A Report On

Assignment -1

---

by

Dinesh Joshi (JVL202215)

FOR THE COURSE

ELP736

Physical Design Laboratory

IIT Delhi

Feb 1, 2022

Table of Contents

Problem Statement

Read the Sunburst Async FIFO paper. Then do PD(Physical Design) flow. Assume any clock frequency like (10MHz or 20MHz).

Task to be done in GENUS

• create clock definition	• max capacitance
• generated clock definition	• max fanout
• Virtual clock	• clock latency
• input delay	• clock uncertainty
• output delay	• Multicycle Path
• max delay	• False Path
• min delay	• Half Cycle Path
• max transition	• disable timing arcs
	• case analysis

Sanity checks after performing above task

• Multidriven nets	• Wire loops across hierarchies
• Floating pins	• If any unconstrained paths exist in the design then PNR tool will not optimize that path, so these checks are used to report unconstrained paths
• Undriven input ports	• Checks whether the clock is reaching to all the clock pin of the flip-flop.
• Unloaded outputs	• Check if multiple clock are driving same registers
• Unconstrained pins	• Check unconstrained endpoints
• Pin mismatch counts between an instance and its reference	• Port missing input/output delay.
• Tristate buses with non-tristate drivers	• Port missing slew/load constraints.

Design Assumptions:

wclk: 100MHz
rclk: 300MHz

Note: The timebase is 1000ps as indicated in Fig1. The numbers in the subsequent figures should be considered accordingly, i.e., 3 would indicate 3000ps.

1. Genus constraints

1.1 Create clock definition

The ‘create_clock’ construct is used to define clocks in the sdc file.
For the assignment, we have assumed 100MHz wclk and 300MHz rclk.
Accordingly, the clocks are defined with 10000ps and 3333ps period as indicated in Fig 1.

```
 7 set_units -time 1000ps
 8
 9 # Current design ------------------
10 current_design fifo
11 #---------------------------------
12
13 # Clock declaration ---------------
14 create_clock -name "wclk" -period 10.0 -waveform {0.0 5.0} [get_ports wclk]
15 create_clock -name "rclk" -period 3.333 -waveform {0.0 1.6665} [get_ports rclk]
16 #---------------------------------
~/elp736/assignments/assignment1/SYNTH/sdc/fifo.sdc
```

Fig1: create clock definition

1.2 Generated clock definition

The ‘create_generated_clock’ construct is used to define generated clocks in case of internally generated clocks in the design. Such cases can be there in case of clock muxing, clock division, etc. In the async fifo design, there is no clock generation happening, hence the ‘create_generated_clock’ construct becomes irrelevant.

1.3 Virtual clock definition

The virtual clocks are defined to associate pins with a clock which is not physically present. Hence it acts as a reference only which can be used for associativity of internal signals. In the async fifo design, the pins are syncd with respect to the clock domains apart from the async assertion of reset pin, which is asynchronously

asserted and synchronously deasserted, hence there is no virtual clock definition needed. The virtual clocks are defined using create_clock construct itself.

1.4 Input delay

The ‘set_input_delay’ construct is used to declare the delays on the input ports with respect to design clocks. The Fig 2 below indicates the input delay declaration.

```
70 # I/O Delays ----------------------
71 set_input_delay -clock [get_clocks rclk] -add_delay 0.0001 [get_ports rrst_n]
72 set_input_delay -clock [get_clocks rclk] -add_delay 0.0001 [get_ports rinc]
73 set_input_delay -clock [get_clocks wclk] -add_delay 0.0001 [get_ports winc]
74 set_input_delay -clock [get_clocks wclk] -add_delay 0.0001 [get_ports {wdata[0]}]
75 set_input_delay -clock [get_clocks wclk] -add_delay 0.0001 [get_ports {wdata[1]}]
76 set_input_delay -clock [get_clocks wclk] -add_delay 0.0001 [get_ports {wdata[2]}]
77 set_input_delay -clock [get_clocks wclk] -add_delay 0.0001 [get_ports {wdata[3]}]
78 set_input_delay -clock [get_clocks wclk] -add_delay 0.0001 [get_ports {wdata[4]}]
79 set_input_delay -clock [get_clocks wclk] -add_delay 0.0001 [get_ports {wdata[5]}]
80 set_input_delay -clock [get_clocks wclk] -add_delay 0.0001 [get_ports {wdata[6]}]
81 set_input_delay -clock [get_clocks wclk] -add_delay 0.0001 [get_ports {wdata[7]}]
82 set_input_delay -clock [get_clocks wclk] -add_delay 0.0001 [get_ports wrst_n]
~/elp736/assignments/assignment1/SYNTH/sdc/fifo.sdc
```

Fig2: input delay

1.5 Output delay

The ‘set_output_delay’ construct is used to declare the delays on the output ports with respect to design clocks. The Fig 3 below indicates the output delay declaration.

```
83 set_output_delay -clock [get_clocks wclk] -add_delay 0.0001 [get_ports wfull]
84 set_output_delay -clock [get_clocks rclk] -add_delay 0.0001 [get_ports rempty]
85 set_output_delay -clock [get_clocks rclk] -add_delay 0.0001 [get_ports {rdata[0]}]
86 set_output_delay -clock [get_clocks rclk] -add_delay 0.0001 [get_ports {rdata[1]}]
87 set_output_delay -clock [get_clocks rclk] -add_delay 0.0001 [get_ports {rdata[2]}]
88 set_output_delay -clock [get_clocks rclk] -add_delay 0.0001 [get_ports {rdata[3]}]
89 set_output_delay -clock [get_clocks rclk] -add_delay 0.0001 [get_ports {rdata[4]}]
90 set_output_delay -clock [get_clocks rclk] -add_delay 0.0001 [get_ports {rdata[5]}]
91 set_output_delay -clock [get_clocks rclk] -add_delay 0.0001 [get_ports {rdata[6]}]
92 set_output_delay -clock [get_clocks rclk] -add_delay 0.0001 [get_ports {rdata[7]}]
93 #---------------------------------
~/elp736/assignments/assignment1/SYNTH/sdc/fifo.sdc
```

Fig3: output delay

1.6 Max delay

The ‘set_max_delay’ construct is used to declare the maximum allowed delays on the ports with respect to design clocks. The Fig 4 below indicates the max delay declaration.

```
95 # Max/min delays ------------------
96 set_max_delay 1 -from [get_ports {wrst_n} -to [get_clocks wclk]
97 set_min_delay 1 -from [get_ports {rrst_n} -to [get_clocks rclk]
98 #---------------------------------
~/elp736/assignments/assignment1/SYNTH/sdc/fifo.sdc
```

Fig4: max delay

1.7 Min delay

The ‘set_min_delay’ construct is used to declare the minimum allowed delays on the ports with respect to design clocks. The Fig 5 below indicates the min delay declaration.

```
95 # Max/min delays ------------------
96 set_max_delay 1 -from [get_ports {wrst_n} -to [get_clocks wclk]
97 set_min_delay 1 -from [get_ports {rrst_n} -to [get_clocks rclk]
98 #---------------------------------
~/elp736/assignments/assignment1/SYNTH/sdc/fifo.sdc
```

Fig5: min delay

1.8 Max transition

The ‘set_max_transition’ construct is used to declare the maximum allowed rise/fall transition time of design ports. The Fig 6 below indicates the max transition declaration.

```
24 # Data transition -----------------
25 set_max_transition 3.0 [get_ports {wdata[7]}]
26 set_max_transition 3.0 [get_ports {wdata[6]}]
27 set_max_transition 3.0 [get_ports {wdata[5]}]
28 set_max_transition 3.0 [get_ports {wdata[4]}]
29 set_max_transition 3.0 [get_ports {wdata[3]}]
30 set_max_transition 3.0 [get_ports {wdata[2]}]
31 set_max_transition 3.0 [get_ports {wdata[1]}]
32 set_max_transition 3.0 [get_ports {wdata[0]}]
33 set_max_transition 3.0 [get_ports winc]
34 set_max_transition 3.0 [get_ports wclk]
35 set_max_transition 2.0 [get_ports wrst_n]
36 set_max_transition 3.0 [get_ports rinc]
37 set_max_transition 3.0 [get_ports rclk]
38 set_max_transition 2.0 [get_ports rrst_n]
39 set_max_transition 3.0 [get_ports {rdata[7]}]
40 set_max_transition 3.0 [get_ports {rdata[6]}]
41 set_max_transition 3.0 [get_ports {rdata[5]}]
42 set_max_transition 3.0 [get_ports {rdata[4]}]
43 set_max_transition 3.0 [get_ports {rdata[3]}]
44 set_max_transition 3.0 [get_ports {rdata[2]}]
45 set_max_transition 3.0 [get_ports {rdata[1]}]
46 set_max_transition 3.0 [get_ports {rdata[0]}]
47 set_max_transition 3.0 [get_ports wfull]
48 set_max_transition 3.0 [get_ports rempty]
49 #---------------------------------
~/elp736/assignments/assignment1/SYNTH/sdc/fifo.sdc
```

Fig6: max transition

1.9 Max capacitance

The ‘set_max_capacitance’ construct is used to declare the maximum allowed capacitance on design pins. The Fig. 7 below declares a max_cap of 2 on all the design pins apart from wclk and rclk for which the max_cap of 0.5 and 0.45 are declared.

```
64 # Setting max cap -----------------
65 set_max_capacitance 2.0 [current_design]
66 set_max_capacitance 0.5 [get_ports wclk]
67 set_max_capacitance 0.45 [get_ports rclk]
68 #---------------------------------
~/elp736/assignments/assignment1/SYNTH/sdc/fifo.sdc
```

Fig7: max capacitance

1.10 Max fanout

The ‘max_fanout’ construct is used to limit the fanout to a specified value. The figure below indicates the max_fanout definitions for different design ports. Also min_fanout is also indicated in below snippet which indicates minimum allowed fanout. The max_fanout is used since there is cell drive strength limitations.

```
100 # Max/min fanouts -----------------
101 set_max_fanout 4.000 [get_ports winc]
102 set_max_fanout 2.000 [get_ports wrst_n]
103 set_max_fanout 4.000 [get_ports rinc]
104 set_max_fanout 2.000 [get_ports rrst_n]
105 set_min_fanout 4.000 [get_ports {wdata[7]}]
106 set_min_fanout 4.000 [get_ports {wdata[6]}]
107 set_min_fanout 4.000 [get_ports {wdata[5]}]
108 set_min_fanout 4.000 [get_ports {wdata[4]}]
109 set_min_fanout 4.000 [get_ports {wdata[3]}]
110 set_min_fanout 4.000 [get_ports {wdata[2]}]
111 set_min_fanout 4.000 [get_ports {wdata[1]}]
112 set_min_fanout 4.000 [get_ports {wdata[0]}]
113 set_min_fanout 4.000 [get_ports winc]
114 set_min_fanout 2.000 [get_ports wclk]
115 set_min_fanout 4.000 [get_ports wrst_n]
116 set_min_fanout 4.000 [get_ports rinc]
117 set_min_fanout 4.000 [get_ports rclk]
118 set_min_fanout 4.000 [get_ports rrst_n]
119 #---------------------------------
~/elp736/assignments/assignment1/SYNTH/sdc/fifo.sdc
```

Fig8: max fanout

1.11 Clock latency

The ‘set_clock_latency’ construct is used to declare the clock path latency in the design. The latency can be either source latency or the network latency. This is

specified with the -source switch as indicated in Fig. 9 wherein the latency in the wclk is specified as source latency whereas the rclk latency is specified as network latency.

```
51 # Clock latency -------------------
52 # The -source switch is providing the source latency whereas without
    -source switch network latency is being specified
53 set_clock_latency -source 0.1 [get_clocks wclk]
54 set_clock_latency  0.1 [get_clocks rclk]
55 #---------------------------------
~/elp736/assignments/assignment1/SYNTH/sdc/fifo.sdc   55,35          23%
```

Fig9: clock latency

1.12 Clock uncertainty

The ‘set_clock_uncertainty’ construct is used to declare the clock path uncertainty. The clock path uncertainty can be specified for both setup as well as hold as well as for data path crossing independently. The Fig. 10 indicates the set_clock_uncertainty declaration for the async fifo design under consideration.

```
57 # Clock uncertainty --------------
58 # The clock uncertainty can be specified for both setup as well as hold as well as
    a data path crossing.
59 # The uncertainty can also be specified separately for rise and fall transitions.
60 set_clock_uncertainty -setup 0.2 [get_clocks wclk]
61 set_clock_uncertainty -from [get_clocks wclk] -to [get_clocks rclk]  -hold 0.3
62 #---------------------------------
~/elp736/assignments/assignment1/SYNTH/sdc/fifo.sdc                38,41          29%
```

Fig10: clock uncertainty

1.13 Multicycle path

The async fifo design consists of two async clock domains and there are just two clocks, hence there is no scope of MCPs. However the paper specifies that the resets are asynchronously asserted and synchronously deasserted. Hence we can put reset assertion as false using setup as false path and reset deassertion can be assumed to happen over atleast 2 clock cycles, which is a fair assumption since reset will be sufficiently long enough. Hence an MCP on hold only for async reset deassertion can be applied as indicated in Fig 11.

```
130 # Multicycle path ----------------
131 # Hold of reset pins is treated as MCP assuming reset is long enough as
    compared to clk, which is a fair assumption.
132 set_multicycle_path -from [get_ports wrst_n] -hold -start 1
133 set_multicycle_path -from [get_ports rrst_n] -hold -start 1
134 #---------------------------------
~/elp736/assignments/assignment1/SYNTH/sdc/fifo.sdc       132,59          86%
```

Fig11: multicycle path

1.14 False path

The ‘set_false_path’ construct is used to indicate false/invalid timing paths. Since the async fifo consists of two asynchronous domains, hence paths in between wclk and rclk can be put as false paths.
Also the reset strategy is specified as async assert and sync deassert, hence setup path of reset can also be treated as false path. The Fig. 12 indicates the false path declaration examples.

```
121 # False paths ---------------------
122 # Putting the async crossings as false paths
123 set_false_path -from [get_clocks rclk] -to [get_clocks wclk]
124 set_false_path -from [get_clocks wclk] -to [get_clocks rclk]
125 # The reset is async assert sync deassert. Marking setup as false.
126 set_false_path -from [get_ports wrst_n] -setup
127 set_false_path -from [get_ports rrst_n] -setup
128 #---------------------------------
~/elp736/assignments/assignment1/SYNTH/sdc/fifo.sdc
```

Fig12: false path

1.15 Half-cycle path

The half-cycle paths are used to indicate the conditions wherein the timing has to be met in half-cycles. In the async fifo design under consideration, all the flops are running on posedge of clocks, hence either the paths are full cycle timing paths or asynchronous/false paths. Hence half-cycle paths are irrelevant for the async fifo design under consideration.

1.16 Disable timing arcs

On evaluation of the netlist, we see that the scannable flops are present in the techlib which are used. Now, since we are considering only the functional mode timing, hence the scan data path timings can be safely considered as invalid. The same can be done using set_disable_timing construct. The Fig. 13 indicates an example wherein the paths from/to the SD (scan data) pins of fifomem registers is considered as invalid by disabling the timing arcs therein.

```
136 # Disable timing arc -------------
137 # Disabling timing arcs from and to the scan data pins of fifomem/mem_reg
     flops. This is fine since we are considering functional mode timing cons
    traints.
138 set_disable_timing -from [get_pins {fifomem/mem_reg\\[*\\]\\[*\\]/SD}]
139 set_disable_timing -to   [get_pins {fifomem/mem_reg\\[*\\]\\[*\\]/SD}]
140 #---------------------------------
~/elp736/assignments/assignment1/SYNTH/sdc/fifo.sdc       140,1          91%
```

Fig13: disable timing arcs

1.17 Case analysis

The ‘set_case_analysis’ construct is used to indicate constant values. Since in the async fifo design, there is no constant value, hence there is no scope of using ‘set_case_analysis’, however we see that the scannable flops have SE pins, which can be safely constrained to inactive (0) value. Hence to indicate an example of set_case_analysis, the SE (scan enable) pins of fifomem registers is constrained as 0 as indicated in Fig. 14.

```
142 # Set case analysis ---------------
143 # Since we are considering hte functional mode, we can put set case
    analysis on scan enable pins. Putting it on fifomem registers as an
    example.
144 set_case_analysis 0 [get_pins {fifomem/mem_reg\\[*\\]\\[*\\]/SE}]
145 #---------------------------------
~/elp736/assignments/assignment1/SYNTH/sdc/fifo.sdc   145,35          95%
```

Fig14: case analysis

2. Sanity reports

The genus synthesis has been for the ‘async fifo’ design with the above indicated constraints with UMC65 library.

The timing lint, check design, qor, area, power reports and the netlist are extracted. The sanity checks reports are indicated as shown in subsequent sections:

2.1 Check-design report

The checkdesign report is extracted using below command in genus:
check_design > fifo_synthesis_report_checkdesign.rep

```
 1
 2          Check Design Report (c)
 3          -----------------------
 4
 5  Summary
 6  -------
 7
 8                Name                              Total
 9 ------------------------------------------------------
10 Unresolved References                               0
11 Empty Modules                                       0
12 Unloaded Port(s)                                    0
13 Unloaded Sequential Pin(s)                          0
14 Unloaded Combinational Pin(s)                       0
15 Assigns                                             0
16 Undriven Port(s)                                    0
17 Undriven Leaf Pin(s)                                0
18 Undriven hierarchical pin(s)                        0
19 Multidriven Port(s)                                 0
20 Multidriven Leaf Pin(s)                             0
21 Multidriven hierarchical Pin(s)                     0
22 Multidriven unloaded net(s)                         0
23 Constant Port(s)                                    0
24 Constant Leaf Pin(s)                                0
25 Constant hierarchical Pin(s)                        0
26 Preserved leaf instance(s)                          0
27 Preserved hierarchical instance(s)                  0
28 Feedthrough Modules(s)                              0
29 Libcells with no LEF cell                           0
30 Physical (LEF) cells with no libcell                0
31 Subdesigns with long module name                    0
32 Physical only instance(s)                           0
33 Logical only instance(s)                          340
34
35   Done Checking the design.
~
~/elp736/assignments/assignment1/SYNTH/logs/fifo_synthesis_report_checkdesign.rep
```

Fig15: check design report

2.2 Timing lint report

The timing lint report is extracted using below command in genus:
report timing -lint > fifo_synthesis_report_timinglint.rep

```
 52
 53 Lint summary
 54  Unconnected/logic driven clocks                          0
 55  Sequential data pins driven by a clock signal            0
 56  Sequential clock pins without clock waveform             0
 57  Sequential clock pins with multiple clock waveforms      0
 58  Generated clocks without clock waveform                  0
 59  Generated clocks with incompatible options               0
 60  Generated clocks with multi-master clock                 0
 61  Paths constrained with different clocks                  0
 62  Loop-breaking cells for combinational feedback           0
 63  Nets with multiple drivers                               0
 64  Timing exceptions with no effect                         0
 65  Suspicious multi_cycle exceptions                        0
 66  Pins/ports with conflicting case constants               0
 67  Inputs without clocked external delays                   0
 68  Outputs without clocked external delays                  0
 69  Inputs without external driver/transition               12
 70  Outputs without external load                           10
 71  Exceptions with invalid timing start-/endpoints          0
 72
 73                                                Total:    22
 74
~/elp736/assignments/assignment1/SYNTH/logs/fifo_synthesis_report_timinglint.rep
```

Fig16: Timing Lint report

2.3 QOR report

The QOR report is extracted using below command in genus:
report qor > fifo_synthesis_report_qor.rep

```
1  ============================================================
2    Generated by:           Genus(TM) Synthesis Solution 19.12-s121_1
3    Generated on:           Feb 01 2022  10:12:28 pm
4    Module:                 fifo
5    Technology library:     uk65lscllmvbbr_100c25_tc
6    Operating conditions:   uk65lscllmvbbr_100c25_tc (balanced_tree)
7    Wireload mode:          top
8    Area mode:              timing library
9  ============================================================
10
11 Timing
12 --------
13
14 Clock  Period
15 --------------
16 rclk   3333.0
17 wclk  10000.0
18
19
20   Cost    Critical         Violating
21  Group   Path Slack  TNS     Paths
22 -------------------------------------
23 default    No paths   0.0
24 rclk         2153.8   0.0          0
25 wclk         8629.7   0.0          0
26 -------------------------------------
27 Total                 0.0          0
28
29 Instance Count
30 --------------
31 Leaf Instance Count             340
32 Physical Instance count           0
33 Sequential Instance Count       168
34 Combinational Instance Count    172
35 Hierarchical Instance Count       5
36
37 Area
38 ----
39 Cell Area                          1897.200
40 Physical Cell Area                 0.000
41 Total Cell Area (Cell+Physical)    1897.200
42 Net Area                           0.000
43 Total Area (Cell+Physical+Net)     1897.200
44
45 Max Fanout                         148 (wclk)
46 Min Fanout                         0 (rrst_n)
47 Average Fanout                     3.2
48 Terms to net ratio                 4.2881
49 Terms to instance ratio            4.4647
50 Runtime                            66.50061199999999 seconds
51 Elapsed Runtime                    63 seconds
52 Genus peak memory usage            1542.78
53 Innovus peak memory usage          no_value
54 Hostname                           dirac1.vlsi.ee.iitd.ac.in
~
~/elp736/assignments/assignment1/SYNTH/logs/fifo_synthesis_report_qor.rep
```

Fig17: QOR report

3. Conclusion

With the help of timing lint report, we can infer that all the constraints are valid, there is no non-matching constraint.
The 22 reported points are due to the fact that we are doing analysis at block level wherein the top level IOs are not having any driver/loads connected. These are expected.

The check design summary is also clean, there is no anomaly in the design.

The QOR report indicates that the timing is met, there is no violating path.
