

Jasper[®] CDC

NYCU-EE IC LAB FALL 2023

Lecturer: Ting-Yu Chang

Agenda

- Structural Analysis
- Functional Checks

Structural Analysis

Check the circuit structure

In our tcl scripts

```
# Find Clock Domains
check_cdc -clock_domain -find
# Find CDC pairs
check_cdc -pair -find
# Find Schemes
check_cdc -scheme -add handshake -module Handshake_syn -map
    {{data din} {sreq sreq} {dreq dreq} {dack dack} {sack sack}}
check_cdc -scheme -add fifo -module FIFO_syn -map
    {{rdata rdata} {wdata wdata} {wptr wptr} {rptr rptr} {wfull wfull} {rempty rempty} {winc winc} {rinc rinc}}
check_cdc -scheme -find
# Find Convergence
check_cdc -group -find
```

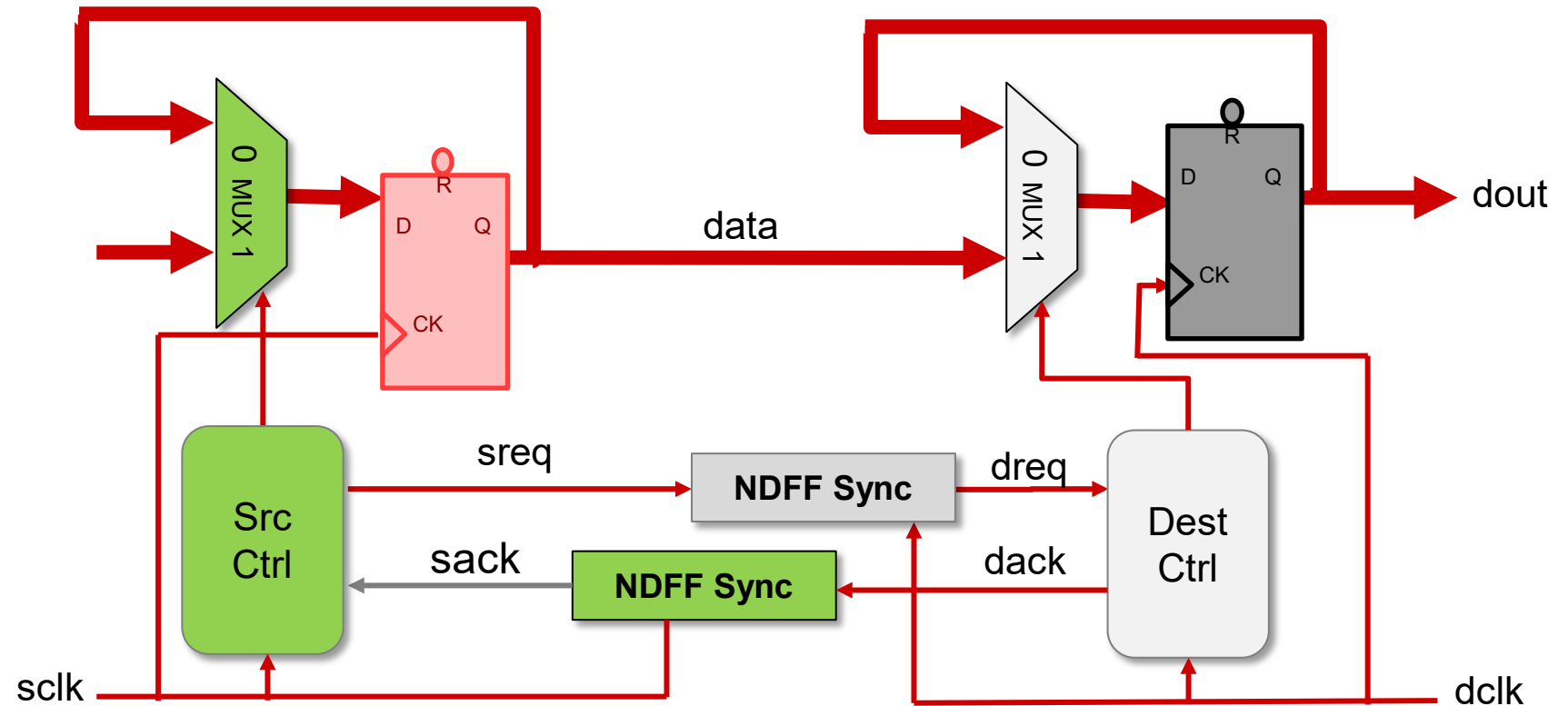
Find Schemes

- In this lab, there are two synchronizers you need to design.
 - Handshake
 - FIFO
- Please use the synchronizer we designed to complete your synchronizer.
 - Use NDFF_syn in Handshake to transfer req / ack signal.
 - Use NDFF_BUS_syn in FIFO to transfer gray-coded pointers.
- We have already map the module name to the pre-defined schemes
 - DO NOT change the module name.
- We have already map the signal to the formal signal
 - DO NOT change the signal name.

Handshake

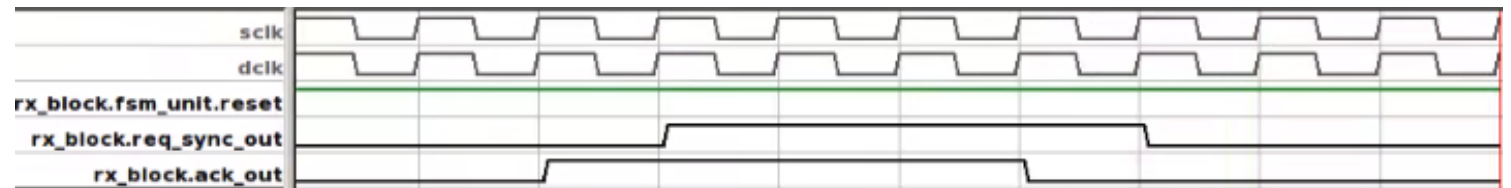
```
check_cdc -scheme -add handshake -module Handshake_syn -map
```

```
{{data din} {sreq sreq} {dreq dreq} {dack dack} {sack sack}}
```

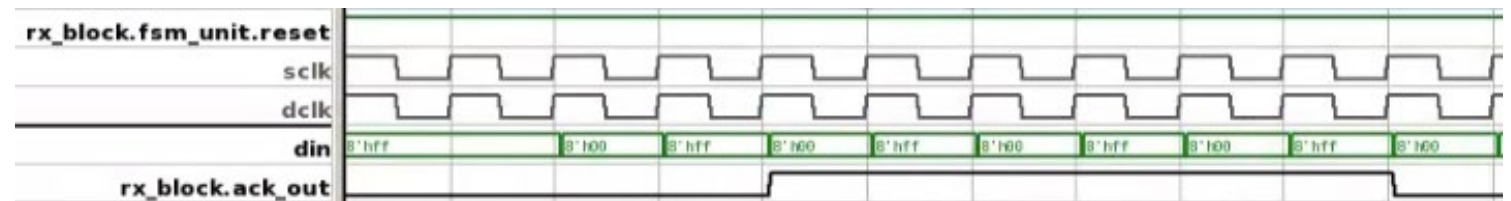


Handshake

- We have design the NDFF synchronizer for you.
- This step checks the following structures.
 - **ACK_WO_SREQ**: This check indicates that the acknowledgment signal in the destination domain is de-asserted before the request signal is de-asserted in the source domain.



- **DAT_HS_STBL**: This check indicates that the data from the sender becomes unstable before it receives an acknowledgment from the receiver.

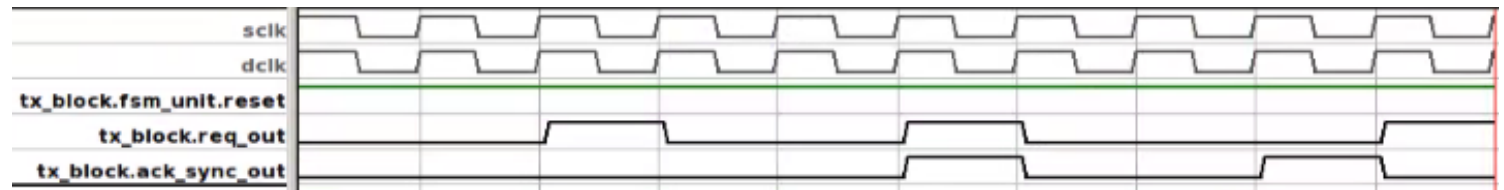


Handshake

- **NAK_WO_SREQ:** This check indicates that the receiver asserts a new acknowledgment before a new request is received.



- **NRQ_WO_DACK:** This check indicates that the sender submits a new request before the acknowledgment for the previous transfer is de-asserted.



Handshake

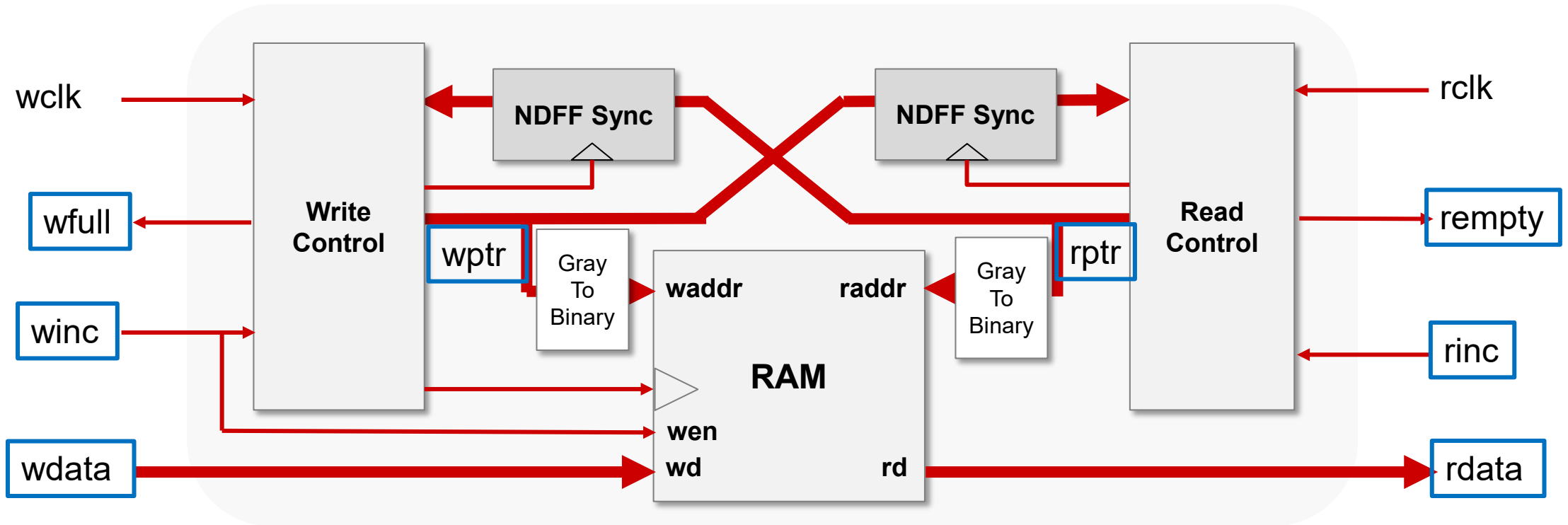
- **REQ_NO_HOLD**: This check indicates that the request signal is de-asserted before receiving acknowledgment from the destination clock domain.



FIFO

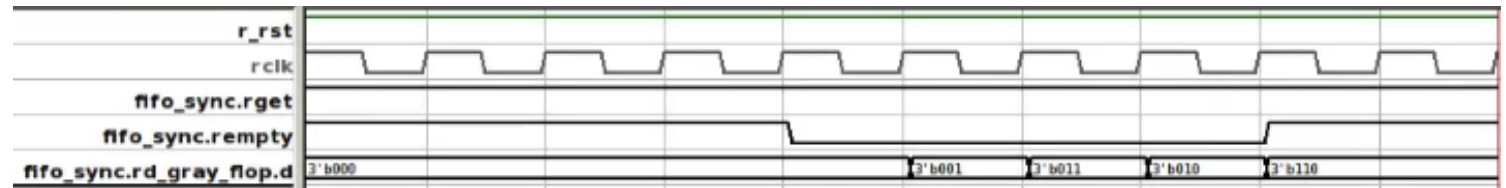
```
check_cdc -scheme -add fifo -module FIFO_syn -map
```

```
{rdata rdata} {wdata wdata} {wptr wptr} {rptr rptr} {wfull wfull} {rempty rempty}  
{winc winc} {rinc rinc}
```



FIFO

- We have design the NDFF_BUS synchronizer for you.
- This step checks the following structures.
 - **POP_ON_EMPTY**: This check indicates that it is possible to read from an empty FIFO.



- **PSH_ON_FULL**: This check indicates that it is possible to write to a full FIFO.

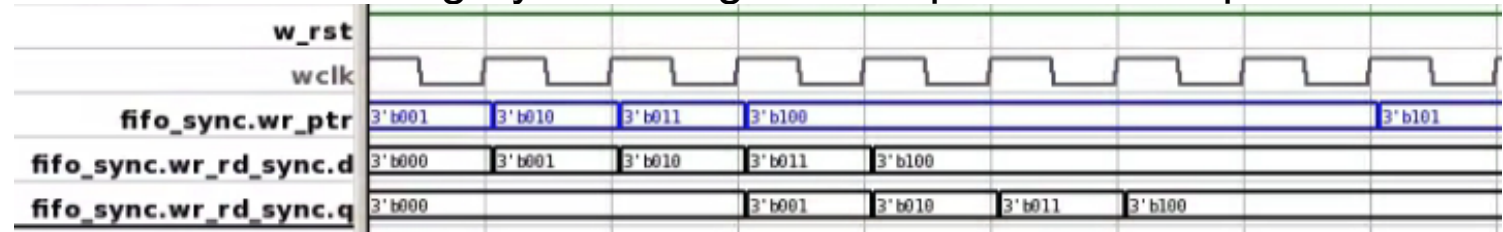


FIFO

- **RPT_NO_GRAY**: This check indicates that the gray encoding for the specified read pointer of the specified FIFO has failed.



- **WPT_NO_GRAY**: This check indicates that the gray encoding for the specified write pointer of the specified FIFO has failed.



- You should add one more stage of the DFF after the read data output by our dual port SRAM to make sure the data synchronize to read clock domain.
 - Or you will get CDC_NO_SYNC violation

After Structure Analysis

- You may have problems with pairs / schemes / convergence.

CDC Phases

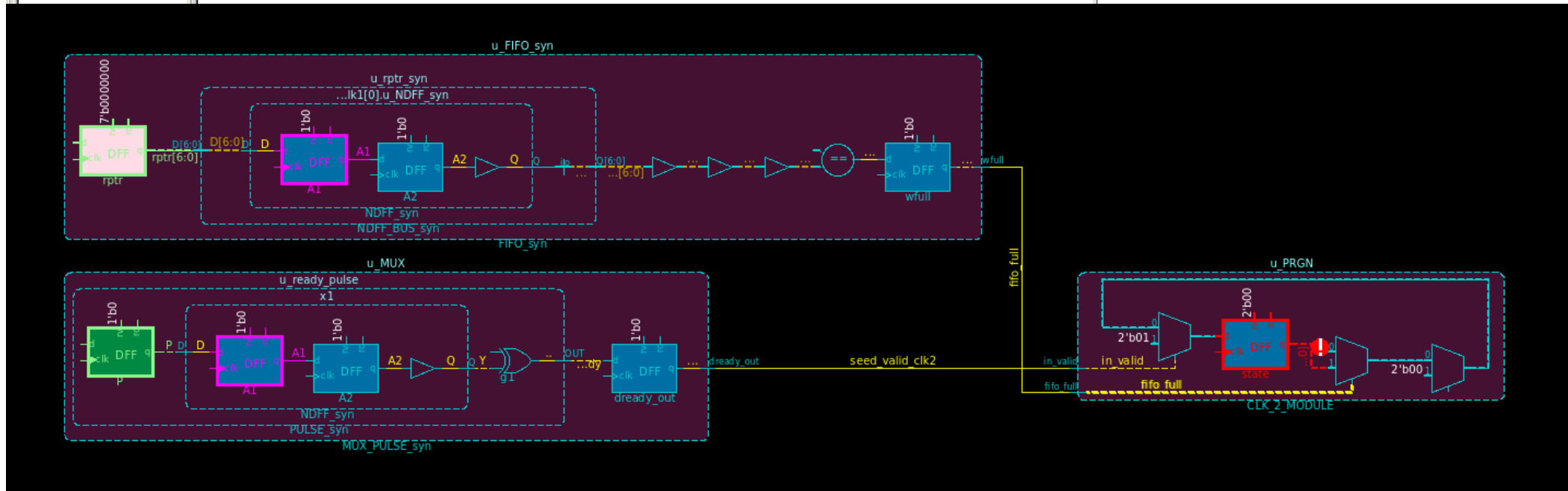
Pairs Schemes Convergence Functional Metastability

All

Convergence

	CDC Group	Type	Same Source Domain	Same Source Signal	FSM	Depth
✖	u_PRGN.state	Convergence	False	False	False	0
✖	u_PRGN.out_cnt	Convergence	False	False	False	0

Double click



After Structure Analysis

- You should pass all check for pairs / schemes / convergence.

The image displays three screenshots of the CDC Phases tool, demonstrating successful verification results for clock pairs, schemes, and convergence.

Top Screenshot: CDC Phases - Pairs

Source Unit	Destination Unit	Sou	De	Lo	Loc	Pair	Tag	Scheme
clk2	u_FIFO_syn.rdata	clk2	clk3	clk2	clk3	Data		u_FIFO_syn.rdata
u_FIFO_syn.rptr[0]	...yn.genblk1[0].u_NDFF_syn.A1	clk3	clk2	clk3	clk2	Control		u_FIFO_syn.rdata
u_FIFO_syn.rptr[1]	...yn.genblk1[1].u_NDFF_syn.A1	clk3	clk2	clk3	clk2	Control		u_FIFO_syn.rdata
u_FIFO_syn.rptr[2]	...yn.genblk1[2].u_NDFF_syn.A1	clk3	clk2	clk3	clk2	Control		u_FIFO_syn.rdata
u_FIFO_syn.rptr[3]	...yn.genblk1[3].u_NDFF_syn.A1	clk3	clk2	clk3	clk2	Control		u_FIFO_syn.rdata
u_FIFO_syn.rptr[4]	...yn.genblk1[4].u_NDFF_syn.A1	clk3	clk2	clk3	clk2	Control		u_FIFO_syn.rdata
u_FIFO_syn.rptr[5]	...yn.genblk1[5].u_NDFF_syn.A1	clk3	clk2	clk3	clk2	Control		u_FIFO_syn.rdata
u_FIFO_syn.rptr[6]	...yn.genblk1[6].u_NDFF_syn.A1	clk3	clk2	clk3	clk2	Control		u_FIFO_syn.rdata
u_FIFO_syn.waddr[5:0]	u_FIFO_syn.rdata	clk2	clk3	clk2	clk3	Data		u_FIFO_syn.rdata
u_FIFO_syn.wfull	u_FIFO_syn.rdata	clk2	clk3	clk2	clk3	Data		u_FIFO_syn.rdata
u_FIFO_syn.wptr[0]	...yn.genblk1[0].u_NDFF_syn.A1	clk2	clk3	clk2	clk3	Control		u_FIFO_syn.rdata
u_FIFO_syn.wptr[1]	...yn.genblk1[1].u_NDFF_syn.A1	clk2	clk3	clk2	clk3	Control		u_FIFO_syn.rdata
u_FIFO_syn.wptr[2]	...yn.genblk1[2].u_NDFF_syn.A1	clk2	clk3	clk2	clk3	Control		u_FIFO_syn.rdata

Bottom Left Screenshot: CDC Phases - Schemes

Scheme	Scheme Type	Detection Type	Tag
u_Handshake_syn.din	Handshake	UserDefined	
u_FIFO_syn.rdata	FIFO	UserDefined	

Bottom Right Screenshot: CDC Phases - Convergence

Source Unit	Destination Unit	Source Clock Domain	Destination Clock Domain	Local Source Clock	Local Destination Clock	Pair Class
dshake_syn.dack	u_Handshake_syn.u_ack_syn.A1	clk2	clk1	clk2	clk1	
dshake_syn.data	u_Handshake_syn.u_req_syn.A1	clk1	clk2	clk1	clk2	
N.state	u_FIFO_syn.rdata	clk2	clk3	clk2	clk3	
N.temp	u_FIFO_syn.rdata	clk2	clk3	clk2	clk3	

Functional Analysis

Check the usage of the synchronizers

In our tcl scripts

```
## ----- Functional Checks ----- ##  
check_cdc -protocol_check -generate  
check_cdc -protocol_check -prove
```


Functional Analysis

- In this lab, there are two synchronizers you need to design.
 - Handshake
 - MUX_PULSE
 - FIFO
- You must abide by these synchronizer usage rules.
- In this step, JG will use formal verification to prove your circuit follows the rule (pre-defined assertion) we introduce in structure analysis.
- You will learn the formal verification in the later lab, at that time you will write your own assertion and use JG to prove it.

Handshake

- **ACK_WO_SREQ:**

- `@(posedge u_Handshake_syn.dclk) disable iff (~u_Handshake_syn.rst_n) (u_Handshake_syn.dack) && (u_Handshake_syn.dreq) | => (u_Handshake_syn.dack)`
- At every positive edge of the dclk clock signal, if the reset RST_N is high, then if the dack and dreq is high at the destination domain, dack must be high at the next cycle.

- **DAT_HS_STBL:**

- `@(posedge u_Handshake_syn.dclk) disable iff (~u_Handshake_syn.rst_n) (u_Handshake_syn.dreq && !(u_Handshake_syn.dack)) | => $stable(u_Handshake_syn.din)`
- At every positive edge of the dclk clock signal, if the reset RST_N is high, and if the dreq is high and dack is low at the destination domain, then din must be stable at the next cycle.

Handshake

- **NAK_WO_SREQ:**

- `@(posedge u_Handshake_syn.dclk) disable iff (~u_Handshake_syn.rst_n) !(u_Handshake_syn.dack) && !(u_Handshake_syn.dreq) | => !(u_Handshake_syn.dack)`
- At every positive edge of the dclk clock signal, if the reset RST_N is high, then if the dack and dreq is low at the destination domain, dack must be low at the next cycle.

- **NRQ_WO_DACK:**

- `@(posedge u_Handshake_syn.sclk) disable iff (~u_Handshake_syn.rst_n) (!(u_Handshake_syn.sreq) && u_Handshake_syn.sack) | => !(u_Handshake_syn.sreq)`
- At every positive edge of the sclk clock signal, if the reset RST_N is high, and if the sreq is low and sack is high at the source domain, then sreq must be low at the next cycle.

Handshake

- **REQ_NO_HOLD:**
 - `@(posedge u_Handshake_syn.sclk) disable iff (~u_Handshake_syn.rst_n) (u_Handshake_syn.sreq && !(u_Handshake_syn.sack)) | => (u_Handshake_syn.sreq)`
 - At every positive edge of the sclk clock signal, if the reset RST_N is high, then if the sreq is high and sack is low at the source domain, sreq must be high at the next cycle.

FIFO

- **POP_ON_EMPTY:**

- `@(posedge u_FIFO_syn.rclk) u_FIFO_syn.rinc |-> !(u_FIFO_syn.empty)`
- At every positive edge of the rclk clock signal, if rinc is high, then empty should be low at that same clock cycle.

- **PSH_ON_FULL:**

- `@(posedge u_FIFO_syn.wclk) u_FIFO_syn.winc |-> !(u_FIFO_syn.wfull)`
- At every positive edge of the wclk clock signal, if winc is high, then wfull should be low at the same clock cycle.

FIFO

- **RPT_NO_GRAY:**

- `@(posedge u_FIFO_syn.rclk) disable iff (~u_FIFO_syn.rst_n) ##1 $changed(u_FIFO_syn.rptr) |-> ($onehot(u_FIFO_syn.rptr ^ $past(u_FIFO_syn.rptr)))`
- At every positive edge of the rclk clock signal, if the reset rst_n is high, if the rptr changes value in the next clock cycle, then exactly one bit must change in rptr (The result of performing a bitwise XOR between the value of the rptr in the next clock cycle and its value in the current clock cycle must yield a one-hot encoded value).

- **WPT_NO_GRAY:**

- `@(posedge u_FIFO_syn.wclk) disable iff (~u_FIFO_syn.rst_n) ##1 $changed(u_FIFO_syn.wptr) |-> ($onehot(u_FIFO_syn.wptr ^ $past(u_FIFO_syn.wptr)))`
- At every positive edge of the wclk clock signal, if the reset rst_n is high, if the wptr changes value in the next clock cycle, then exactly one bit must change in wptr

After Functional Analysis

- You should pass all check for Functional / Metastability.

CDC Phases

Pairs Schemes Convergence Functional Metastability

All Checks

- clk2 -> clk3
- clk3 -> clk2
- clk1 -> clk2
- clk2 -> clk1

Scheme Property	Tag	Scheme	Scheme Type	Detection Type
...O_syn.rdata_no_write_on_full	PSH_ON_FULL	u_FIFO_syn.rdata	FIFO	UserDefined
...syn.rdata_no_read_on_empty	POP_ON_EMPTY	u_FIFO_syn.rdata	FIFO	UserDefined
...O_syn.rdata_wptr_gray_coded	WPT_NO_GRAY	u_FIFO_syn.rdata	FIFO	UserDefined
...O_syn.rdata_rptr_gray_coded	RPT_NO_GRAY	u_FIFO_syn.rdata	FIFO	UserDefined
...dshake_syn.dreq_data_stable	CTL_NO_STBL	u_Handshake_syn.dreq	NDFF	Automatic
...dshake_syn.sack_data_stable	CTL_NO_STBL	u_Handshake_syn.sack	NDFF	Automatic
...ndshake_syn.din_src_req_hold	REQ_NO_HOLD	u_Handshake_syn.din	Handshake	UserDefined
...ndshake_syn.din_src_new_req	NRQ_WO_DACK	u_Handshake_syn.din	Handshake	UserDefined
...shake_syn.din_dest_ack_hold	ACK_WO_SREQ	u_Handshake_syn.din	Handshake	UserDefined
...shake_syn.din_dest_new_ack	NAK_WO_SREQ	u_Handshake_syn.din	Handshake	UserDefined
...e_syn.din_data_stability_dest	DAT_HS_STBL	u_Handshake_syn.din	Handshake	UserDefined

Total: 11 Filtered: 11

CDC Configuration CDC Phases Waivers

CDC Phases

Pairs Schemes Convergence Functional Metastability

All Tasks

- <embedded> 0:0:0:0
- <CDC_scheme_propert... 11:0:0:0

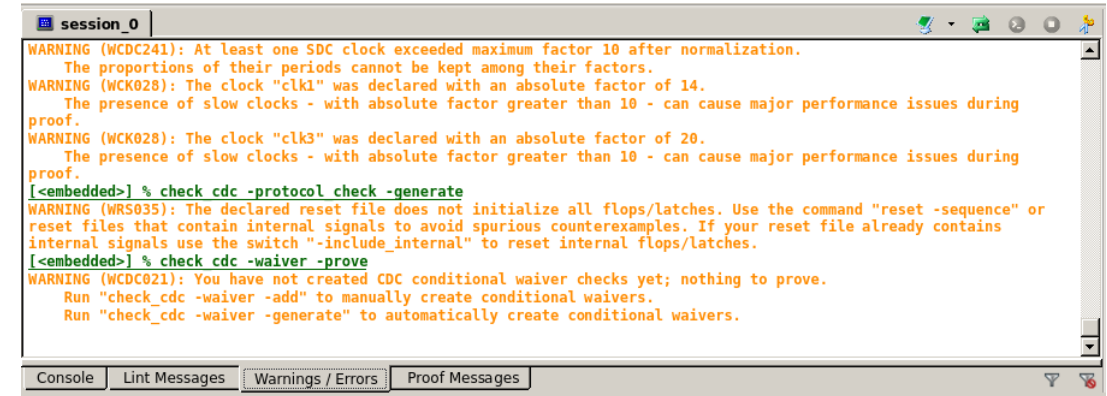
Type	Name	Engine	Bound
Assert	CDC_u_FIFO_syn.rdata_no_write_on_full	PRE	Infinite
Assert	CDC_u_FIFO_syn.rdata_no_read_on_empty	PRE	Infinite
Assert	CDC_u_FIFO_syn.rdata_wptr_gray_coded	Hp (13)	Infinite
Assert	CDC_u_FIFO_syn.rdata_rptr_gray_coded	Hp (41)	Infinite
Assert	CDC_u_Handshake_syn.dreq_data_stable	Hp (8)	Infinite
Assert	CDC_u_Handshake_syn.sack_data_stable	Hp (62)	Infinite
Assert	CDC_u_Handshake_syn.din_src_req_hold	Hp (15)	Infinite
Assert	CDC_u_Handshake_syn.din_src_new_req	Hp (15)	Infinite
Assert	CDC_u_Handshake_syn.din_dest_ack_hold	Hp (7)	Infinite
Assert	CDC_u_Handshake_syn.din_dest_new_ack	Hp (7)	Infinite
Assert	CDC_u_Handshake_syn.din_data_stability_dest	Hp (103)	Infinite

Total: 11 Filtered: 11 Selected: 0 Validity: 11:0:0:0 Run: 0:0:0:11 Total: 22 Filtered: 0

CDC Configuration CDC Phases Waivers

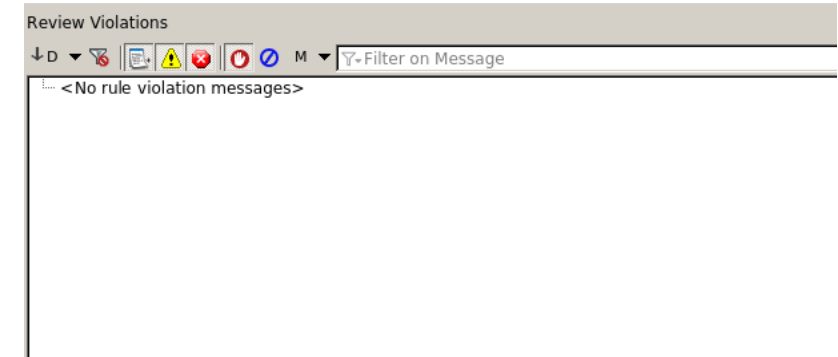
Summary

☐ There should be no error message in console.



```
session_0
WARNING (WDC241): At least one SDC clock exceeded maximum factor 10 after normalization.
The proportions of their periods cannot be kept among their factors.
WARNING (WCK028): The clock "clk1" was declared with an absolute factor of 14.
The presence of slow clocks - with absolute factor greater than 10 - can cause major performance issues during
proof.
WARNING (WCK028): The clock "clk3" was declared with an absolute factor of 20.
The presence of slow clocks - with absolute factor greater than 10 - can cause major performance issues during
proof.
[<embedded>] % check_cdc -protocol check -generate
WARNING (WRS035): The declared reset file does not initialize all flops/latches. Use the command "reset -sequence" or
reset files that contain internal signals to avoid spurious counterexamples. If your reset file already contains
internal signals use the switch "-include_internal" to reset internal flops/latches.
[<embedded>] % check_cdc -waiver -prove
WARNING (WDC021): You have not created CDC conditional waiver checks yet; nothing to prove.
Run "check_cdc -waiver -add" to manually create conditional waivers.
Run "check_cdc -waiver -generate" to automatically create conditional waivers.
```

☐ No violation message



Violation Key	Violation Type	Check	Tag	Severity	Rule Value	Design Value	Failure Reason	# Pairs	Source Reset Domain	Destination Re

☐ Nothing in violations.csv