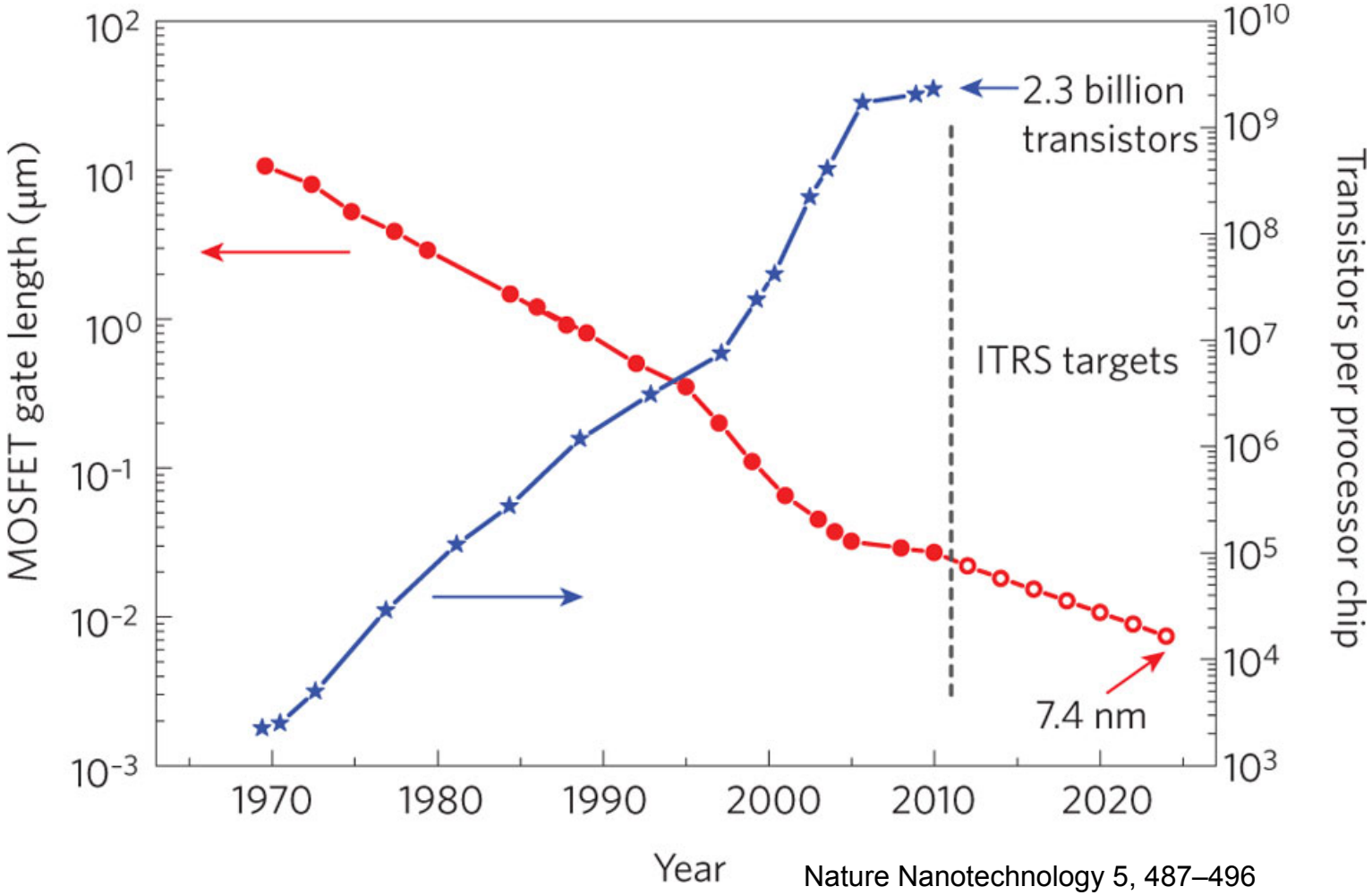
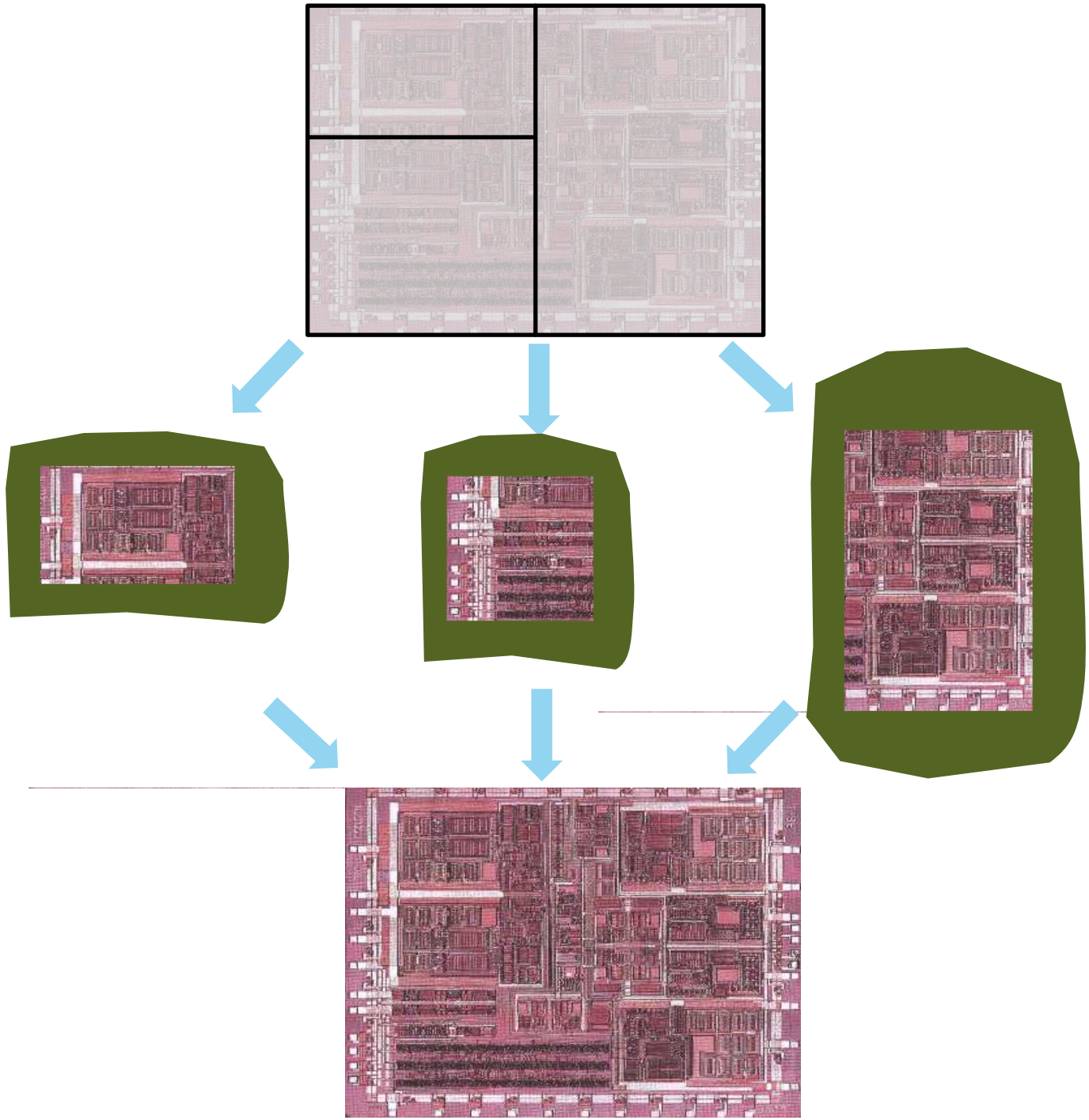
Sign Off Quality Hierarchical Timing Constraints – Wishful Thinking Or Reality

Oleg Levitsky

Cadence Design Systems

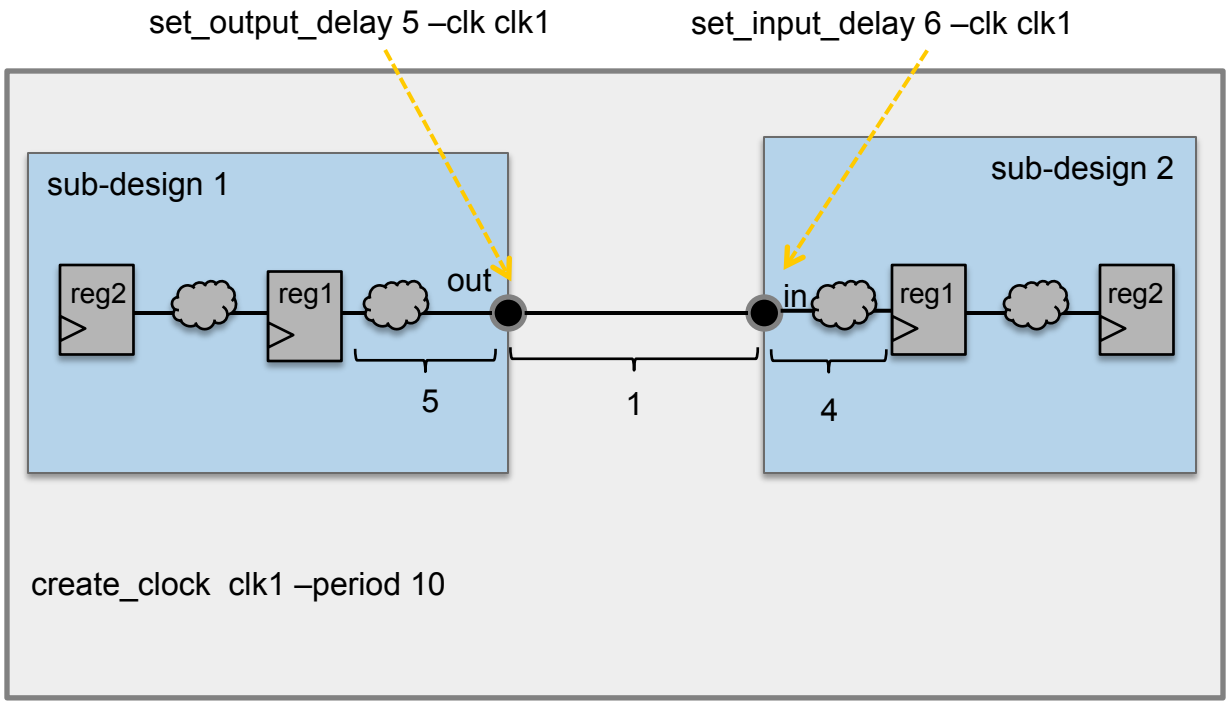


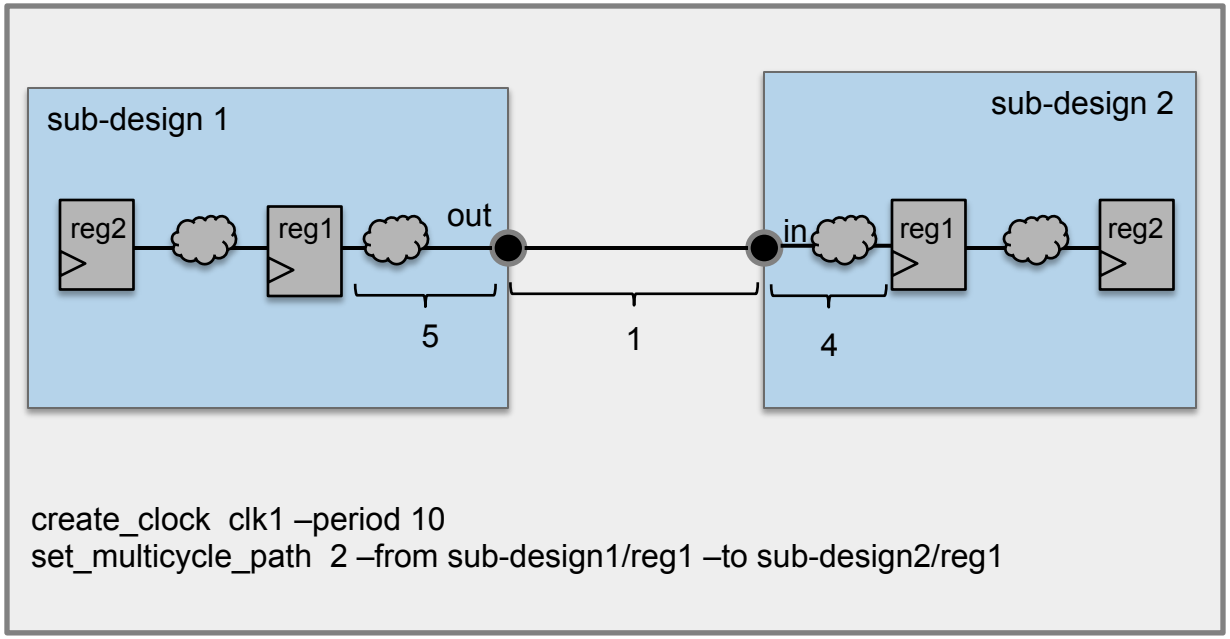
Nature Nanotechnology 5, 487–496 (2010)



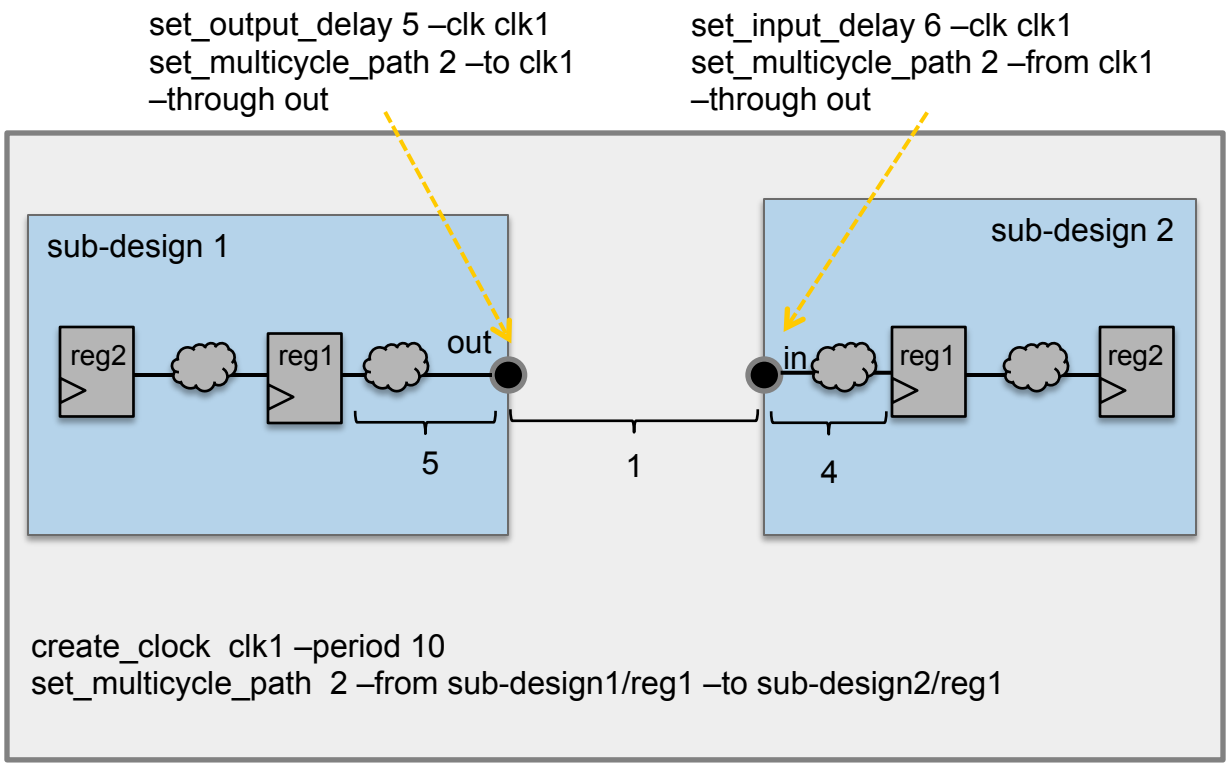
What is the main challenge for timing boundary constraints generation?

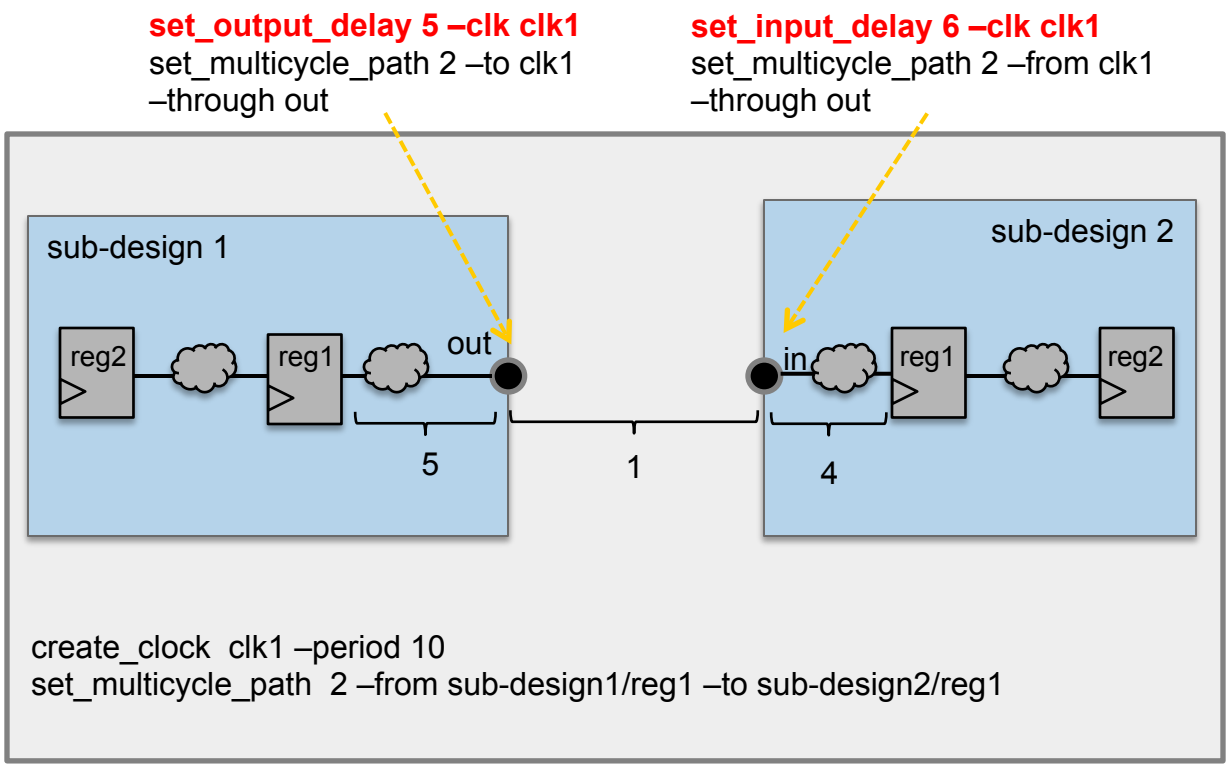
Guarantee identical timing for the same path analyzed on a block and a full chip level


set_output_delay 5 –clk clk1
set_input_delay 6 –clk clk1
sub-design 1
sub-design 2
reg2
reg1
out
in
reg1
reg2
5
1
4
create_clock clk1 –period 10



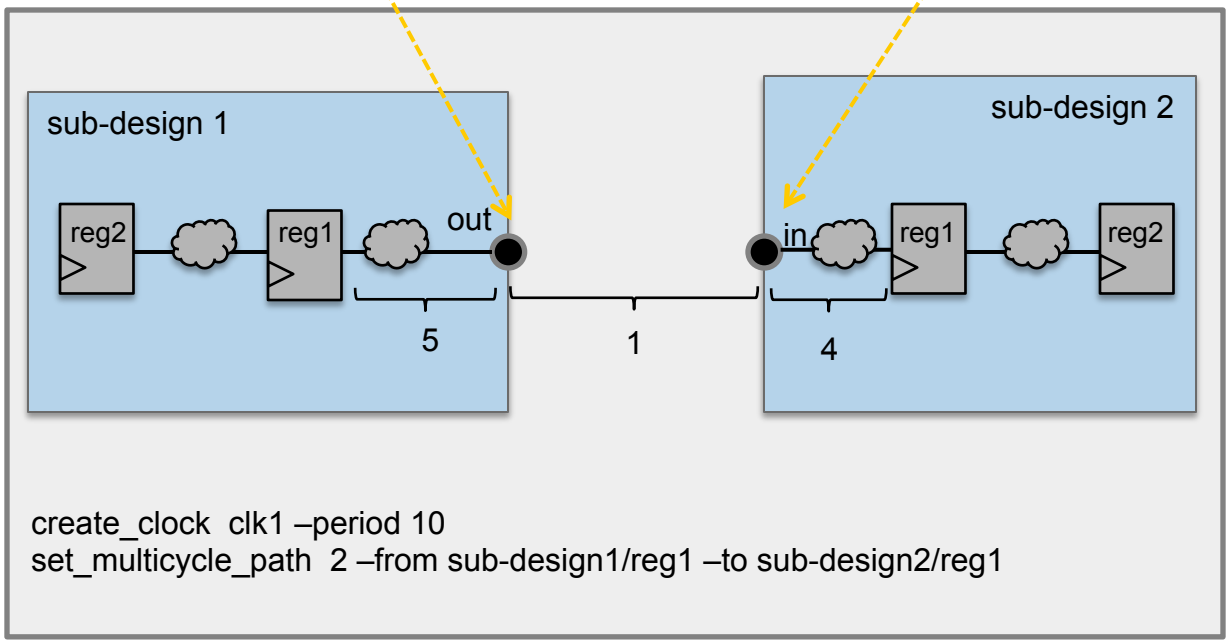

```
create_clock  clk1 –period 10
set_multicycle_path  2 –from sub-design1/reg1 –to sub-design2/reg1
```


set_output_delay 5 –clk clk1
set_multicycle_path 2 –to clk1
–through out
set_input_delay 6 –clk clk1
set_multicycle_path 2 –from clk1
–through out
sub-design 1
sub-design 2
reg2
reg1
out
in
reg1
reg2
5
1
4
create_clock clk1 –period 10
set_multicycle_path 2 –from sub-design1/reg1 –to sub-design2/reg1


set_output_delay 5 –clk clk1
set_multicycle_path 2 –to clk1
–through out
set_input_delay 6 –clk clk1
set_multicycle_path 2 –from clk1
–through out
sub-design 1
sub-design 2
reg2
reg1
out
in
reg1
reg2
5
1
4
create_clock clk1 –period 10
set_multicycle_path 2 –from sub-design1/reg1 –to sub-design2/reg1

set_output_delay 10 –clk clk1
set_multicycle_path 2 –to clk1
–through out

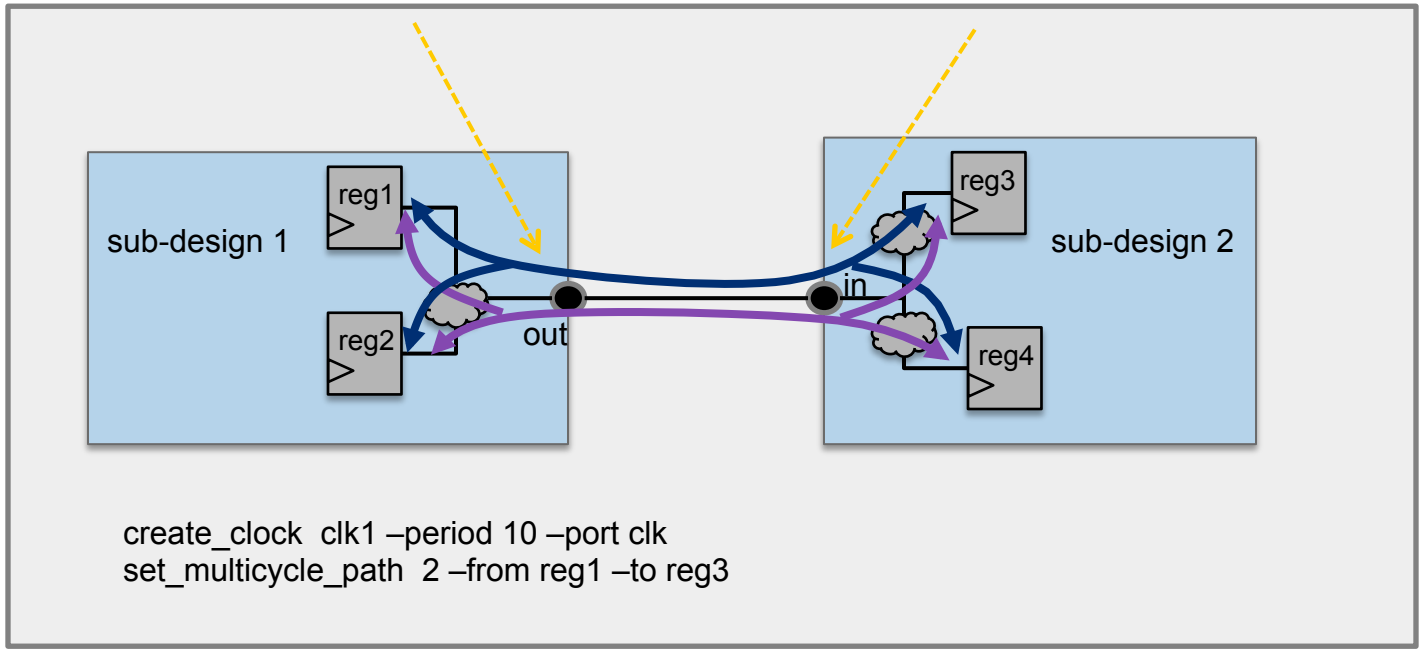
set_input_delay 12 –clk clk1
set_multicycle_path 2 –from clk1
–through out



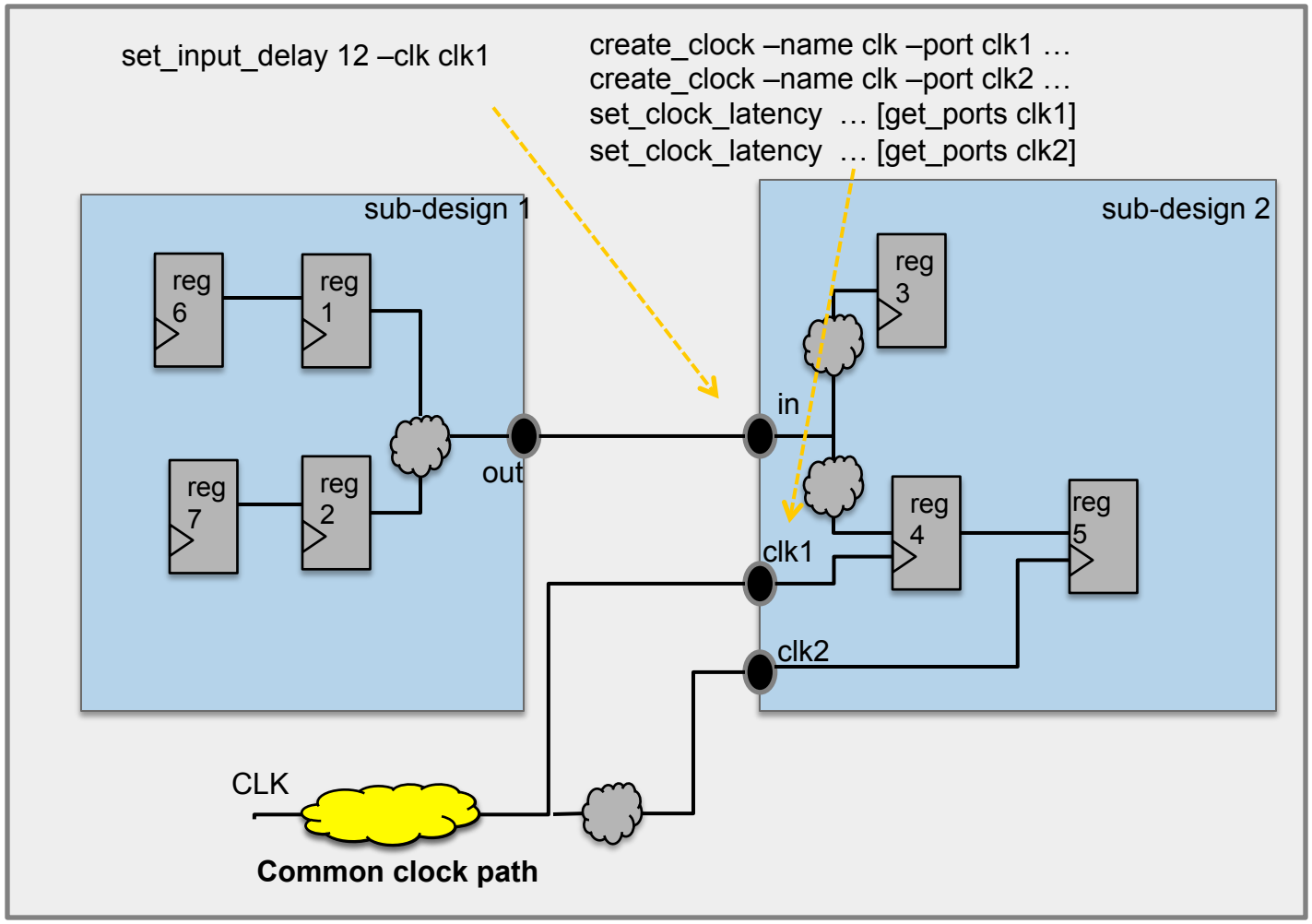
New goal:
convergence on the
full chip level!

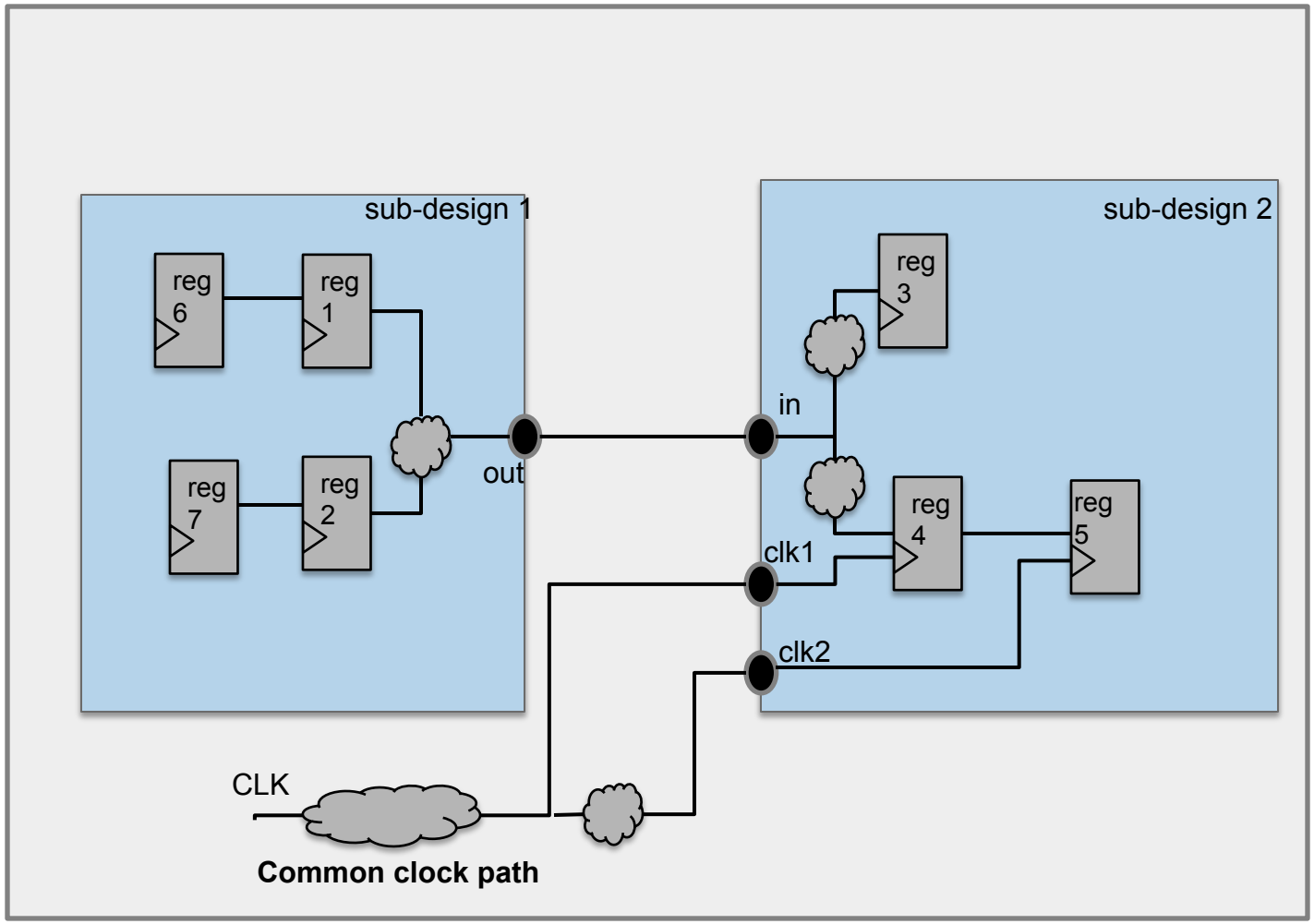
set_output_delay … –clk clk1
set_multicycle_path 2 –to clk1
–through out –from reg 1
create_clock clk_v1
set_output_delay … –clk **clk_v1**

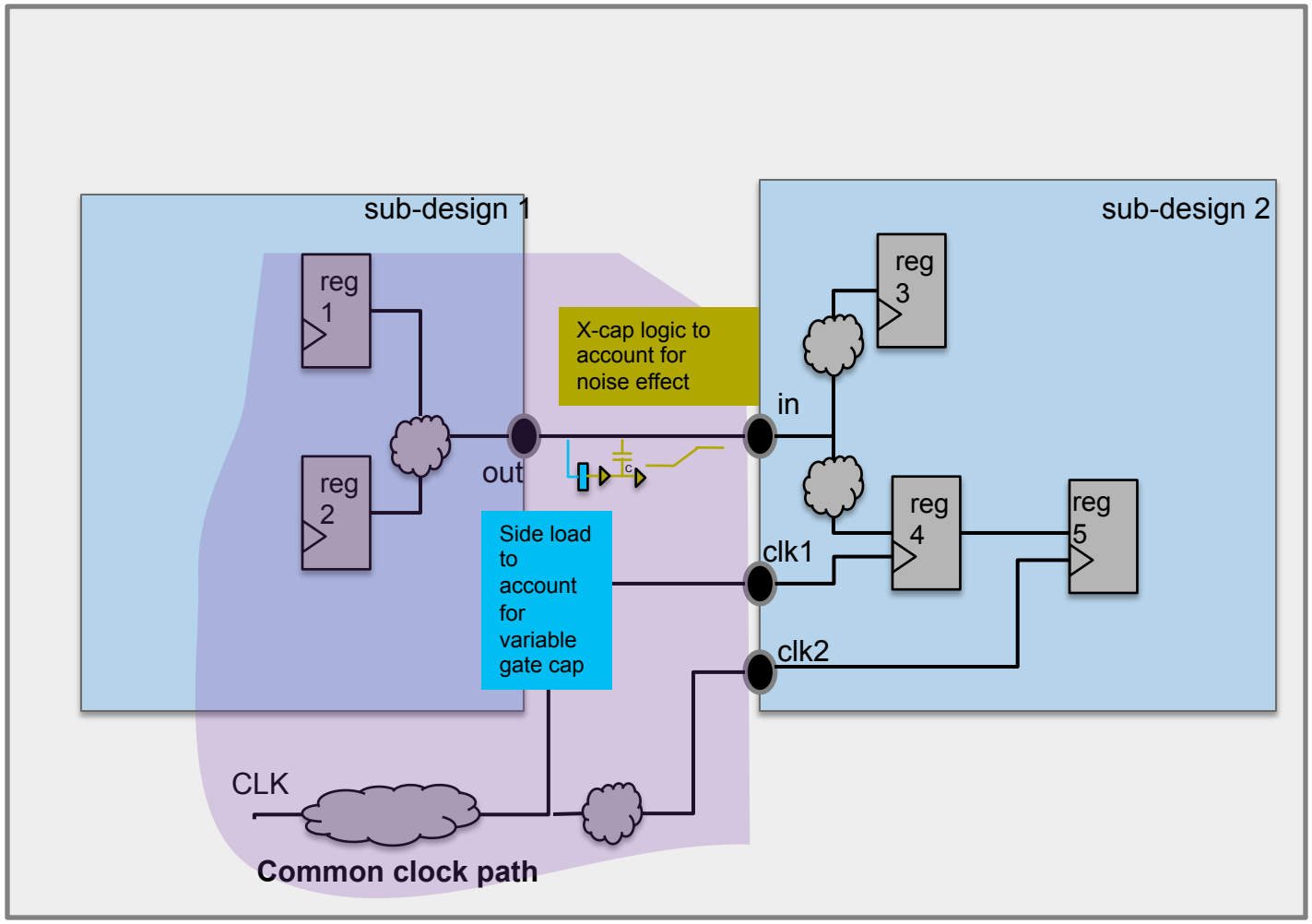
set_input_delay … –clk clk1
set_multicycle_path 2 –from clk1
–through out –to reg3
create_clock clk_v1
set_input_delay … –clk **clk_v1**

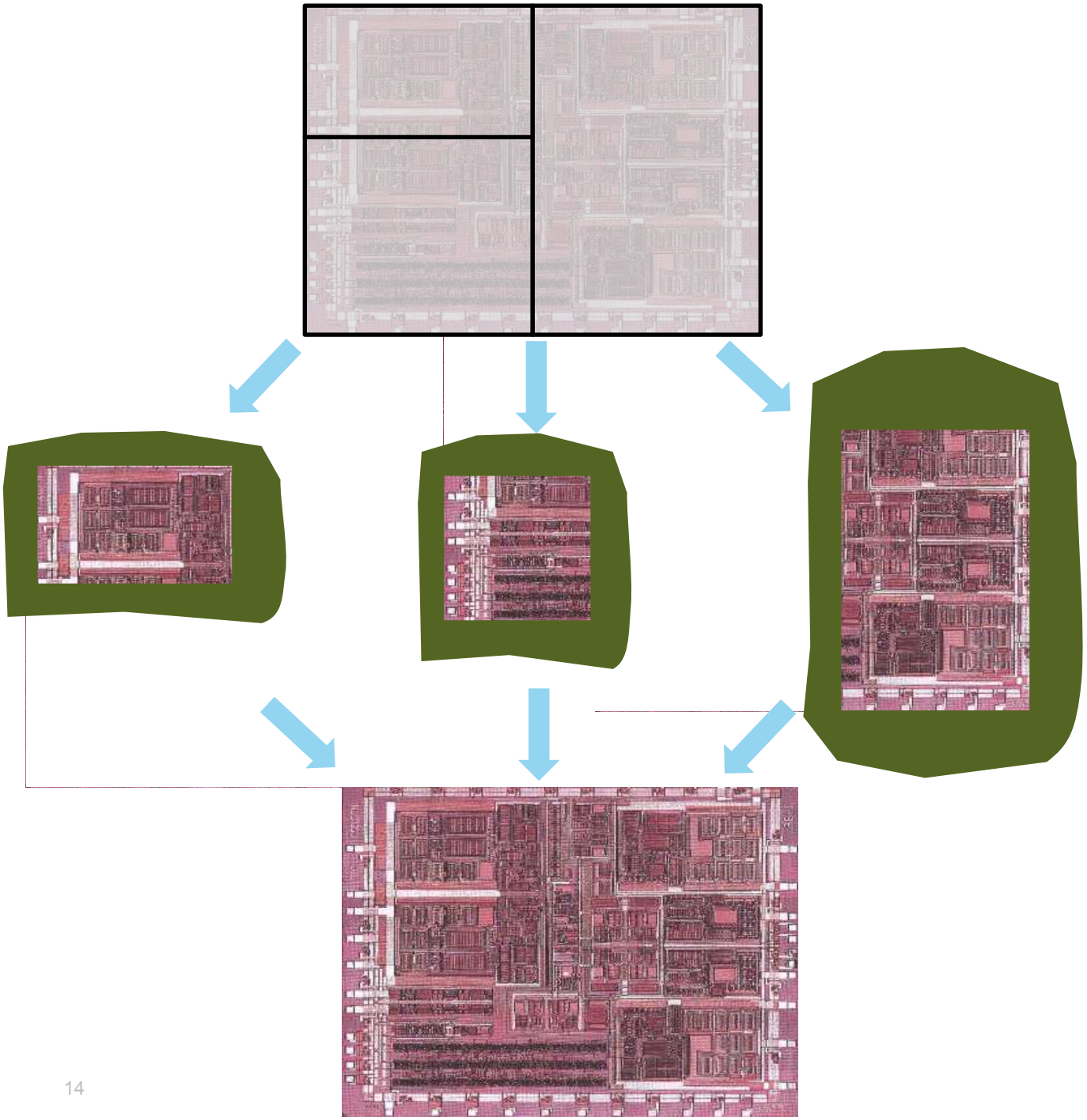


create_clock clk1 –period 10 –port clk
set_multicycle_path 2 –from reg1 –to reg3


set_input_delay 12 –clk clk1
create_clock –name clk –port clk1 …
create_clock –name clk –port clk2 …
set_clock_latency … [get_ports clk1]
set_clock_latency … [get_ports clk2]
sub-design 1
sub-design 2
reg 6
reg 1
reg 7
reg 2
reg 3
reg 4
reg 5
out
in
clk1
clk2
CLK
Common clock path


sub-design 1
sub-design 2
reg 6
reg 1
reg 7
reg 2
out
in
reg 3
reg 4
reg 5
clk1
clk2
CLK
Common clock path


sub-design 1
sub-design 2
reg 1
reg 2
reg 3
reg 4
reg 5
X-cap logic to account for noise effect
in
out
Side load to account for variable gate cap
clk1
clk2
CLK
Common clock path



Proposed Flow

Analyze block boundary paths

Generate boundary sdc constraints

Preserve boundary logic

Hierarchical implementation

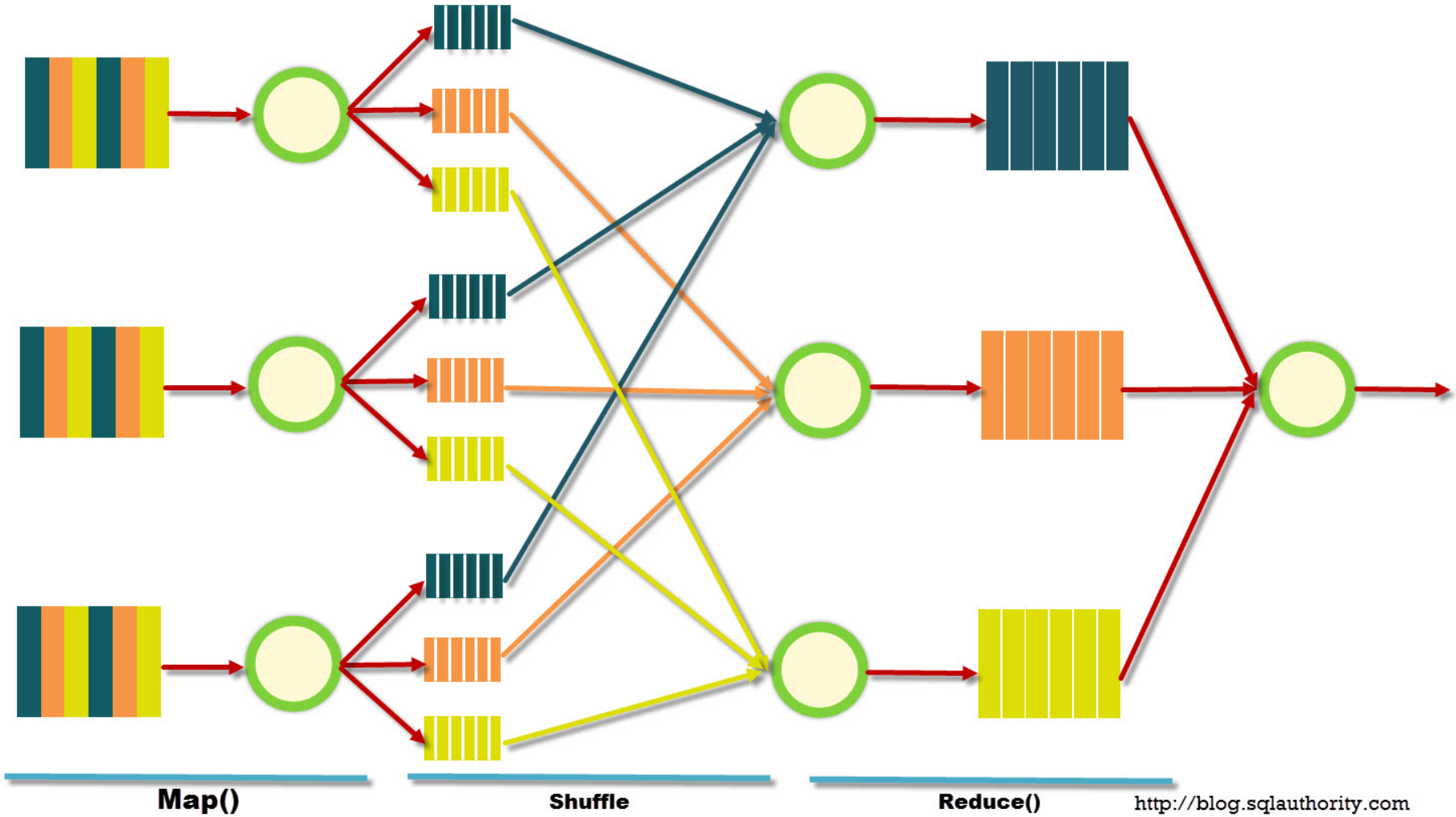
Assemble design & Final Verification (optional)

Experimental Results

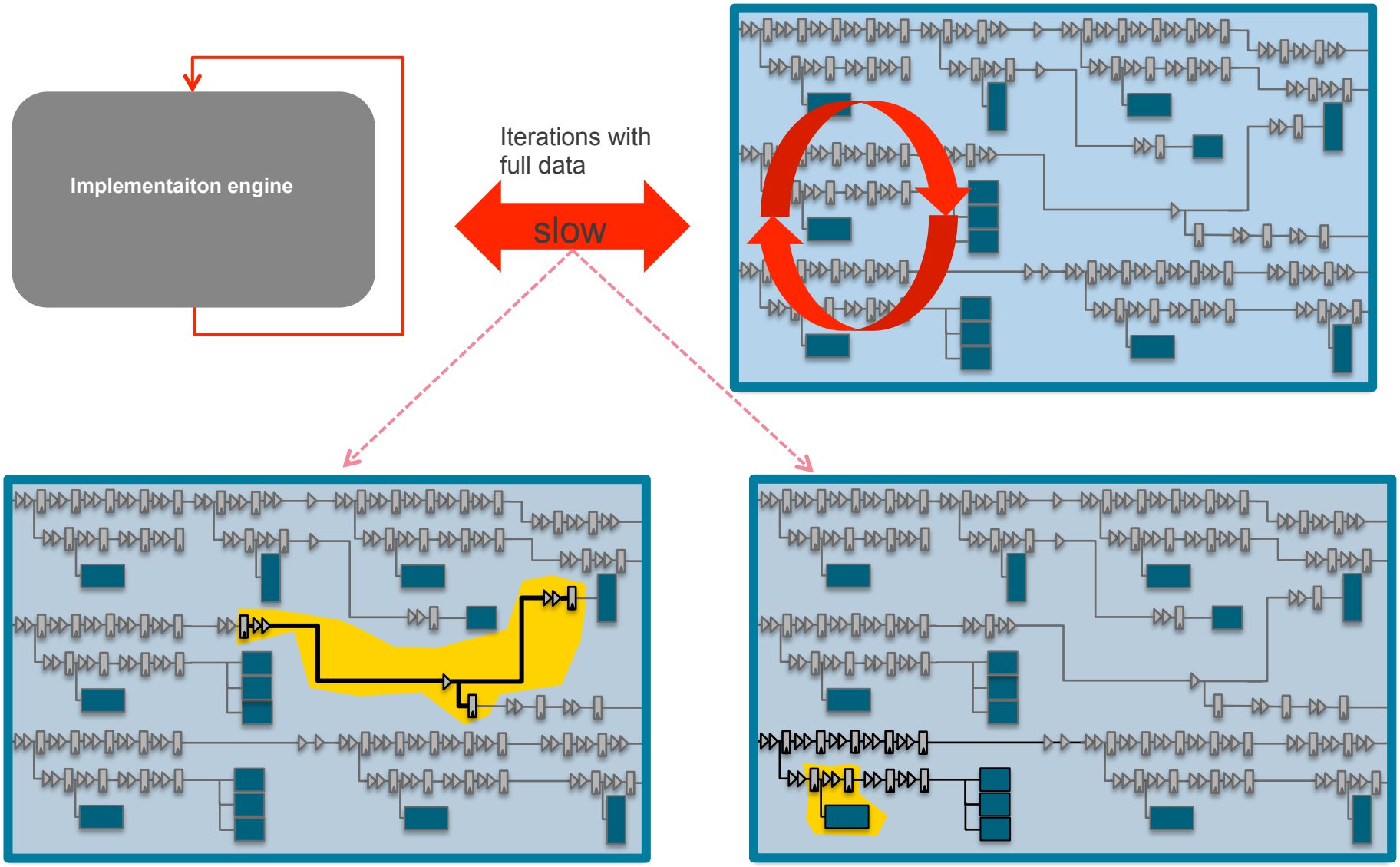
Design	Gate Count (Mgates)	Flat Implementation		Hierarchical Implementation	
		TAT (Hrs)	Peak Mem (GB)	TAT (Hrs)	Peak Mem (GB)
DesignA	8.5	96	16	35	11
Design B	30	52	75	9.5	25

Who cares about accuracy of hierarchical constraints?

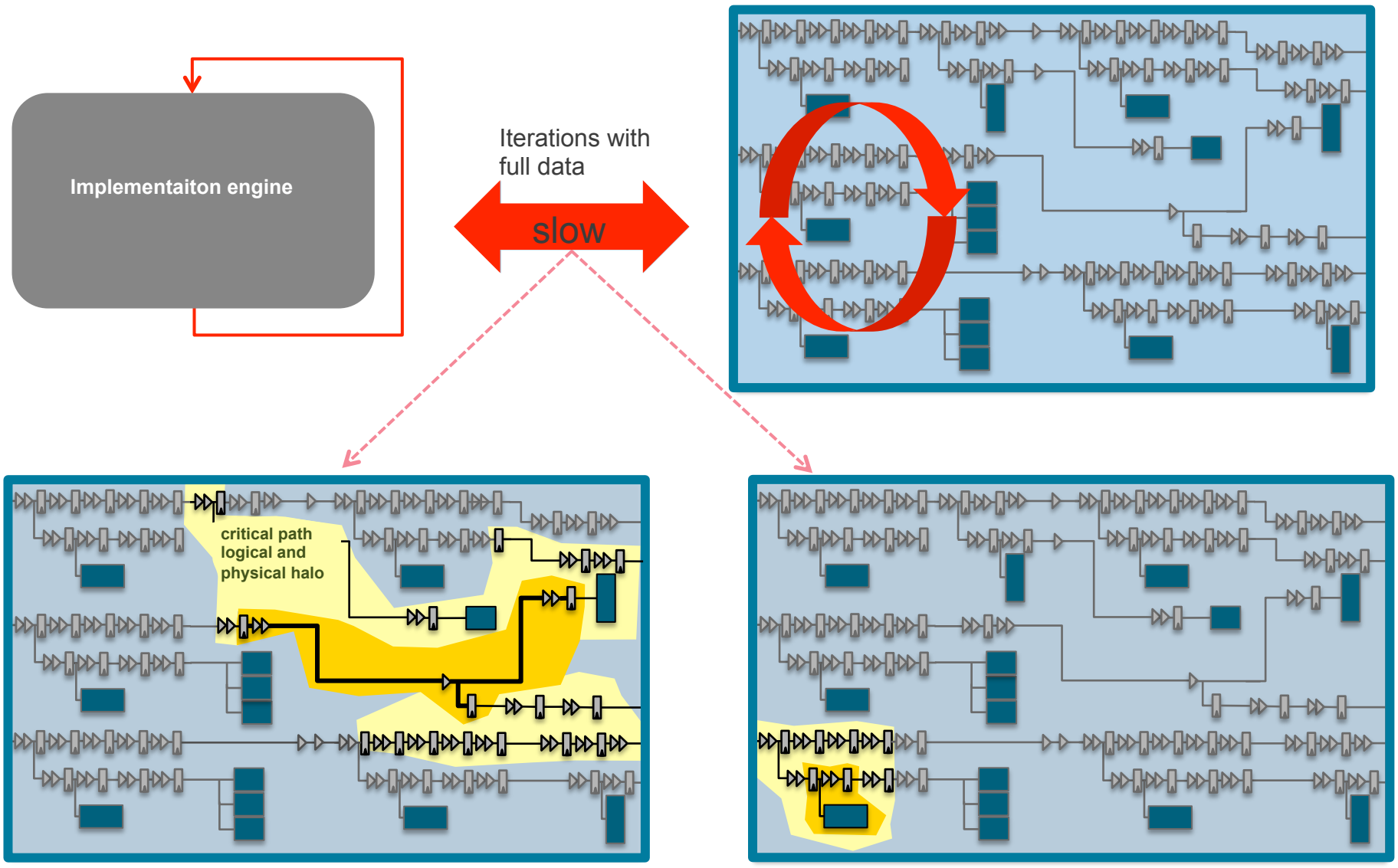
How MapReduce Works?



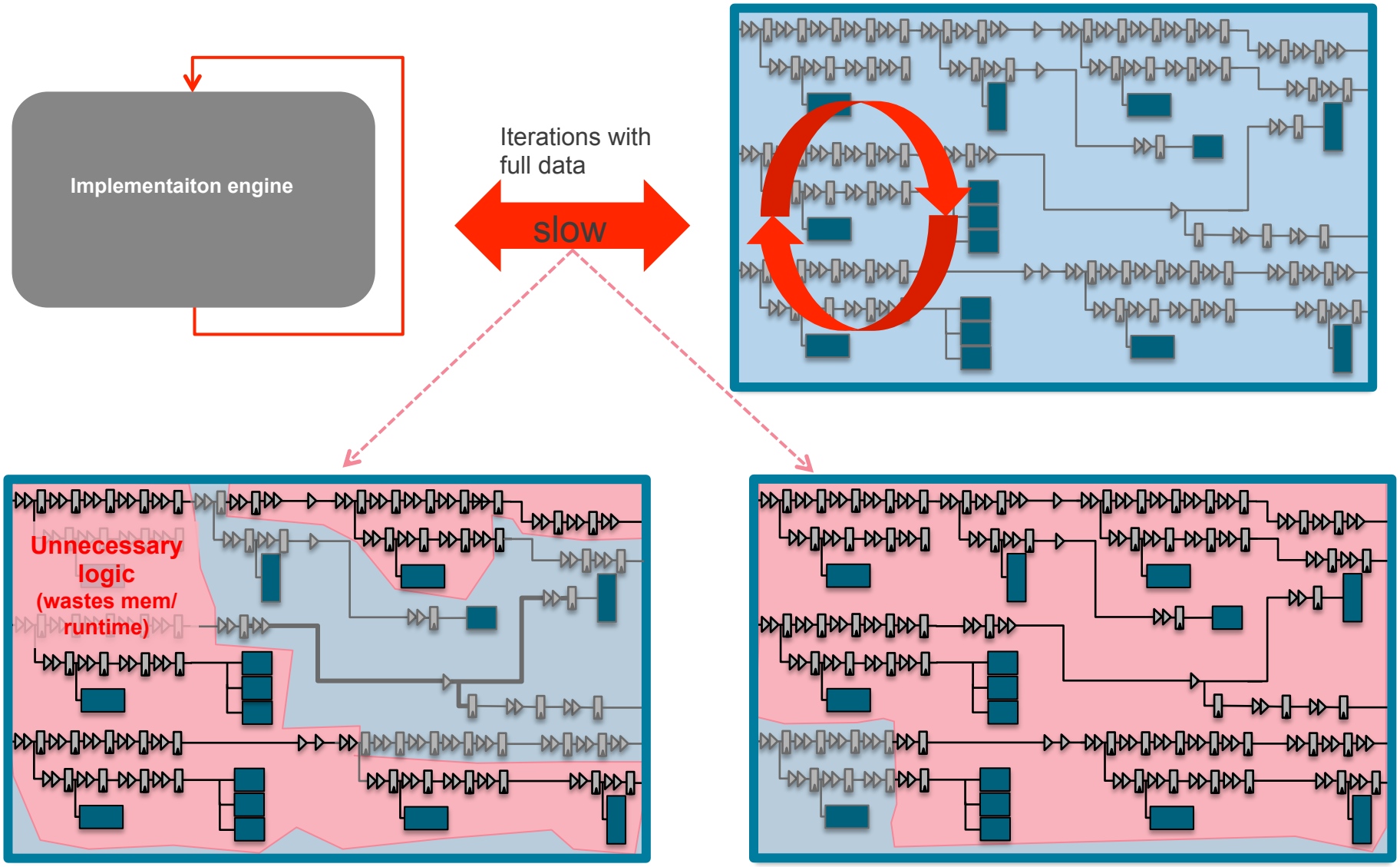
Implementation challenges



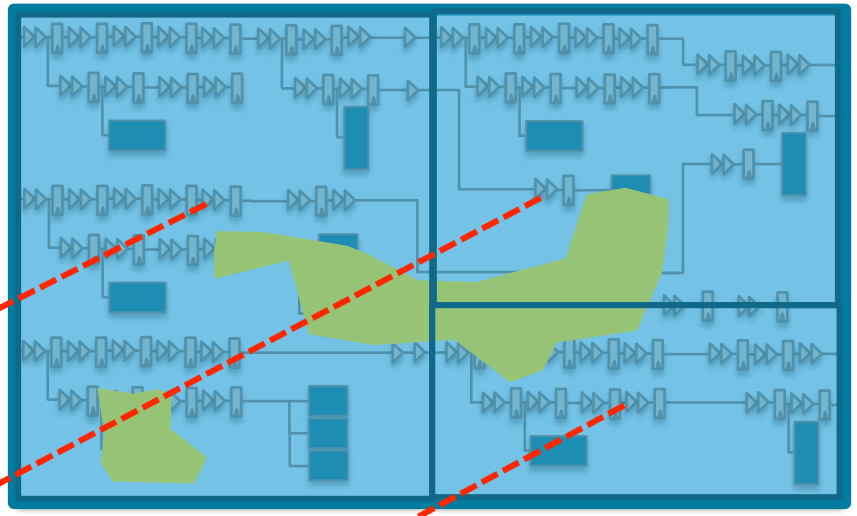
Implementation challenges

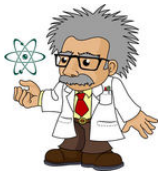


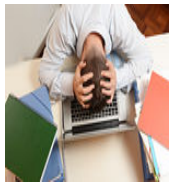
Implementation challenges

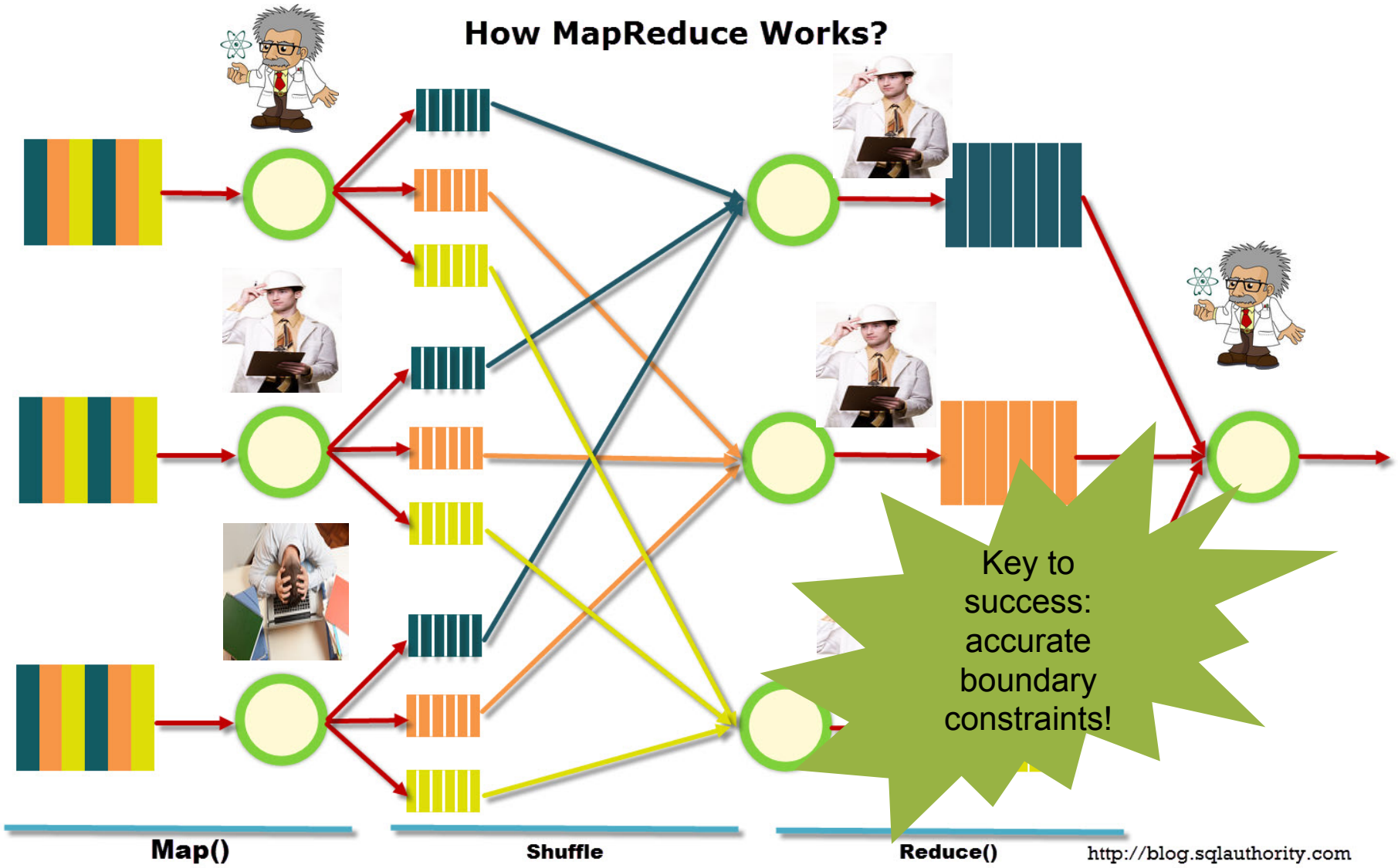


Implementation challenges








How MapReduce Works?
Key to success: accurate boundary constraints!
Map()
Shuffle
Reduce()
http://blog.sqlauthority.com