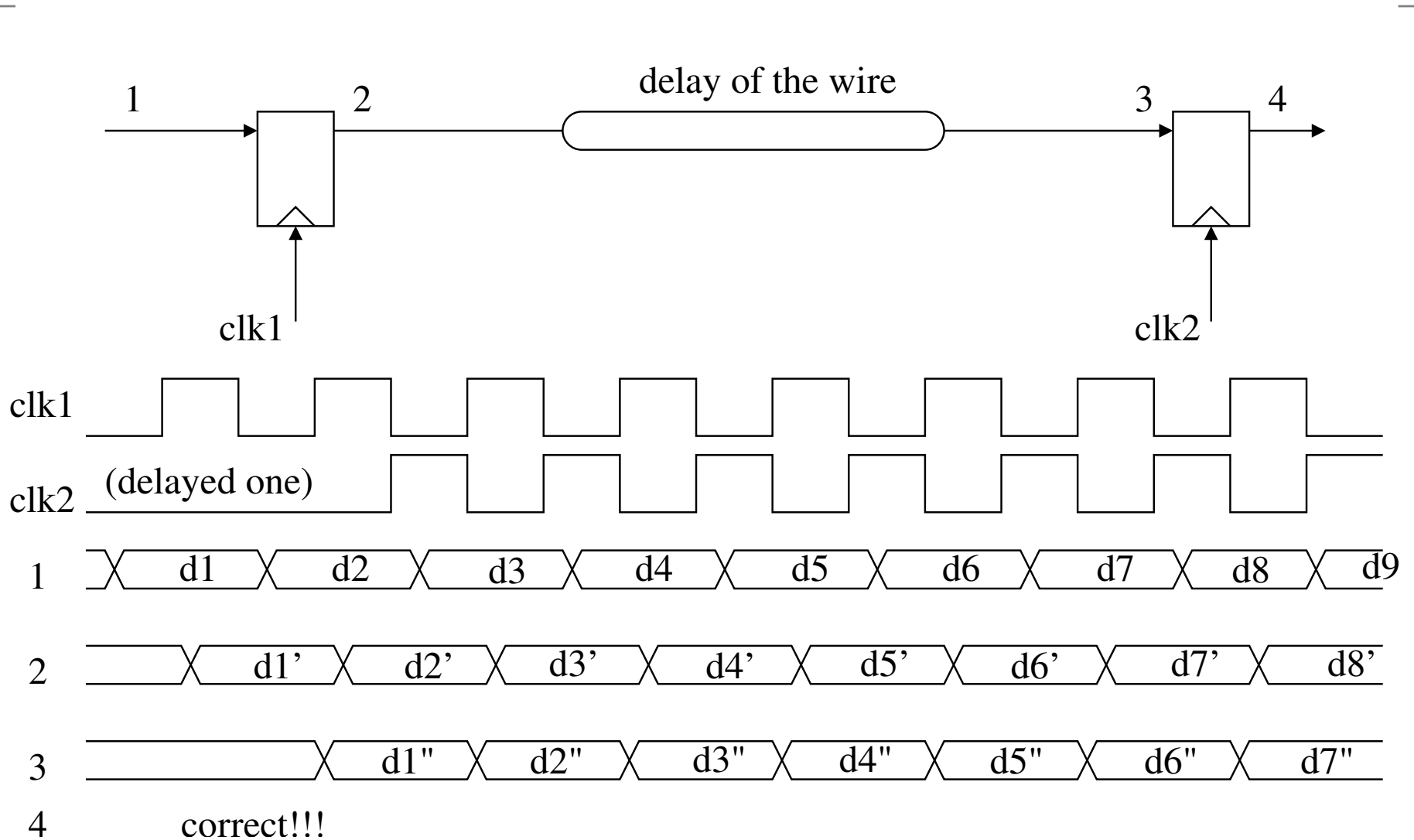


Faster communication: reducing speed



reduce frequency

Faster communication: phase adjustment



Works for uni-directional and buffered duplex channels

Summary (the same as Expectations)

- Obvious features:
 - Synchronous inter-block communication
 - Improved latency and throughput
 - Improved power through the power-performance trade-off
- Not-so-obvious features:
 - Switching between FIXED clocks, no stretching
 - Dynamically formed synchronous clusters
 - Clock is treated as a resource
 - “Intelligent” clock allocation
 - Utilisation of communication info. of all system levels
 - Phase adjustment for the long interconnect, etc.