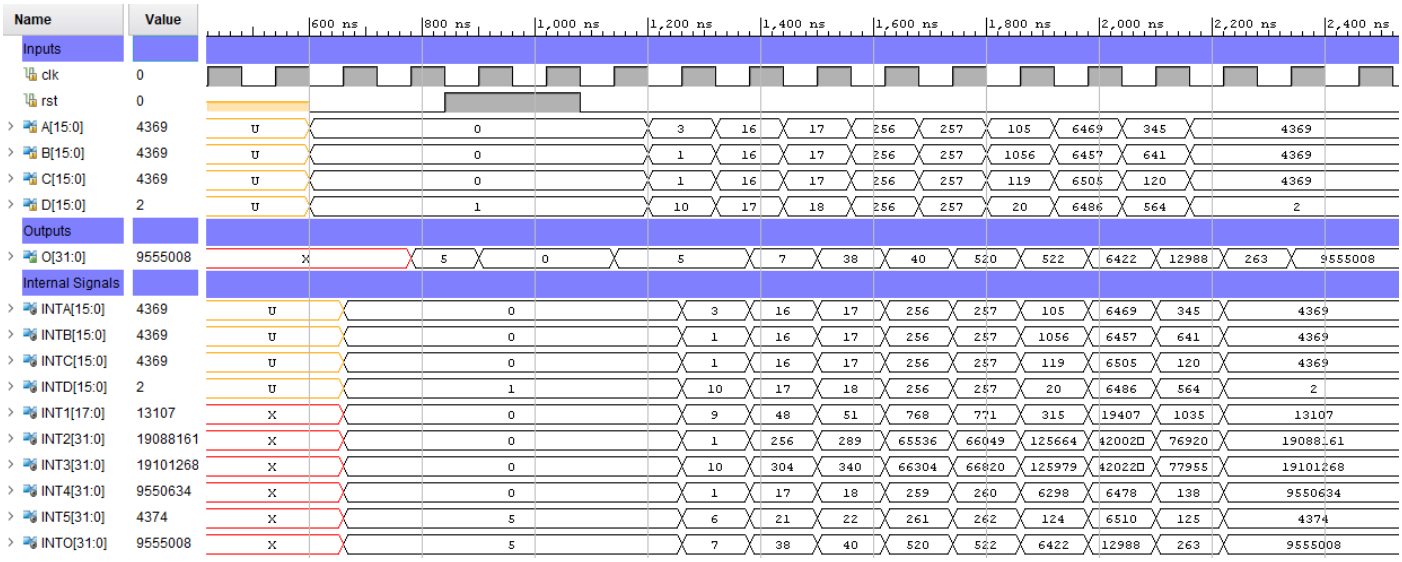
Lab 3 - Task A:

With No Pipelines

2.1.1: Screenshot of the simulation window with all inputs, internal data busses, and the output in unsigned decimal format used in the same testbench in lab 1. Include the console output (as a separate screenshot).

Screenshot of the simulation window with all inputs, internal data busses, and the output in unsigned decimal format used in the same testbench in lab 1



Console output

```
launch_simulation
INFO: [Vivado 12-5682] Launching behavioral simulation in 'D:/files/uni/digitalengineering/lab3/2-1-1_2-1-3/Performance_Check_On_Algorithm_Circuit/Performance_Check_On_Algorithm_Circuit.sim/sim_1/behav/xsim'
INFO: [SIM-utils-51] Simulation object is 'sim_1'
INFO: [SIM-utils-54] Inspecting design source files for 'two_process_TB' in fileset 'sim_1'...
INFO: [USF-XSim-97] Finding global include files...
INFO: [USF-XSim-98] Fetching design files from 'sim_1'...
INFO: [USF-XSim-2] XSim::Compile design
INFO: [USF-XSim-61] Executing 'COMPILE and ANALYZE' step in 'D:/files/uni/digitalengineering/lab3/2-1-1_2-1-3/Performance_Check_On_Algorithm_Circuit/Performance_Check_On_Algorithm_Circuit.sim/sim_1/behav/xsim'
"xvhdl --incr --relax -prj two_process_TB_vhdl.prj"
INFO: [USF-XSim-69] 'compile' step finished in '2' seconds
INFO: [USF-XSim-3] XSim::Elaborate design
INFO: [USF-XSim-61] Executing 'ELABORATE' step in 'D:/files/uni/digitalengineering/lab3/2-1-1_2-1-3/Performance_Check_On_Algorithm_Circuit/Performance_Check_On_Algorithm_Circuit.sim/sim_1/behav/xsim'
Vivado Simulator 2017.4
Copyright 1986-1999, 2001-2016 Xilinx, Inc. All Rights Reserved.
Running: D:/programs/uni/xilinx/Vivado/Vivado/2017.4/bin/unwrapped/win64.o/xelab.exe -wto 14a433ed9f5e47a497118842e777e334 --incr --debug typical --relax --mt 2 -L xil_defaultlib -L secureip --snapshot two_process_TB_behav xil_
Using 2 slave threads.
Starting static elaboration
Completed static elaboration
INFO: [XSIM 43-4323] No Change in HDL. Linking previously generated obj files to create kernel
INFO: [USF-XSim-69] 'elaborate' step finished in '2' seconds
INFO: [USF-XSim-4] XSim::Simulate design
INFO: [USF-XSim-61] Executing 'SIMULATE' step in 'D:/files/uni/digitalengineering/lab3/2-1-1_2-1-3/Performance_Check_On_Algorithm_Circuit/Performance_Check_On_Algorithm_Circuit.sim/sim_1/behav/xsim'
INFO: [USF-XSim-98] *** Running xsim
   with args "two_process_TB_behav -key {Behavioral:sim_1:Functional:two_process_TB} -tclbatch {two_process_TB.tcl} -view {D:/files/uni/digitalengineering/lab3/2-1-1_2-1-3/Performance_Check_On_Algorithm_Circuit/two_process_TB_b
INFO: [USF-XSim-8] Loading simulator feature
Vivado Simulator 2017.4
Time resolution is 1 ps
open_wave_config D:/files/uni/digitalengineering/lab3/2-1-1_2-1-3/Performance_Check_On_Algorithm_Circuit/two_process_TB_behav.wcfg
source two_process_TB.tcl
# set curr_wave [current_wave_config]
# if { [string length $curr_wave] == 0 } {
#   if { [llength [get_objects]] > 0} {
#     add_wave /
#     set_property needs_save false [current_wave_config]
#   } else {
#      send_msg_id Add_Wave-1 WARNING "No top level signals found. Simulator will start without a wave window. If you want to open a wave window go to 'File->New Waveform Configuration' or type 'create_wave_config' in the TCL c
#   }
# }
# run 1000ns
INFO: [USF-XSim-96] XSim completed. Design snapshot 'two_process_TB_behav' loaded.
INFO: [USF-XSim-97] XSim simulation ran for 1000ns
launch_simulation: Time (s): cpu = 00:00:02 ; elapsed = 00:00:06 . Memory (MB): peak = 933.516 ; gain = 0.000
run 100 us
```

2.1.2: Best period where the constraints are met; screenshot of the “Design Runs” tab with all columns up to “DSP”; what is the highest frequency at which our design should be able to run according to the WNS results?

Best period where the constraints are met

106ns

Screenshot of the “Design Runs” tab with all columns up to “DSP”

Name	Constraints	Status	WNS	TNS	WHS	THS	TPWS	Total Power	Failed Routes	LUT	FF	BRAMs	URAM	DSP
✓ synth_1	constrs_1	synth_design Complete!								897	96	0.00	0	0
✓ impl_1	constrs_1	Running route_design...	0.318	0.000	0.476	0.000	0.000	0.109	0	887	96	0.00	0	0

The highest frequency at which our design should be able to run according to the WNS results

The highest frequency is 9.48 MHz ($\frac{1}{106ns-0.537ns}$). If we are rounding to the nearest nanosecond the highest frequency is 9.43 MHz ($\frac{1}{106ns}$).

2.1.3: Screenshot of the first few lines of the first Max Delay Path in the Post-Route Timing Report.

```
Max Delay Paths
--------------------------------------------------------------------------------------
Slack (MET) :             0.318ns  (required time - arrival time)
  Source:                 INTB_reg[4]/C
                            (rising edge-triggered cell FDRE clocked by clk  {rise@0.000ns fall@53.000ns period=106.000ns})
  Destination:            O_reg[29]/D
                            (rising edge-triggered cell FDRE clocked by clk  {rise@0.000ns fall@53.000ns period=106.000ns})
  Path Group:             clk
  Path Type:              Setup (Max at Slow Process Corner)
  Requirement:            106.000ns  (clk rise@106.000ns - clk rise@0.000ns)
  Data Path Delay:        105.662ns  (logic 49.382ns (46.736%)  route 56.280ns (53.264%))
  Logic Levels:           211  (CARRY4=174 LUT2=4 LUT3=29 LUT4=1 LUT6=3)
```

With 1 pipeline:

A note on variable names

For the first setup we are using the following 'stages' of the process for variable names:

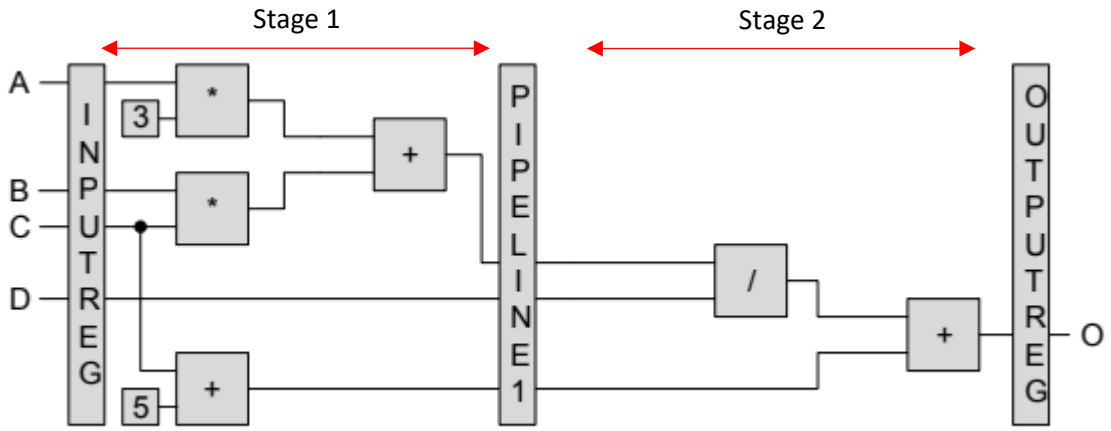
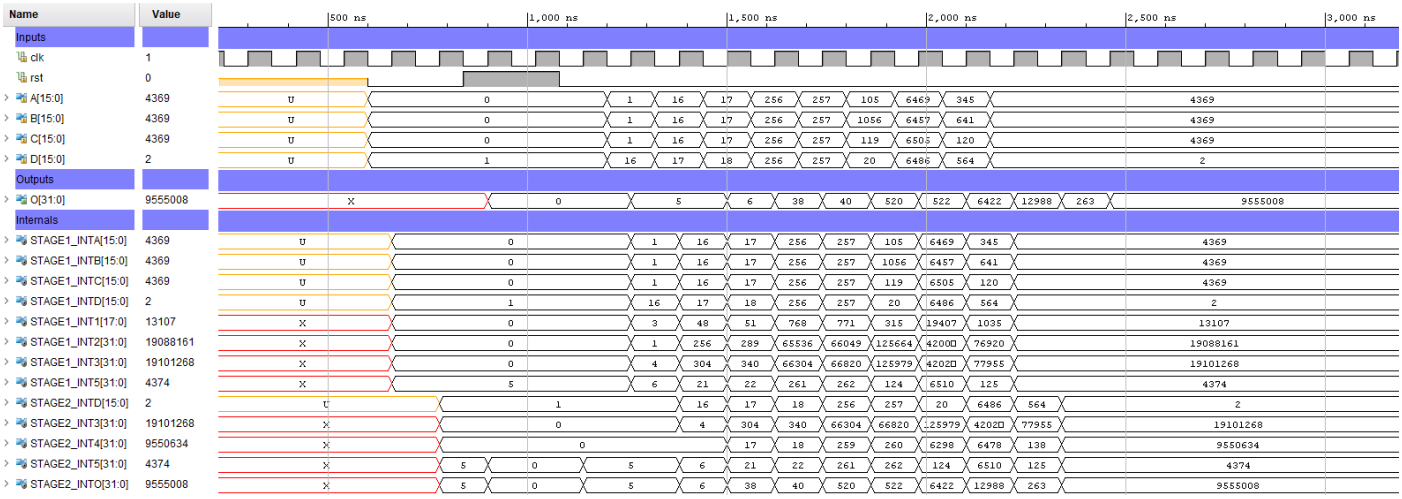


Figure 1.2

2.1.4: Screenshot of the simulation window with all inputs, internal data busses, and the output in unsigned decimal format for the same sets of input (and output) values used in the preceding simulations. Include the console output (as a separate screenshot).

Screenshot of the simulation window with all inputs, internal data busses, and the output in unsigned decimal format for the same sets of inputs and outputs used in the preceding simulations



Screenshot of the console output

```
launch_simulation
INFO: [Vivado 12-5682] Launching behavioral simulation in 'D:/files/uni/digitalengineering/lab3/2-1-4_2-1-6/Performance_Check_On_Algorithm_Circuit/Performance_Check_On_Algorithm_Circuit.sim/sim_1/behav/xsim'
INFO: [SIM-utils-51] Simulation object is 'sim_1'
INFO: [SIM-utils-54] Inspecting design source files for 'two_process_TB' in fileset 'sim_1'...
INFO: [USF-XSim-97] Finding global include files...
INFO: [USF-XSim-98] Fetching design files from 'sim_1'...
INFO: [USF-XSim-2] XSim::Compile design
INFO: [USF-XSim-61] Executing 'COMPILE and ANALYZE' step in 'D:/files/uni/digitalengineering/lab3/2-1-4_2-1-6/Performance_Check_On_Algorithm_Circuit/Performance_Check_On_Algorithm_Circuit.sim/sim_1/behav/xsim'
"xvhdl --incr --relax -prj two_process_TB_vhdl.prj"
INFO: [USF-XSim-69] 'compile' step finished in '2' seconds
INFO: [USF-XSim-3] XSim::Elaborate design
INFO: [USF-XSim-61] Executing 'ELABORATE' step in 'D:/files/uni/digitalengineering/lab3/2-1-4_2-1-6/Performance_Check_On_Algorithm_Circuit/Performance_Check_On_Algorithm_Circuit.sim/sim_1/behav/xsim'
Vivado Simulator 2017.4
Copyright 1986-1999, 2001-2016 Xilinx, Inc. All Rights Reserved.
Running: D:/programs/uni/xilinx/Vivado/Vivado/2017.4/bin/unwrapped/win64.o/xelab.exe -wto 14a433ed9f5e47a497118842e777e334 --incr --debug typical --relax --mt 2 -L xil_defaultlib -L secureip --snapshot two_process_TB_behav xil
Using 2 slave threads.
Starting static elaboration
Completed static elaboration
INFO: [XSIM 43-4323] No Change in HDL. Linking previously generated obj files to create kernel
INFO: [USF-XSim-69] 'elaborate' step finished in '2' seconds
INFO: [USF-XSim-4] XSim::Simulate design
INFO: [USF-XSim-61] Executing 'SIMULATE' step in 'D:/files/uni/digitalengineering/lab3/2-1-4_2-1-6/Performance_Check_On_Algorithm_Circuit/Performance_Check_On_Algorithm_Circuit.sim/sim_1/behav/xsim'
INFO: [USF-XSim-98] *** Running xsim
   with args "two_process_TB_behav -key {Behavioral:sim_1:Functional:two_process_TB} -tclbatch {two_process_TB.tcl} -view {D:/files/uni/digitalengineering/lab3/2-1-4_2-1-6/Performance_Check_On_Algorithm_Circuit/two_process_TB_
INFO: [USF-XSim-8] Loading simulator feature
Vivado Simulator 2017.4
Time resolution is 1 ps
open_wave_config D:/files/uni/digitalengineering/lab3/2-1-4_2-1-6/Performance_Check_On_Algorithm_Circuit/two_process_TB_behav.wcfg
source two_process_TB.tcl
# set curr_wave [current_wave_config]
# if { [string length $curr_wave] == 0 } {
#   if { [llength [get_objects]] > 0} {
#     add_wave /
#     set_property needs_save false [current_wave_config]
#   } else {
#      send_msg_id Add_Wave-1 WARNING "No top level signals found. Simulator will start without a wave window. If you want to open a wave window go to 'File->New Waveform Configuration' or type 'create_wave_config' in the TCL
#   }
# }
# run 1000ns
INFO: [USF-XSim-96] XSim completed. Design snapshot 'two_process_TB_behav' loaded.
INFO: [USF-XSim-97] XSim simulation ran for 1000ns
launch_simulation: Time (s): cpu = 00:00:03 ; elapsed = 00:00:07 . Memory (MB): peak = 888.961 ; gain = 7.645
run 100 us
```

2.1.5: Write the best period where the constraints are met (i.e. the one just before it starts to fail) and screenshot of the “Design Runs” tab with all columns up to “DSP”. What is the highest frequency at which your design should be able to run according to the WNS results?

The best period where the constraints are met

94ns

Screenshot of the “Design Runs” tab with all columns up to “DSP”.

Name	Constraints	Status	WNS	TNS	WHS	THS	TPWS	Total Power	Failed Routes	LUT	FF	BRAMs	URAM	DSP
✔ synth_1	constrs_1	synth_design Complete!								897	161	0.00	0	0
✔ impl_1	constrs_1	route_design Complete!	0.059	0.000	0.191	0.000	0.000	0.110	0	887	161	0.00	0	0

Highest frequency at which your design should be able to run according to the WNS results?

The highest frequency is 10.64 MHz ($\frac{1}{94ns-0.059ns}$). If we are rounding to the nearest nanosecond the highest frequency is 10.64 MHz ($\frac{1}{94ns}$).

2.1.6: Screenshot of the first few lines of the first Max Delay Path in the Post-Route Timing Report. Can you identify where the new critical path lies, with respect to the pipeline above (i.e. in which pipeline stage) and which of the mathematical operation(s) in the algorithm are in the critical path? Based on the lecture material, what are you expecting to happen to the critical path, with respect to the circuit without this pipeline stage, and why? Does the practice match the theory (compare the new clock period to the one you obtained in step 2.1.2)? Provide answers to each of these questions.

Screenshot of the first few lines of the first Max Delay Path in the Post-Route Timing Report

```
Max Delay Paths
--------------------------------------------------------------------------------------
Slack (MET) :             0.059ns  (required time - arrival time)
  Source:                 STAGE2_INTD_reg[5]/C
                            (rising edge-triggered cell FDRE clocked by clk  {rise@0.000ns fall@47.000ns period=94.000ns})
  Destination:            O_reg[29]/D
                            (rising edge-triggered cell FDRE clocked by clk  {rise@0.000ns fall@47.000ns period=94.000ns})
  Path Group:             clk
  Path Type:              Setup (Max at Slow Process Corner)
  Requirement:            94.000ns  (clk rise@94.000ns - clk rise@0.000ns)
  Data Path Delay:        93.776ns  (logic 43.145ns (46.008%)  route 50.631ns (53.992%))
  Logic Levels:           186  (CARRY4=155 LUT1=1 LUT2=1 LUT3=29)
```

Where the new critical path lies, with respect to the pipeline above?

The critical path is in the second stage. It goes between the copy of int B produced by the pipeline and the output.

Which of the mathematical operation(s) in the algorithm are in the critical path?

The critical path goes through the division and addition operators.

What was being expected to happen to the critical path, with respect to the circuit without this pipeline stage, and why?

The clock period should reduce because the signal only needs to cross between the input and output in two clock cycles, rather than one. In other words in a clock cycle the signal only needs to go across one stage so the circuit should be faster.

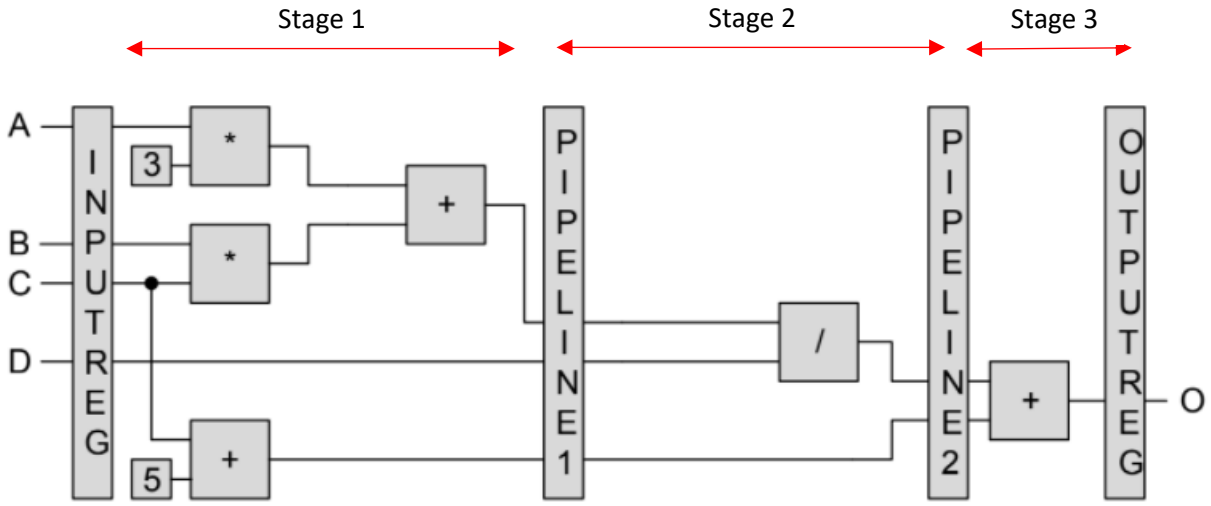
Does the practice match the theory (compare the new clock period to the one you obtained in step 2.1.2)?

Yes, the new clock period (94ns) is substantially smaller than the old one (106ns).

With 2 pipelines

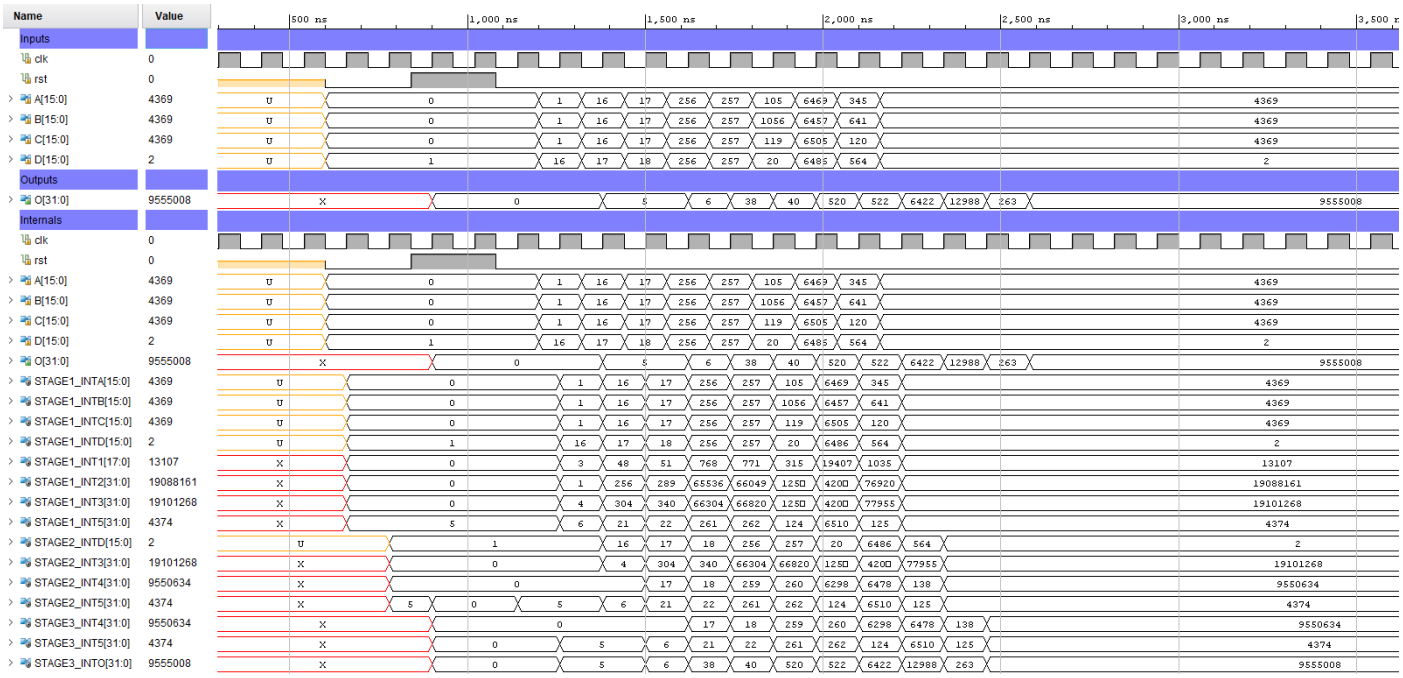
A note on variable names

For this setup we are using the following 'stages' of the process for variable names:



2.1.7: With 2 pipelines, screenshot of the simulation window with all inputs, internal data busses, and the output in unsigned decimal format for the same sets of input (and output) values used in the preceding simulations. Include the console output (as a separate screenshot).

Screenshot of the simulation window with all inputs, internal data busses, and the output in unsigned decimal format



Screenshot of the console output

```
launch_simulation
INFO: [Vivado 12-5682] Launching behavioral simulation in 'D:/files/uni/digitalengineering/lab3/2-1-7_2-1-9/Performance_Check_On_Algorithm_Circuit/Performance_Check_On_Algorithm_Circuit.sim/sim_1/behav/xsim'
INFO: [SIM-utils-51] Simulation object is 'sim_1'
INFO: [SIM-utils-54] Inspecting design source files for 'two_process_TB' in fileset 'sim_1'...
INFO: [USF-XSim-97] Finding global include files...
INFO: [USF-XSim-98] Fetching design files from 'sim_1'...
INFO: [USF-XSim-2] XSim::Compile design
INFO: [USF-XSim-61] Executing 'COMPILE and ANALYZE' step in 'D:/files/uni/digitalengineering/lab3/2-1-7_2-1-9/Performance_Check_On_Algorithm_Circuit/Performance_Check_On_Algorithm_Circuit.sim/sim_1/behav/xsim'
"xvhdl --incr --relax -prj two_process_TB_vhdl.prj"
INFO: [USF-XSim-69] 'compile' step finished in '2' seconds
INFO: [USF-XSim-3] XSim::Elaborate design
INFO: [USF-XSim-61] Executing 'ELABORATE' step in 'D:/files/uni/digitalengineering/lab3/2-1-7_2-1-9/Performance_Check_On_Algorithm_Circuit/Performance_Check_On_Algorithm_Circuit.sim/sim_1/behav/xsim'
Vivado Simulator 2017.4
Copyright 1986-1999, 2001-2016 Xilinx, Inc. All Rights Reserved.
Running: D:/programs/uni/xilinx/Vivado/Vivado/2017.4/bin/unwrapped/win64.o/xelab.exe -wto 14a433ed9f5e47a497118842e777e334 --incr --debug typical --relax --mt 2 -L xil_defaultlib -L secureip --snapshot two_process_TB_behav
Using 2 slave threads.
Starting static elaboration
Completed static elaboration
INFO: [XSIM 43-4323] No Change in HDL. Linking previously generated obj files to create kernel
INFO: [USF-XSim-69] 'elaborate' step finished in '3' seconds
INFO: [USF-XSim-4] XSim::Simulate design
INFO: [USF-XSim-61] Executing 'SIMULATE' step in 'D:/files/uni/digitalengineering/lab3/2-1-7_2-1-9/Performance_Check_On_Algorithm_Circuit/Performance_Check_On_Algorithm_Circuit.sim/sim_1/behav/xsim'
INFO: [USF-XSim-98] *** Running xsim
   with args "two_process_TB_behav -key {Behavioral:sim_1:Functional:two_process_TB} -tclbatch {two_process_TB.tcl} -view {D:/files/uni/digitalengineering/lab3/2-1-7_2-1-9/Performance_Check_On_Algorithm_Circuit/two_process_
INFO: [USF-XSim-8] Loading simulator feature
Vivado Simulator 2017.4
Time resolution is 1 ps
open_wave_config D:/files/uni/digitalengineering/lab3/2-1-7_2-1-9/Performance_Check_On_Algorithm_Circuit/two_process_TB_behav.wcfg
source two_process_TB.tcl
# set curr_wave [current_wave_config]
# if { [string length $curr_wave] == 0 } {
#   if { [llength [get_objects]] > 0} {
#     add_wave /
#     set_property needs_save false [current_wave_config]
#   } else {
#      send_msg_id Add_Wave-1 WARNING "No top level signals found. Simulator will start without a wave window. If you want to open a wave window go to 'File->New Waveform Configuration' or type 'create_wave_config' in the T
#   }
# }
# run 1000ns
INFO: [USF-XSim-96] XSim completed. Design snapshot 'two_process_TB_behav' loaded.
INFO: [USF-XSim-97] XSim simulation ran for 1000ns
launch_simulation: Time (s): cpu = 00:00:02 ; elapsed = 00:00:06 . Memory (MB): peak = 892.699 ; gain = 0.000
run 100 us
```

2.1.8: Write the best period where the constraints are met (i.e. the one just before it starts to fail) and print out a screenshot of the "Design Runs" tab, showing all columns up to "DSP". What is the highest frequency at which your design should be able to run according to the WNS results?

The best period where the constraints are met

90ns

Screenshot of the "Design Runs" tab with all columns up to "DSP".

Name	Constraints	Status	WNS	TNS	WHS	THS	TPWS	Total Power	Failed Routes	LUT	FF	BRAMs	URAM	DSP
✓ synth_1	constrs_1	synth_design Complete!								897	210	0.00	0	0
✓ impl_1	constrs_1	route_design Complete!	0.447	0.000	0.079	0.000	0.000	0.110	0	887	210	0.00	0	0

Highest frequency at which your design should be able to run according to the WNS results?

The highest frequency is 11.16 MHz ($\frac{1}{90ns-0.447ns}$). If we are rounding to the nearest nanosecond the highest frequency is 11.11 MHz ($\frac{1}{90ns}$).

2.1.9: Print out (use a screenshot) the first few lines of the first Max Delay Path in the Post-Route Timing Report (see previous script for details). Can you identify where the new critical path lies, with respect to the pipeline above (i.e. in which pipeline stage) and which of the mathematical operation(s) in the algorithm are in the critical path? Based on the lecture material, what are you expecting to happen to the critical path, with respect to the circuit without this pipeline stage, and why? Does the practice match the theory (compare the new clock period to the one you obtained in step 2.1.4)? Provide answers to each of these questions.

Screenshot of the first few lines of the first Max Delay Path in the Post-Route Timing Report

```
Max Delay Paths
--------------------------------------------------------------------------------------
Slack (MET) :             0.447ns  (required time - arrival time)
  Source:                 STAGE2_INTD_reg[2]/C
                            (rising edge-triggered cell FDRE clocked by clk  {rise@0.000ns fall@45.000ns period=90.000ns})
  Destination:            STAGE3_INT4_reg[0]/D
                            (rising edge-triggered cell FDRE clocked by clk  {rise@0.000ns fall@45.000ns period=90.000ns})
  Path Group:             clk
  Path Type:              Setup (Max at Slow Process Corner)
  Requirement:            90.000ns  (clk rise@90.000ns - clk rise@0.000ns)
  Data Path Delay:        89.478ns  (logic 42.152ns (47.109%)  route 47.326ns (52.891%))
  Logic Levels:           181  (CARRY4=150 LUT1=1 LUT3=30)
```

Where the new critical path lies, with respect to the pipeline above?

The critical path is in the second stage. It goes between the copy of int D in the pipeline 1 register and the copy of int 4 in the pipeline 2 register. It is essentially the previous critical path without the addition operator at the end.

Which of the mathematical operation(s) in the algorithm are in the critical path?

The path goes through the division operator.

What was being expected to happen to the critical path, with respect to the circuit without this pipeline stage, and why?

The clock period should reduce since the new pipeline we added was between the two operators in the previous critical path, meaning that that path can now be taken in two clock cycles rather than 1.

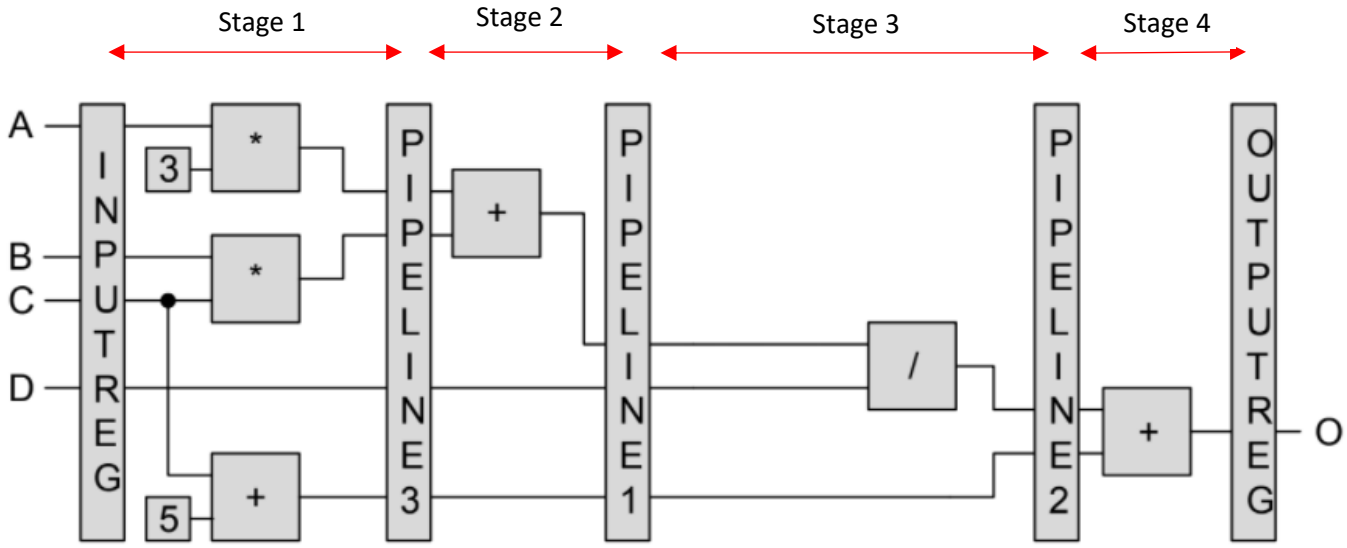
Does the practice match the theory (compare the new clock period to the one you obtained in step 2.1.4)?

Yes. There is still a reduction in clock period.

With 3 pipelines

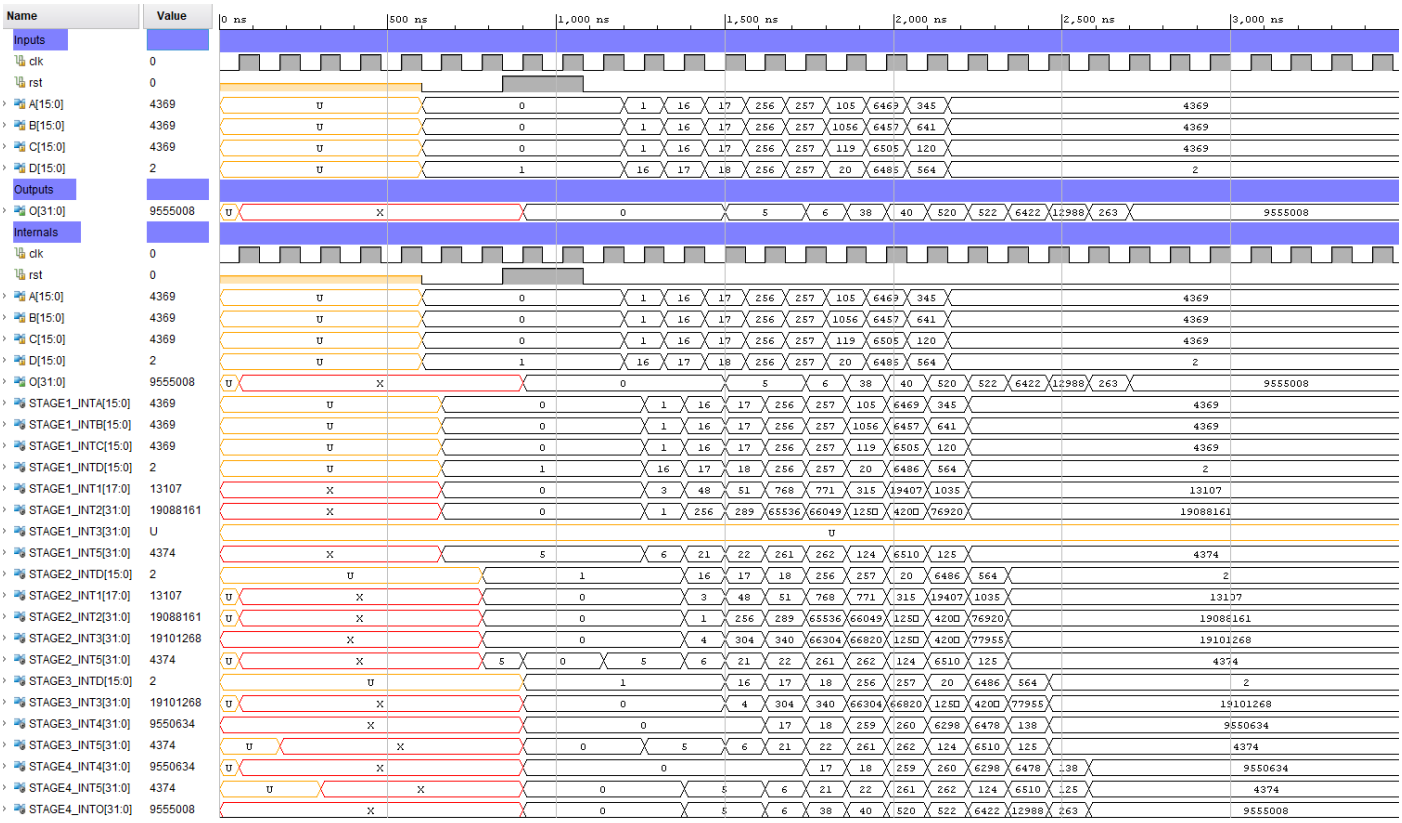
A note on variable names

For this setup we are using the following 'stages' of the process for variable names:



2.1.10: Screenshot of the simulation window with all inputs, internal data busses, and the output in unsigned decimal format for the same sets of input (and output) values used in the preceding simulations. Include the console output (as a separate screenshot).

Screenshot of the simulation window with all inputs, internal data busses, and the output in unsigned decimal format



Screenshot of the console output

```
launch_simulation
INFO: [Vivado 12-5682] Launching behavioral simulation in 'D:/files/uni/digitalengineering/lab3/2-1-10_2-1-12/Performance_Check_On_Algorithm_Circuit/Performance_Check_On_Algorithm_Circuit.sim/sim
INFO: [SIM-utils-51] Simulation object is 'sim_1'
INFO: [SIM-utils-54] Inspecting design source files for 'two_process_TB' in fileset 'sim_1'...
INFO: [USF-XSim-97] Finding global include files...
INFO: [USF-XSim-98] Fetching design files from 'sim_1'...
INFO: [USF-XSim-2] XSim::Compile design
INFO: [USF-XSim-61] Executing 'COMPILE and ANALYZE' step in 'D:/files/uni/digitalengineering/lab3/2-1-10_2-1-12/Performance_Check_On_Algorithm_Circuit/Performance_Check_On_Algorithm_Circuit.sim/s
"xvhdl --incr --relax -prj two_process_TB_vhdl.prj"
INFO: [USF-XSim-69] 'compile' step finished in '2' seconds
INFO: [USF-XSim-3] XSim::Elaborate design
INFO: [USF-XSim-61] Executing 'ELABORATE' step in 'D:/files/uni/digitalengineering/lab3/2-1-10_2-1-12/Performance_Check_On_Algorithm_Circuit/Performance_Check_On_Algorithm_Circuit.sim/sim_1/behav
Vivado Simulator 2017.4
Copyright 1986-1999, 2001-2016 Xilinx, Inc. All Rights Reserved.
Running: D:/programs/uni/xilinx/Vivado/Vivado/2017.4/bin/unwrapped/win64.o/xelab.exe -wto 14a433ed9f5e47a497118842e777e334 --incr --debug typical --relax --mt 2 -L xil_defaultlib -L secureip --sn
Using 2 slave threads.
Starting static elaboration
Completed static elaboration
INFO: [XSIM 43-4323] No Change in HDL. Linking previously generated obj files to create kernel
INFO: [USF-XSim-69] 'elaborate' step finished in '2' seconds
INFO: [USF-XSim-4] XSim::Simulate design
INFO: [USF-XSim-61] Executing 'SIMULATE' step in 'D:/files/uni/digitalengineering/lab3/2-1-10_2-1-12/Performance_Check_On_Algorithm_Circuit/Performance_Check_On_Algorithm_Circuit.sim/sim_1/behav/
INFO: [USF-XSim-98] *** Running xsim
   with args "two_process_TB_behav -key {Behavioral:sim_1:Functional:two_process_TB} -tclbatch {two_process_TB.tcl} -view {D:/files/uni/digitalengineering/lab3/2-1-10_2-1-12/Performance_Check_On_
INFO: [USF-XSim-8] Loading simulator feature
Vivado Simulator 2017.4
Time resolution is 1 ps
open_wave_config D:/files/uni/digitalengineering/lab3/2-1-10_2-1-12/Performance_Check_On_Algorithm_Circuit/two_process_TB_behav.wcfg
source two_process_TB.tcl
# set curr_wave [current_wave_config]
# if { [string length $curr_wave] == 0 } {
#   if { [llength [get_objects]] > 0} {
#     add_wave /
#     set_property needs_save false [current_wave_config]
#   } else {
#      send_msg_id Add_Wave-1 WARNING "No top level signals found. Simulator will start without a wave window. If you want to open a wave window go to 'File->New Waveform Configuration' or type '
#   }
# }
# run 1000ns
INFO: [USF-XSim-96] XSim completed. Design snapshot 'two_process_TB_behav' loaded.
INFO: [USF-XSim-97] XSim simulation ran for 1000ns
launch_simulation: Time (s): cpu = 00:00:02 ; elapsed = 00:00:06 . Memory (MB): peak = 1007.234 ; gain = 0.000
run 100 us
```

2.1.11: Write the best period where the constraints are met (i.e. the one just before it starts to fail) and print out a screenshot of the “Design Runs” tab, showing all columns up to “DSP”. What is the highest frequency at which your design should be able to run according to the WNS results?

The best period where the constraints are met

94ns

Screenshot of the "Design Runs" tab with all columns up to "DSP".

Name	Constraints	Status	WNS	TNS	WHS	THS	TPWS	Total Power	Failed Routes	LUT	FF	BRAMs	URAM	DSP
✓ synth_1	constrs_1	synth_design Complete!								897	293	0.00	0	0
✓ impl_1	constrs_1	route_design Complete!	0.642	0.000	0.055	0.000	0.000	0.110	0	887	293	0.00	0	0

Highest frequency at which your design should be able to run according to the WNS results?

The highest frequency is 10.64 MHz ($\frac{1}{94ns-0.059ns}$). If we are rounding to the nearest nanosecond the highest frequency is 10.64 MHz ($\frac{1}{94ns}$).

2.1.12: Print out (use a screenshot) the first few lines of the first Max Delay Path in the Post-Route Timing Report (see previous script for details). Can you identify where the new critical path lies, with respect to the pipeline above (i.e. in which pipeline stage) and which of the mathematical operation(s) in the algorithm are in the critical path? Based on the lecture material, what are you expecting to happen to the critical path, with respect to the circuit without this pipeline stage, and why? Does the practice match the theory (compare the new clock period to the one you obtained in step 2.1.8)? Provide answers to each of these questions.

Screenshot of the first few lines of the first Max Delay Path in the Post-Route Timing Report

```
Max Delay Paths
--------------------------------------------------------------------------------------
Slack (MET) :             0.642ns  (required time - arrival time)
  Source:                 STAGE3_INTD_reg[5]/C
                            (rising edge-triggered cell FDRE clocked by clk  {rise@0.000ns fall@47.000ns period=94.000ns})
  Destination:            STAGE4_INT4_reg[0]/D
                            (rising edge-triggered cell FDRE clocked by clk  {rise@0.000ns fall@47.000ns period=94.000ns})
  Path Group:             clk
  Path Type:              Setup (Max at Slow Process Corner)
  Requirement:            94.000ns  (clk rise@94.000ns - clk rise@0.000ns)
  Data Path Delay:        93.229ns  (logic 42.318ns (45.392%)  route 50.911ns (54.608%))
  Logic Levels:           185  (CARRY4=154 LUT1=1 LUT3=30)
```

Where the new critical path lies, with respect to the pipeline above?

It is in the same place as before. It goes between the copy of int D in the pipeline 1 register and the copy of int 4 in the pipeline 2 register.

Which of the mathematical operation(s) in the algorithm are in the critical path?

The path goes through the division operator.

What was being expected to happen to the critical path, with respect to the circuit without this pipeline stage, and why?

It should stay the same as the addition of the pipeline is not to the part of the circuit where the critical path is.

Does the practice match the theory (compare the new clock period to the one you obtained in step 2.1.8)?

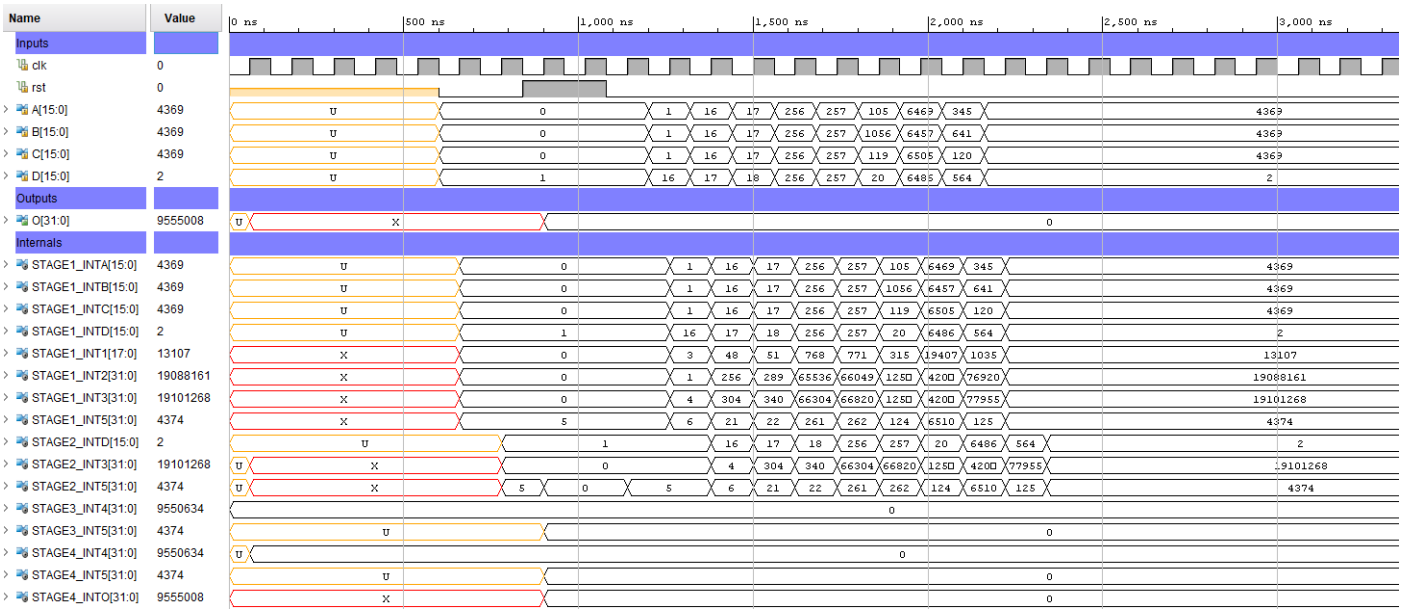
The practice does not match the theory as the clock period increased instead of staying the same.

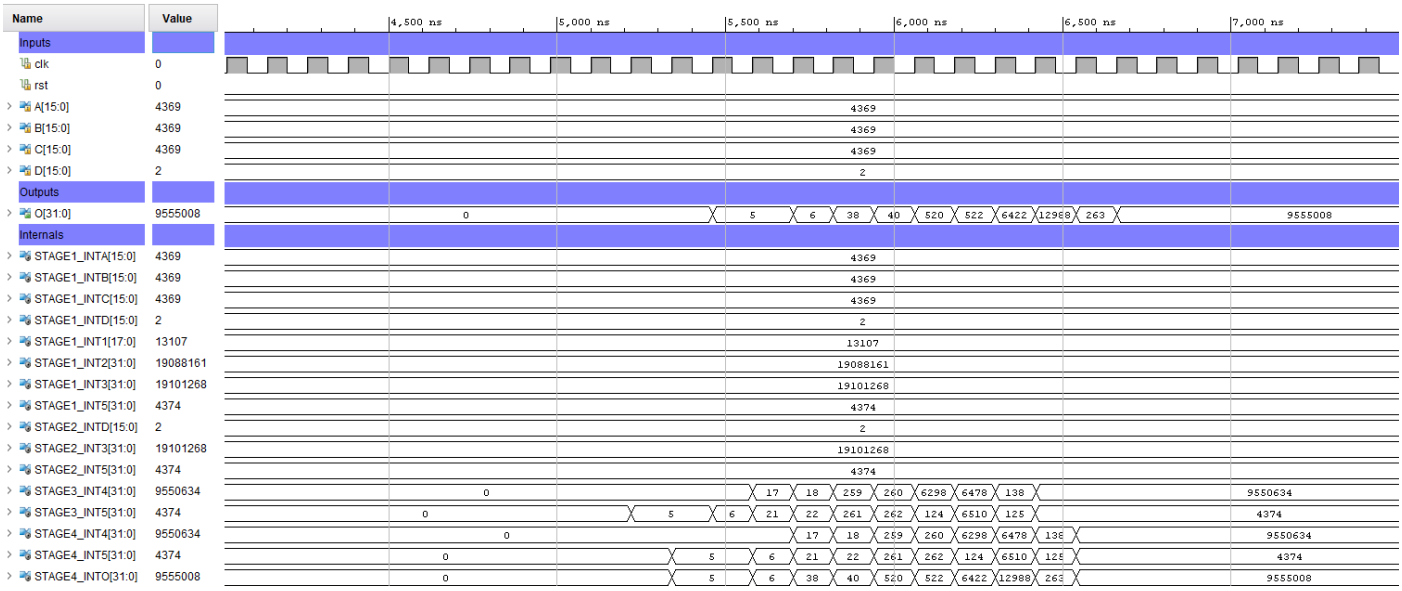
Lab 3 - Task B: IP Components

With 2 pipelines plus new divider

2.2.1: Screenshot of the simulation window, zoomed in to display with all inputs and the output in unsigned decimal format for the same sets of input (and output) values used in the preceding simulations. You will need two separate screenshots because of the depth of the pipeline (do not try to fit inputs and outputs in one screenshot!) Include the console output (as a separate screenshot).

2 screenshots of the simulation window with all inputs, internal data busses, and the output in unsigned decimal





Screenshot of the console output

```
launch_simulation
INFO: [Vivado 12-5698] Checking validity of IPs in the design for the 'XSim' simulator...
WARNING: [Runs 36-337] The following IPs are either missing output products or output products are not up-to-date for Simulation target. Since these IPs are loc
Please select 'Report IP Status' from the 'Tools/Report' menu or run Tcl command 'report_ip_status' for more information.
D:/files/uni/digitalengineering/lab3/2-2-1_2-2-3/Performance_Check_On_Algorithm_Circuit/Performance_Check_On_Algorithm_Circuit.srcs/sources_1/ip/divider/divider

INFO: [Vivado 12-5682] Launching behavioral simulation in 'D:/files/uni/digitalengineering/lab3/2-2-1_2-2-3/Performance_Check_On_Algorithm_Circuit/Performance_C
INFO: [Vivado 12-4795] Using compiled simulation libraries for IPs
INFO: [SIM-utils-51] Simulation object is 'sim_1'
INFO: [USF-XSim-7] Finding pre-compiled libraries...
INFO: [USF-XSim-11] File 'D:/programs/uni/xilinx/Vivado/Vivado/2017.4/data/xsim/ip/xsim_ip.ini' copied to run dir:'D:/files/uni/digitalengineering/lab3/2-2-1_2-
INFO: [SIM-utils-54] Inspecting design source files for 'two_process_TB' in fileset 'sim_1'...
WARNING: [SIM-utils-52] IP component XML file does not exist: 'd:/files/uni/digitalengineering/lab3/2-2-1_2-2-3/Performance_Check_On_Algorithm_Circuit/Performan
INFO: [USF-XSim-97] Finding global include files...
INFO: [USF-XSim-98] Fetching design files from 'sim_1'...
INFO: [USF-XSim-2] XSim::Compile design
INFO: [USF-XSim-61] Executing 'COMPILE and ANALYZE' step in 'D:/files/uni/digitalengineering/lab3/2-2-1_2-2-3/Performance_Check_On_Algorithm_Circuit/Performance
"xvhdl --incr --relax -prj two_process_TB_vhdl.prj"
INFO: [VRFC 10-163] Analyzing VHDL file "D:/files/uni/digitalengineering/lab3/2-2-1_2-2-3/Performance_Check_On_Algorithm_Circuit/Performance_Check_On_Algorithm_
INFO: [VRFC 10-307] analyzing entity algorithm
INFO: [VRFC 10-163] Analyzing VHDL file "D:/files/uni/digitalengineering/lab3/2-2-1_2-2-3/Performance_Check_On_Algorithm_Circuit/Performance_Check_On_Algorithm_
INFO: [VRFC 10-307] analyzing entity two_process_TB
INFO: [USF-XSim-69] 'compile' step finished in '2' seconds
INFO: [USF-XSim-3] XSim::Elaborate design
INFO: [USF-XSim-61] Executing 'ELABORATE' step in 'D:/files/uni/digitalengineering/lab3/2-2-1_2-2-3/Performance_Check_On_Algorithm_Circuit/Performance_Check_On_
Vivado Simulator 2017.4
Copyright 1986-1999, 2001-2016 Xilinx, Inc. All Rights Reserved.
Running: D:/programs/uni/xilinx/Vivado/Vivado/2017.4/bin/unwrapped/win64.o/xelab.exe -wto 14a433ed9f5e47a497118842e777e334 --incr --debug typical --relax --mt 2
Using 2 slave threads.
Starting static elaboration
Completed static elaboration
Starting simulation data flow analysis
Completed simulation data flow analysis
Time Resolution for simulation is 1ps
Compiling package std.standard
Compiling package std.textio
Compiling package ieee.std_logic_1164
Compiling package ieee.numeric_std
Compiling package unisim.vcomponents
Compiling package ieee.vital_timing
Compiling package ieee.vital_primitives
Compiling package unisim.vpkg
Using 2 slave threads.
Starting static elaboration
Completed static elaboration
Starting simulation data flow analysis
Completed simulation data flow analysis
Time Resolution for simulation is 1ps
Compiling package std.standard
Compiling package std.textio
Compiling package ieee.std_logic_1164
Compiling package ieee.numeric_std
Compiling package unisim.vcomponents
Compiling package ieee.vital_timing
Compiling package ieee.vital_primitives
Compiling package unisim.vpkg
Compiling architecture vcc_v of entity unisim.VCC [vcc_default]
Compiling architecture gnd_v of entity unisim.GND [gnd_default]
Compiling architecture inv_v of entity unisim.INV [inv_default]
Compiling architecture lut3_v of entity unisim.LUT3 [\LUT3(init="01101010")(0,7)\]
Compiling architecture lut3_v of entity unisim.LUT3 [\LUT3(init="01101001")(0,7)\]
Compiling architecture lut2_v of entity unisim.LUT2 [\LUT2(init="1001")(0,3)\]
Compiling architecture lut1_v of entity unisim.LUT1 [\LUT1(init="0001")(0,3)\]
Compiling architecture fdre_v of entity unisim.FDRE [fdre_default]
Compiling architecture fdr_v of entity unisim.FDR [fdr_default]
Compiling architecture lut2_v of entity unisim.LUT2 [\LUT2(init="1000")(0,3)\]
Compiling architecture mult_and_v of entity unisim.MULT_AND [mult_and_default]
Compiling architecture muxcy_v of entity unisim.MUXCY [muxcy_default]
Compiling architecture xorcy_v of entity unisim.XORCY [xorcy_default]
Compiling architecture fdse_v of entity unisim.FDSE [fdse_default]
Compiling architecture fds_v of entity unisim.FDS [fds_default]
Compiling architecture structure of entity xil_defaultlib.divider [divider_default]
Compiling architecture behavioral of entity xil_defaultlib.algorithm [algorithm_default]
Compiling architecture behavior of entity xil_defaultlib.two_process_tb
Built simulation snapshot two_process_TB_behav
run_program: Time (s): cpu = 00:00:00 ; elapsed = 00:00:08 . Memory (MB): peak = 1058.984 ; gain = 0.000
INFO: [USF-XSim-69] 'elaborate' step finished in '8' seconds
INFO: [USF-XSim-4] XSim::Simulate design
INFO: [USF-XSim-61] Executing 'SIMULATE' step in 'D:/files/uni/digitalengineering/lab3/2-2-1_2-2-3/Performance_Check_On_Algorithm_Circuit/Performance_Check_On_A
INFO: [USF-XSim-98] *** Running xsim
   with args "two_process_TB_behav -key {Behavioral:sim_1:Functional:two_process_TB} -tclbatch {two_process_TB.tcl} -view {D:/files/uni/digitalengineering/lab3/
INFO: [USF-XSim-8] Loading simulator feature
Vivado Simulator 2017.4
Time resolution is 1 ps
open_wave_config D:/files/uni/digitalengineering/lab3/2-2-1_2-2-3/Performance_Check_On_Algorithm_Circuit/two_process_TB_behav.wcfg
source two_process_TB.tcl
source two_process_TB.tcl
# set curr_wave [current_wave_config]
# if { [string length $curr_wave] == 0 } {
#   if { [llength [get_objects]] > 0} {
#     add_wave /
#     set_property needs_save false [current_wave_config]
#   } else {
#      send_msg_id Add_Wave-1 WARNING "No top level signals found. Simulator will start without a wave window. If you want to open a wave window go to 'File->New Waveform Configuration' or type '
#   }
# }
# run 1000ns
INFO: [USF-XSim-96] XSim completed. Design snapshot 'two_process_TB_behav' loaded.
INFO: [USF-XSim-97] XSim simulation ran for 1000ns
launch_simulation: Time (s): cpu = 00:00:03 ; elapsed = 00:00:12 . Memory (MB): peak = 1063.754 ; gain = 4.770
run 100 us
```

2.2.2: Write the best period where the constraints are met (i.e. the one just before it starts to fail) and print out a screenshot of the "Design Runs" tab, showing all columns up to "DSP" (note the massive increase in FFs!) What is the highest frequency at which your design should be able to run according to the WNS results?

The best period where the constraints are met

14ns

Screenshot of the "Design Runs" tab with all columns up to "DSP".

Name	Constraints	Status	WNS	TNS	WHS	THS	TPWS	Total Power	Failed Routes	LUT	FF	BRAMs	URAM	DSP
synth_1	constrs_1	Synthesis Out-of-date								1019	2423	0.00	0	0
impl_1	constrs_1	Implementation Out-of-date	0.679	0.000	-0.069	-0.251	0.000	0.151	0	991	2341	0.00	0	0

Highest frequency at which your design should be able to run according to the WNS results?

The highest frequency is 75.06 MHz ($\frac{1}{14ns-0.679ns}$). If we are rounding to the nearest nanosecond the highest frequency is 71.43 MHz ($\frac{1}{14ns}$).

2.2.3: Print out (use a screenshot) the first few lines of the first Max Delay Path in the Post-Route Timing Report (see previous script for details). Can you identify where the new critical path lies, with respect to the pipeline above (i.e. in which pipeline stage) and which of the mathematical operation(s) in the algorithm are in the critical path?

Screenshot of the first few lines of the first Max Delay Path in the Post-Route Timing Report

```
Max Delay Paths
--------------------------------------------------------------------------------------
Slack (MET) :             0.679ns  (required time - arrival time)
  Source:                 STAGE1_INT1B_reg[7]/C
                            (rising edge-triggered cell FDRE clocked by clk  {rise@0.000ns fall@7.000ns peri
  Destination:            STAGE2_INT3_reg[31]/D
                            (rising edge-triggered cell FDRE clocked by clk  {rise@0.000ns fall@7.000ns peri
  Path Group:             clk
  Path Type:              Setup (Max at Slow Process Corner)
  Requirement:            14.000ns  (clk rise@14.000ns - clk rise@0.000ns)
  Data Path Delay:        13.337ns  (logic 5.201ns (38.995%)  route 8.136ns (61.005%))
  Logic Levels:           13  (CARRY4=7 LUT3=1 LUT4=1 LUT6=4)
```

Where the new critical path lies, with respect to the pipeline above?

The critical path is now from integer B in the input register 1 to integer 3 in the pipeline 1 register.

Which of the mathematical operation(s) in the algorithm are in the critical path?

The path goes through a multiplication operator and an addition operator.

What was being expected to happen to the critical path, with respect to the circuit without this pipeline stage, and why?

It was expected to significantly decrease in duration, as we have split up the longest operator, the division operator.

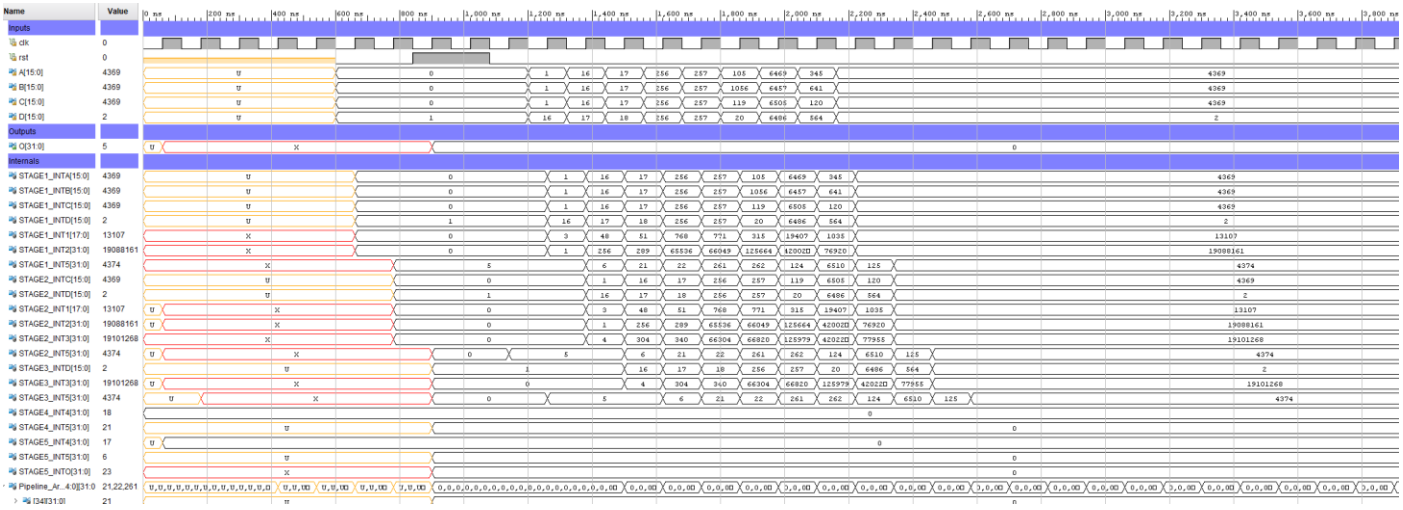
Does the practice match the theory (compare the new clock period to the one you obtained in step 2.1.8)?

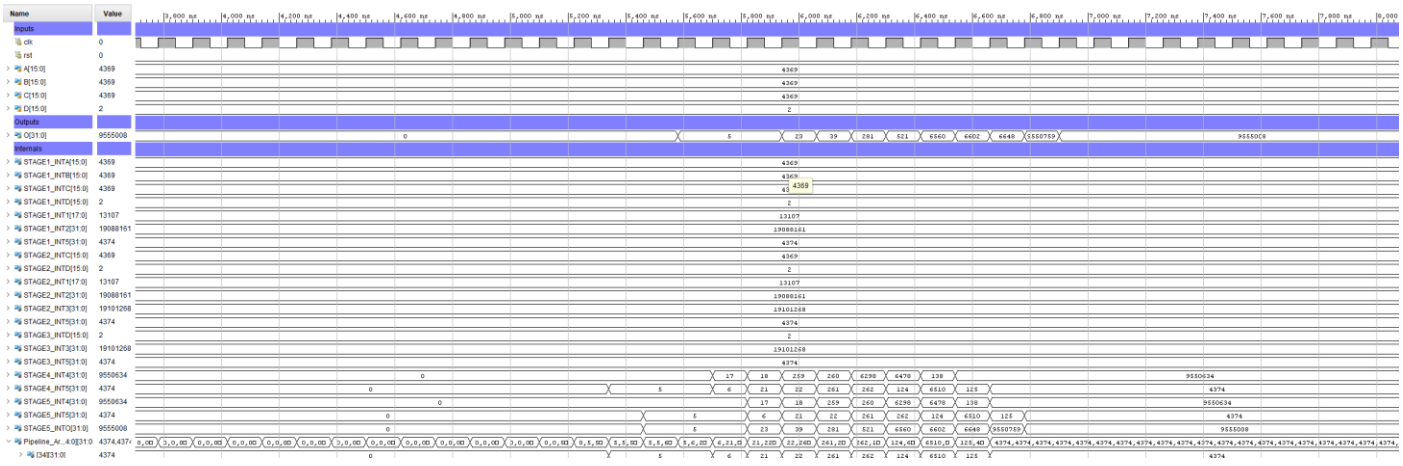
Yes. There is a very large decrease from 90ns to 14ns.

With 3 pipelines plus new divider

2.2.4: Screenshot of the simulation window, zoomed in to display with all inputs and the output in unsigned decimal format for the same sets of input (and output) values used in the preceding simulations. You will need two separate screenshots because of the depth of the pipeline (do not try to fit inputs and outputs in one screenshot!) Include the console output (as a separate screenshot).

2 screenshots of the simulation window with all inputs, internal data busses, and the output in unsigned decimal format used in the preceding simulations





Screenshot of the console output

```
launch_simulation
INFO: [Vivado 12-5698] Checking validity of IPs in the design for the 'XSim' simulator...
WARNING: [Runs 36-337] The following IPs are either missing output products or output products are not up-to-date for Simulation target. Since
Please select 'Report IP Status' from the 'Tools/Report' menu or run Tcl command 'report_ip_status' for more information.
H:/Downloads/lab3/Performance_Check_On_Algorithm_Circuit/Performance_Check_On_Algorithm_Circuit.srcs/sources_1/ip/divider/divider.xco

INFO: [Vivado 12-5682] Launching behavioral simulation in 'H:/Downloads/lab3/Performance_Check_On_Algorithm_Circuit/Performance_Check_On_Algor
INFO: [Vivado 12-4795] Using compiled simulation libraries for IPs
INFO: [SIM-utils-51] Simulation object is 'sim_1'
INFO: [USF-XSim-7] Finding pre-compiled libraries...
INFO: [USF-XSim-11] File 'C:/Xilinx/Vivado/2017.4/data/xsim/ip/xsim_ip.ini' copied to run dir:'H:/Downloads/lab3/Performance_Check_On_Algorith
INFO: [SIM-utils-54] Inspecting design source files for 'two_process_TB' in fileset 'sim_1'...
WARNING: [SIM-utils-52] IP component XML file does not exist: 'h:/Downloads/lab3/Performance_Check_On_Algorithm_Circuit/Performance_Check_On_A
INFO: [USF-XSim-97] Finding global include files...
INFO: [USF-XSim-98] Fetching design files from 'sim_1'...
INFO: [USF-XSim-2] XSim::Compile design
INFO: [USF-XSim-61] Executing 'COMPILE and ANALYZE' step in 'H:/Downloads/lab3/Performance_Check_On_Algorithm_Circuit/Performance_Check_On_Alg
"xvhdl --incr --relax -prj two_process_TB_vhdl.prj"
INFO: [USF-XSim-69] 'compile' step finished in '2' seconds
INFO: [USF-XSim-3] XSim::Elaborate design
INFO: [USF-XSim-61] Executing 'ELABORATE' step in 'H:/Downloads/lab3/Performance_Check_On_Algorithm_Circuit/Performance_Check_On_Algorithm_Cir
Vivado Simulator 2017.4
Copyright 1986-1999, 2001-2016 Xilinx, Inc. All Rights Reserved.
Running: C:/Xilinx/Vivado/2017.4/bin/unwrapped/win64.o/xelab.exe -wto 14a433ed9f5e47a497118842e777e334 --incr --debug typical --relax --mt 2 -
Using 2 slave threads.
Starting static elaboration
Completed static elaboration
INFO: [XSIM 43-4323] No Change in HDL. Linking previously generated obj files to create kernel
INFO: [USF-XSim-69] 'elaborate' step finished in '3' seconds
INFO: [USF-XSim-4] XSim::Simulate design
INFO: [USF-XSim-61] Executing 'SIMULATE' step in 'H:/Downloads/lab3/Performance_Check_On_Algorithm_Circuit/Performance_Check_On_Algorithm_Circ
INFO: [USF-XSim-98] *** Running xsim
   with args "two_process_TB_behav -key {Behavioral:sim_1:Functional:two_process_TB} -tclbatch {two_process_TB.tcl} -view {H:/Downloads/lab3/P
INFO: [USF-XSim-8] Loading simulator feature
Vivado Simulator 2017.4
Time resolution is 1 ps
open_wave_config H:/Downloads/lab3/Performance_Check_On_Algorithm_Circuit/two_process_TB_behav.wcfg
source two_process_TB.tcl
# set curr_wave [current_wave_config]
# if { [string length $curr_wave] == 0 } {
#   if { [llength [get_objects]] > 0} {

#     add_wave /
#     set_property needs_save false [current_wave_config]
#   } else {
#      send_msg_id Add_Wave-1 WARNING "No top level signals found. Simulator will start without a wave window. If you want to open a wave wi
#   }
# }
# run 1000ns
INFO: [USF-XSim-96] XSim completed. Design snapshot 'two_process_TB_behav' loaded.
INFO: [USF-XSim-97] XSim simulation ran for 1000ns
launch_simulation: Time (s): cpu = 00:00:03 ; elapsed = 00:00:08 . Memory (MB): peak = 915.988 ; gain = 2.328
run 5 us
```

2.2.5: Write the best period where the constraints are met (i.e. the one just before it starts to fail) and print out a screenshot of the “Design Runs” tab, showing all columns up to “DSP” (note the massive increase in FFs!) What is the highest frequency at which your design should be able to run according to the WNS results?

The best period where the constraints are met

13ns.

Screenshot of the “Design Runs” tab with all columns up to “DSP”.

Name	Constraints	Status	WNS	TNS	WHS	THS	TPWS	Total Power	Failed Routes	LUT	FF	BRAMs	URAM	DSP
synth_1	constrs_1	synth_design Complete!								1021	2506	0.00	0	0
impl_1	constrs_1	route_design Complete, Failed Timing!	0.847	0.000	-0.083	-0.217	0.000	0.155	0	993	2424	0.00	0	0

Highest frequency at which your design should be able to run according to the WNS results?

The highest frequency is 82.28 MHz ($\frac{1}{13ns-0.847ns}$). If we are rounding to the nearest nanosecond the highest frequency is 76.92 MHz ($\frac{1}{13ns}$).

2.2.6: Print out (use a screenshot) the first few lines of the first Max Delay Path in the Post-Route Timing Report (see previous script for details). Can you identify where the new critical path lies, with respect to the pipeline above (i.e. in which pipeline stage) and which of the mathematical operation(s) in the algorithm are in the critical path?

Screenshot of the first few lines of the first Max Delay Path in the Post-Route Timing Report

```
Max Delay Paths
--------------------------------------------------------------------------------
Slack (MET) :             0.847ns  (required time - arrival time)
  Source:                 STAGE1_INTB_reg[11]/C
                            (rising edge-triggered cell FDRE clocked by clk  {rise
  Destination:            STAGE2_INT2_reg[29]/D
                            (rising edge-triggered cell FDRE clocked by clk  {rise
  Path Group:             clk
  Path Type:              Setup (Max at Slow Process Corner)
  Requirement:            13.000ns  (clk rise@13.000ns - clk rise@0.000ns)
  Data Path Delay:        12.120ns  (logic 3.800ns (31.353%)  route 8.320ns (68.64
  Logic Levels:           12  (CARRY4=6 LUT2=1 LUT4=1 LUT5=2 LUT6=2)
```

The critical path is now from integer B in the input register 1 to integer 2 in the pipeline 3 (named pipeline 0 in the code) register. It is the same path as before but now it is only the first part of it.

Which of the mathematical operation(s) in the algorithm are in the critical path?

The path goes through a multiplication operator.

What was being expected to happen to the critical path, with respect to the circuit without this pipeline stage, and why?

It was expected to decrease in duration, as we have split up the previous critical path.

Does the practice match the theory (compare the new clock period to the one you obtained in step 2.1.8)?

Yes. There is a decrease from 14ns to 13ns, although it is not much.

With 3 pipelines plus new divider and max_dsp set back to -1

2.2.7 Write the best period where the constraints are met (i.e. the one just before it starts to fail) and print out a screenshot of the "Design Runs" tab, showing all columns up to "DSP" (note the massive increase in FFs!) What is the highest frequency at which your design should be able to run according to the WNS results?

The best period where the constraints are met

4ns

Screenshot of the "Design Runs" tab with all columns up to "DSP".

Name	Constraints	Status	WNS	TNS	WHS	THS	TPWS	Total Power	Failed Routes	LUT	FF	BRAMs	URAM	DSP
✓ synth_2	constrs_1	synth_design Complete!								685	2392	0.00	0	2
✓ impl_2	constrs_1	route_design Complete!	0.125	0.000	0.035	0.000	0.000	0.261	0	668	2310	0.00	0	2

Highest frequency at which your design should be able to run according to the WNS results?

The highest frequency is 82.28 MHz ($\frac{1}{4ns-0.847ns}$). If we are rounding to the nearest nanosecond the highest frequency is 250 MHz ($\frac{1}{4ns}$).

2.2.8: Print out (use a screenshot) the first few lines of the first Max Delay Path in the Post-Route Timing Report (see previous script for details). Can you identify where the new critical path lies, with respect to the pipeline above (i.e. in which pipeline stage) and which of the mathematical operation(s) in the algorithm are in the critical path?

Screenshot of the first few lines of the first Max Delay Path in the Post-Route Timing Report

```
Max Delay Paths
--------------------------------------------------------------------------------
Slack (MET) :             0.125ns  (required time - arrival time)
  Source:                 STAGE2_INT1_reg/CLK
                            (rising edge-triggered cell DSP48E1 clocked by clk  {ri
  Destination:            STAGE3_INT3_reg/PCIN[0]
                            (rising edge-triggered cell DSP48E1 clocked by clk  {ri
  Path Group:             clk
  Path Type:              Setup (Max at Slow Process Corner)
  Requirement:            4.000ns  (clk rise@4.000ns - clk rise@0.000ns)
  Data Path Delay:        2.413ns  (logic 2.411ns (99.917%)  route 0.002ns (0.083%)
  Logic Levels:           0
```

Where the new critical path lies, with respect to the pipeline above?

The new critical path is between int 1 on the pipeline 3 register (pipeline 0 in the code) and int 3 on the pipeline 1 register.

Which of the mathematical operation(s) in the algorithm are in the critical path?

The critical path goes through the addition operator.

What was being expected to happen to the critical path, with respect to the circuit without this pipeline stage, and why?

The critical path was expected to be shorter as more block DSPs are allowed.

Does the practice match the theory (compare the new clock period to the one you obtained in step 2.2.5)?

Yes. The critical path is significantly shorter. It also moved.

2.2.9: Print out the commented VHDL code at this step and the “RTL Component Statistics” and “RTL Hierarchical Component Statistics” part of the synthesis report. If you have used a parameterizable register for your pipeline stages, make sure to include the code. If any other components have been created (there should be no need for any), they should also be included.

Algorithm Code

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.NUMERIC_STD.ALL;

-- Entity description:
-- The entity implements, with no optimization, a sequence of operations:
--      O <= (A*3 + B*C)/D + C + 5
-- where A,B,C, and D are UNSIGNED vectors of parameterizable size

-- Note 1: There is no particular "meaning" to the equation - it is designed for
--  experimentation with logic optimization for performance
-- Note 2: There is no provision for overflow. Some input vectors can cause
--  overflow and the result will be incorrect.
-- Note 3: Inputs and outputs are registered (rising edge, synchronous reset).
--  This introduces a latency of 2 clock cycles between inputs and outputs.
-- Note 4: D is the divisor in one of the operations, so can never have value 0

entity algorithm is
    generic (data_size : integer := 16); -- defines the size of the data
    Port ( clk : in  STD_LOGIC;
           rst : in  STD_LOGIC;
           -- The four (parameterizable) data inputs
           A : in  STD_LOGIC_VECTOR (data_size-1 downto 0);
           B : in  STD_LOGIC_VECTOR (data_size-1 downto 0);
           C : in  STD_LOGIC_VECTOR (data_size-1 downto 0);
           D : in  STD_LOGIC_VECTOR (data_size-1 downto 0);
```

```
            -- Output = (A*3 + B*C)/D + C +5
            O : out  STD_LOGIC_VECTOR (data_size*2-1 downto 0)
            );
end algorithm;

architecture Behavioral of algorithm is

constant latency : integer := 34;

-- More efficient divider
component divider
port (
    clk: in std_logic;
    sclr: in std_logic;
    rfd: out std_logic;
    dividend: in std_logic_vector(31 downto 0);
    divisor: in std_logic_vector(15 downto 0);
    quotient: out std_logic_vector(31 downto 0);
    fractional: out std_logic_vector(15 downto 0)
);
end component;

-- Stage 1
signal STAGE1_INTA : UNSIGNED (data_size-1 downto 0);
signal STAGE1_INTB : UNSIGNED (data_size-1 downto 0);
signal STAGE1_INTC : UNSIGNED (data_size-1 downto 0);
signal STAGE1_INTD : UNSIGNED (data_size-1 downto 0);
signal STAGE1_INT1 : UNSIGNED (data_size+1 downto 0);
signal STAGE1_INT2 : UNSIGNED (data_size*2-1 downto 0);
signal STAGE1_INT5 : UNSIGNED (data_size*2-1 downto 0);

-- Stage 2
signal STAGE2_INTC : UNSIGNED (data_size-1 downto 0);
signal STAGE2_INTD : UNSIGNED (data_size-1 downto 0);
signal STAGE2_INT1 : UNSIGNED (data_size+1 downto 0);
signal STAGE2_INT2 : UNSIGNED (data_size*2-1 downto 0);
signal STAGE2_INT3 : UNSIGNED (data_size*2-1 downto 0);
signal STAGE2_INT5 : UNSIGNED (data_size*2-1 downto 0);

-- Stage 3
signal STAGE3_INTD : UNSIGNED (data_size-1 downto 0);
signal STAGE3_INT3 : UNSIGNED (data_size*2-1 downto 0);
signal STAGE3_INT5 : UNSIGNED (data_size*2-1 downto 0);

-- Stage 4
signal STAGE4_INT4 : UNSIGNED (data_size*2-1 downto 0);
signal STAGE4_INT5 : UNSIGNED (data_size*2-1 downto 0);

-- Stage 5
signal STAGE5_INT4 : UNSIGNED (data_size*2-1 downto 0);
signal STAGE5_INT5 : UNSIGNED (data_size*2-1 downto 0);
signal STAGE5_INTO : UNSIGNED (data_size*2-1 downto 0);

-- Pipeline array
type Pipeline_Array_Type is ARRAY (LATENCY downto 0)
  of STD_LOGIC_VECTOR (data_size*2-1 downto 0);
signal Pipeline_Array_Internal : Pipeline_Array_Type;

begin

-- Input registers (D-type, rising edge, synchronous reset)
input_regs: process (clk) is
begin
  if rising_edge(clk) then
    if rst = '1' then
      STAGE1_INTA <= (others => '0');
      STAGE1_INTB <= (others => '0');
      STAGE1_INTC <= (others => '0');
      STAGE1_INTD <= to_unsigned(1,STAGE1_INTD'length); -- type conversion notation
    else
      STAGE1_INTA <= unsigned(A);
      STAGE1_INTB <= unsigned(B);
      STAGE1_INTC <= unsigned(C);
      STAGE1_INTD <= unsigned(D);
    end if;
```

```
  end if;
end process input_regs;

-- Stage 1
STAGE1_INT1 <= STAGE1_INTA * to_unsigned(3, 2);
STAGE1_INT2 <= STAGE1_INTB * Stage1_INTC;
STAGE1_INT5 <= STAGE2_INTC + to_unsigned(5, STAGE2_INT5'length);

-- Pipeline 0 registers (D-type, rising edge, syncronous reset)
-- Called pipeline 3 in the script but that's confusing
pipeline_0_regs: process (clk) is
begin
  if rising_edge(clk) then
    if rst = '1' then
      STAGE2_INTC <= (others => '0');
      STAGE2_INTD <= to_unsigned(1, STAGE2_INTD'length);
      STAGE2_INT1 <= (others => '0');
      STAGE2_INT2 <= (others => '0');
      STAGE2_INT5 <= (others => '0');
    else
      STAGE2_INTC <= STAGE1_INTC;
      STAGE2_INTD <= STAGE1_INTD;
      STAGE2_INT1 <= STAGE1_INT1;
      STAGE2_INT2 <= STAGE1_INT2;
      STAGE2_INT5 <= STAGE1_INT5;
    end if;
  end if;
end process pipeline_0_regs;

-- Stage2
STAGE2_INT3 <= STAGE2_INT1 + Stage2_INT2;

-- Pipeline 1 registers (D-type, rising edge, syncronous reset)
pipeline_1_regs: process (clk) is
begin
  if rising_edge(clk) then
    if rst = '1' then
      STAGE3_INT3 <= (others => '0');
      STAGE3_INTD <= to_unsigned(1, STAGE2_INTD'length);
      STAGE3_INT5 <= (others => '0');
    else
      STAGE3_INT3 <= STAGE2_INT3;
      STAGE3_INTD <= STAGE2_INTD;
      STAGE3_INT5 <= STAGE2_INT5;
    end if;
  end if;
end process pipeline_1_regs;

-- Stage 3 has no combinational logic

my_divider : divider
port map (
    clk => clk,
    sclr => rst,
    dividend => std_logic_vector(STAGE3_INT3),
    divisor => std_logic_vector(STAGE3_INTD),
    unsigned(quotient) => STAGE4_INT4
);

Pipeline_Array_Internal(0) <= std_logic_vector(STAGE3_INT5);
STAGE4_INT5 <= unsigned(Pipeline_Array_Internal(LATENCY));
pipeline_array:
for I in 0 to LATENCY-1 generate
    pipeline_array_pipeline: process (clk) is
    begin
        if rising_edge(clk) then
            if rst = '1' then
                Pipeline_Array_Internal(I+1) <= (others => '0');
            else
                Pipeline_Array_Internal(I+1) <= Pipeline_Array_Internal(I);
            end if;
        end if;
    end process pipeline_array_pipeline;
end generate pipeline_array;
```

```
-- Stage 4 has no combinational logic

-- Pipeline 2 registers (D-type, rising edge, syncronous reset)
pipeline_2_regs: process (clk) is
begin
  if rising_edge(clk) then
    if rst = '1' then
      STAGE5_INT4 <= (others => '0');
      STAGE5_INT5 <= (others => '0');
    else
      STAGE5_INT4 <= STAGE4_INT4;
      STAGE5_INT5 <= STAGE4_INT5;
    end if;
  end if;
end process pipeline_2_regs;

-- Stage 5
STAGE5_INTO <= STAGE5_INT5 + STAGE5_INT4;

-- Output registers (D-type, rising edge, synchronous reset)
output_regs: process (clk) is
begin
  if rising_edge(clk) then
    if rst = '1' then
      O <= (others => '0');
    else
      O <= std_logic_vector(STAGE5_INTO);
    end if;
  end if;
end process output_regs;
end Behavioral;
```

Testbench Code

```
library ieee;
use ieee.std_logic_1164.ALL;
use ieee.numeric_std.ALL;
entity two_process_TB IS

end two_process_TB;
architecture behavior OF two_process_TB IS

    -- Testing Strategy:
    --   This is a two process test bench. This means that it consists of a set of test
       --    vectors and two processes. One of these processes sets the inputs based on the
       --    test vectors and one of the processes checks that the outputs match what is
       --    expected as specified in the test vectors.
    --   There are 10 test vectors in total. They consist of a range of values designed
       --    so that all the different input variables are tested along with the floor
       --    division functionality.

    -  Inputs
    signal A : std_logic_vector(15 downto 0);
    signal B : std_logic_vector(15 downto 0);
    signal C : std_logic_vector(15 downto 0);
    signal D : std_logic_vector(15 downto 0);
    signal clk : std_logic;
    signal rst : std_logic;

    -  Outputs
    signal O : std_logic_vector(31 downto 0);
    constant clk_period : time := 120ns; -- clock period
    constant wait_period: time := 500ns;
    constant latency : natural := 39; -- (input register, output register, 3 pipeline
                                      --  registers, divider/pipline array containing 34
                                      --  registers)

    -- Define and create a record of test patterns to test the circuit
    type test_vector is record
        A : std_logic_vector(15 downto 0);
        B : std_logic_vector(15 downto 0);
        C : std_logic_vector(15 downto 0);
        D : std_logic_vector(15 downto 0);
        O : std_logic_vector(31 downto 0);
    end record;
```

```
    type test_vector_array is array
        (natural range <>) of test_vector;
    constant test_vectors : test_vector_array := (
        -- A, B, C, D, O
        (X"0000", X"0000", X"0000", X"0001", X"00000005"),
        (X"0001", X"0001", X"0001", X"0010", X"00000006"),
        (X"0010", X"0010", X"0010", X"0011", X"00000026"),
        (X"0011", X"0011", X"0011", X"0012", X"00000028"),
        (X"0100", X"0100", X"0100", X"0100", X"00000208"),
        (X"0101", X"0101", X"0101", X"0101", X"0000020A"),
        (X"0069", X"0420", X"0077", X"0014", X"00001916"),
        (X"1945", X"1939", X"1969", X"1956", X"000032BC"),
        (X"0159", X"0281", X"0078", X"0234", X"00000107"),
        (X"1111", X"1111", X"1111", X"0002", X"0091CC40"));

BEGIN
    -- Instantiate the Unit Under Test (UUT)
    uut: entity work.algorithm
    PORT MAP (
        clk => clk,
        rst => rst,
        A => A,
        B => B,
        C => C,
        D => D,
        O => O
    );

    -- Clock process
    clkProcess : process
    begin
        clk <= '0';
        wait for clk_period/2;

        clk <= '1';
        wait for clk_period/2;
    end process;

    set_inputs: process
    begin
           -- Initial pause followed by syncing to the falling edge
        wait for wait_period;
        wait until falling_edge(clk);

        -- Initialise inputs
        rst <= '0';
        A <= x"0000";
        B <= x"0000";
        C <= x"0000";
        D <= x"0001";

        -- Initial reset
        wait for clk_period*2;
        rst <= '1';
        wait for clk_period*2;
        rst <= '0';

        -- test pattern input loop
        for i in test_vectors'range loop
            A <= test_vectors(i).A;
            B <= test_vectors(i).B;
            C <= test_vectors(i).C;
            D <= test_vectors(i).D;
            wait for clk_period;
        end loop;
        wait;
    end process;

    check_outputs: process
    begin
           -- Initial pause followed by syncing to the falling edge
        wait for wait_period;
        wait until falling_edge(clk);

           -- Pause to account for  the delay between input and output
```

```
        wait for clk_period*latency;

           -- Pause to account for the timing of pressing the reset button
        wait for clk_period*2;
        wait for clk_period*2;

        -- test pattern check loop
        for i in test_vectors'range loop
            assert (O = test_vectors(i).O)
            report "Test vector failed"
            severity error;
            wait for clk_period;
        end loop;
        wait;
    end process;
end;
```

RTL Component Statistics

```
---------------------------------------------------------------------------------
Start RTL Component Statistics
---------------------------------------------------------------------------------
Detailed RTL Component Info :
+---Adders :
           2 Input    32 Bit      Adders := 1
           2 Input    17 Bit      Adders := 1
+---Registers :
                 32 Bit    Registers := 39
                 16 Bit    Registers := 7
---------------------------------------------------------------------------------
Finished RTL Component Statistics
---------------------------------------------------------------------------------
```

RTL Hierarchical Component Statistics

```
---------------------------------------------------------------------------------
Start RTL Hierarchical Component Statistics
---------------------------------------------------------------------------------
Hierarchical RTL Component report
Module algorithm
Detailed RTL Component Info :
+---Adders :
           2 Input    32 Bit      Adders := 1
           2 Input    17 Bit      Adders := 1
+---Registers :
                 32 Bit    Registers := 39
                 16 Bit    Registers := 7
---------------------------------------------------------------------------------
Finished RTL Hierarchical Component Statistics
---------------------------------------------------------------------------------
```