# 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

1	lame	Value	60	0 ns	800 ns	1,000 ns	1,200 ns	1,400	ns  1	.,600 ns	1,80	0 ns	2,000 n	s  2	,200 ns  2,400 ns
	Inputs														
	₩ clk	0													
	₩ rst	0													
>	₹ A[15:0]	4369	U		0	X	3 X	16 .	17 / 2	56 2	57 🗶 10	05 (64)	59 🗶 34	15	4369
>	₹ B[15:0]	4369	U		0	X	1	16 )	17 X 2	56 2	57 🗶 10	56 64	57 X 64	11	4369
>	₹ C[15:0]	4369	U		0	X	1	16 )	17 X 2	56 2	57 🗶 1:	19 ( 651	05 X 12	20 X	4369
>	₹ D[15:0]	2	U		1	X	10	17 X :	18 / 2	56 2	57 🗶 2	0 (64)	36 X 56	54 X	2
	Outputs														
>	₹ O[31:0]	9555008	х		5	o X	5	X 7	Х 38	40	520	522	6422	(12988)	263 9555008
	Internal Signals														
>	₹ INTA[15:0]	4369	U		0		Х з	X 16	17	256	287	105	6469	345	4369
>	₹ INTB[15:0]	4369	U		0		1	16	17	256	257	1056	6457	641	4369
>	₹ INTC[15:0]	4369	υ		0		1	X 16	17	256	257	119	6505	120	4369
>	₹ INTD[15:0]	2	υ		1		10	17	18	256	257	20	6486	564	2
>	₹ INT1[17:0]	13107	Х	X	0		9	X 48	51	768	771	315	19407	1035	13107
>	₹ INT2[31:0]	19088161	Х	X	0		1	256	289	65536	66049	125664	420020	76920	19088161
>	₹ INT3[31:0]	19101268	Х	X	0		10	304	340	66304	66820	125979	420220	77955	19101268
>	₹ INT4[31:0]	9550634	Х		0		1	X 17	18	259	260	6298	6478	138	9550634
>	₹ INT5[31:0]	4374	Х		5		X 6	X 21	22	261	262	124	6510	125	4374
>	■ INTO[31:0]	9555008	х		5		7	Х 38	40	520	522	6422	12988	263	9555008

```
| Name |
```

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.

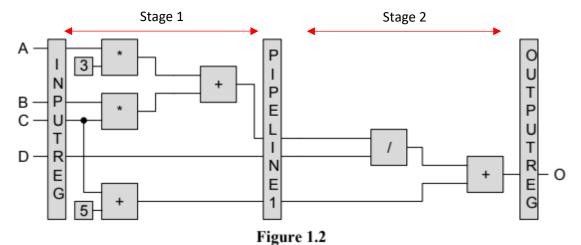
Max Delay Paths

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}) 0 reg[291/D Destination: (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:



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

Name	Value		500 ns			1,000 ns			1,500 n				2,000	ns			2,500	ns			3,000 n	s
Inputs																						
¹a dk	1																					
¼ rst	0																					
> 🔏 A[15:0]	4369	υ			0		X 1 X 1	6 ) 1	17 X 2	6 X 25	7 X 10	5 (64	69 X 34	15 X				43	59			_
> 🛂 B[15:0]	4369	Ū			0		X 1 X 1	6 )	7 ( 21	6 ( 25	7 (10	56 ( 64	57 X 64	11 X				43	59			_
> 챜 C[15:0]	4369	Ū	$\square$		0		X 1 X 1	6 X 1	L7 X 21	6 X 25	7 X 11	.9 × 65	05 X 12	:0 X				431	59			
> 🖥 D[15:0]	2	υ	$\sim$		1		X 16 X 1	7 X 1	L8 X 2	6 X 25	57 X 21	D X 64	86 X 56	4 X				2				
Outputs																						
> 🛂 O[31:0]	9555008		х		$\overline{}$	0	X	5	X 6	38	40	520	522	6422	12988	263 X			95	55008		_
Internals																						
> NSTAGE1_INTA[15:0]	4369	U		X	0		(1	16	17	256	257	105	6469	345					1369			
> NSTAGE1_INTB[15:0]	4369	U		X	0		X 1	16	17	256	257	1056	6457	641					1369			
> NSTAGE1_INTC[15:0]	4369	υ		X	0		X 1	16	X 17	256	257	119	6505	120					1369			
> NSTAGE1_INTD[15:0]	2	U		X	1		16	17	18	256	257	20	6486	564					2			
> 3 STAGE1_INT1[17:0]	13107	х		X	0		Х з	48	51	768	771	315	19407	1035				1	3107			_
> 3 STAGE1_INT2[31:0]	19088161	X		X	0		/ 1	256	289	(65536)	66049	(125664)	(42000)	76920				19	088161			
> 3 STAGE1_INT3[31:0]	19101268	×		X	0		X 4	304	340	66304	66820	(125979)	42020	77955				19.	101268			
> N STAGE1_INT5[31:0]	4374	×		X	5		X 6	21	22	261	262	124	6510	125					1374			_
> NSTAGE2_INTD[15:0]	2		U.	X		1		16	17	18	256	257	(20)	6486	564				2			
> 3 STAGE2_INT3[31:0]	19101268		х	X		0		X 4	304	340	66304	66820	(25979)	42020	77955				1910126	58		_
> 3 STAGE2_INT4[31:0]	9550634		×	X		0			17	18	259	260	6298	6478	138				955063	4		
> = STAGE2_INT5[31:0]	4374		×	X	5 X	0 X	5	6	21	( 22 )	261	262	124	6510	125				4374			
> N STAGE2_INTO[31:0]	9555008		х	X	5 )	0	5	( 6	38	40	520	522	6422	12988	263				955500	8		_

### Screenshot of the console output

```
| Land, Standards
| Landards
```

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)
                        STAGE2_INTD_reg[5]/C
 Source:
                          (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
                       Setup (Max at Slow Process Corner)
 Path Type:
 Requirement:
                        94.000ns (clk rise@94.000ns - clk rise@0.000ns)
 Data Path Delay:
                        93.776ns (logic 43.145ns (46.008%) route 50.63lns (53.992%))
                       186 (CARRY4=155 LUT1=1 LUT2=1 LUT3=29)
 Logic Levels:
```

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?

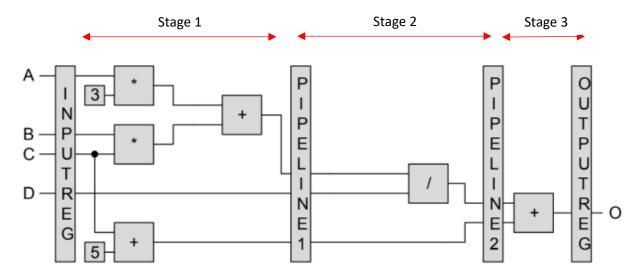
The clock period should reduce because the signal only needs to cross between the input and output in two clock cycles, rather than one. In other words in a clock cycle the signal only needs to go across one stage so the circuit should be faster.

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

																				_					
Name	е	Value	.  5	00 ns		, 1,	000 ns			1,500 r	ıs			2,000	ns			2,500	ns		3,000	ns			3,500 r
Inpu	uts																								
⅓ d	dk	0																							
₩ rs	st	0																							
> 🔏 A	(15:0]	4369	U	$\overline{}$		0			16 X 1	7 X 2!	56 X 25	57 ) 1	5 (64	69 X 3∙	45							4	369		
> 🦥 B	3[15:0]	4369	U	$\overline{}$		0			16 X 1	7 X 2!	6 25	57 (10	56 ( 64	57 (6	41 X							4	369		
> 🖥 C	0[15:0]	4369	U	$\overline{}$		0			16 X 1	7 X 2!	6 25	57 (1	.9 (65	05 ( 1:	20 X							4	369		
> 🔏 D	0[15:0]	2	U	$\overline{}$		1		16 X	17 X 1	8 X 2!	56 X 25	57 🔾 2	0 (64:	85 🛚 5	54 X								2		
Outp	puts																								
> 🖥 0	0[31:0]	9555008		Х		X	0		χ:		(6)	(38	40	520	522	6422	(12988)	263 X						955501	08
Inter	mals																								
₩ d	dk	0																							
V≟ rs	st	0																							
> 🔧 A	(15:0]	4369	U			0		1	16 / 1	7 X 28	56 X 25	57 ) 1	5 (64)	69 X 3	15							4	369		
> 📑 B	3[15:0]	4369	U			0		1	16 / 1	7 X 28	56 X 25	57 (10	56 X 64	57 X 6	41 X							4	369		
> 🗃 C	C[15:0]	4369	υ	$\overline{}$		0		1	16 / 1	7 X 28	56 X 25	57 ( 1.	.9 (65	05 X 1:	20 X							4	369		
> 🥞 D	0[15:0]	2	υ	$\overline{}$		1		16	17 / 1	8 X 25	56 X 25	57 X 2	0 (64	85 🔾 50	54								2		
> 🔏 0	0[31:0]	9555008		Х		X	0		χ ,		(6)	38	40	520	522	6422	12988	263 X						95550	08
> 🥞 S	STAGE1_INTA[15:0]	4369	υ	X		0		1	X 16	17	256	257	105	6469	345								4369		
> 💐 S	STAGE1_INTB[15:0]	4369	U	X		0		1	16	17	256	257	1056	6457	641	$\overline{}$							4369		
> 🔫 S	STAGE1_INTC[15:0]	4369	U	X		0		1	16	17	256	257	119	6505	120								4369		
> 🔫 S	STAGE1_INTD[15:0]	2	U	X		1		16	X 17	18	256	257	20	6486	564	$\overline{}$							2		
> 🔫 S	STAGE1_INT1[17:0]	13107	×	X		0		Х 3	X 48	51	768	771	315	19407	1035	$\overline{}$							13107		
> 🥞 S	STAGE1_INT2[31:0]	19088161	×	X		0		1	256	(289	(65536)	66049	1250	4200	76920							1	9088161		
> 🔫 S	STAGE1_INT3[31:0]	19101268	×	X		0		X 4	304	340	(66304)	66820	1250	4200	(77955)							1	9101268		
> 💐 S	STAGE1_INT5[31:0]	4374	х	X		5		χ 6	21	22	261	262	124	6510	125								4374		
> 💐 S	STAGE2_INTD[15:0]	2		υ	X		1		16	17	18	256	257	20	6486	564	$\overline{}$						2		
> 💐 S	STAGE2_INT3[31:0]	19101268		х	X		0		X 4	304	340	66304	66820	1250	4200	77955							19101:	268	
> 🥞 S	STAGE2_INT4[31:0]	9550634		х			0			(17)	18	259	260	6298	6478	138							95506	34	
> 🔫 S	STAGE2_INT5[31:0]	4374		х	5	χ ο	$\overline{X}$	5	X 6	21	(22)	261	262	124	(6510)	125	$\equiv$						437	1	
> 🔫 S	STAGE3_INT4[31:0]	9550634		Х		X		0			17	18	259	260	6298	6478	138						9.	550634	
> 🔫 S	STAGE3_INT5[31:0]	4374		Х		X	0	$\propto \overline{}$	5	( 6 )	21	22	261	262	124	6510	125							4374	
> 🥞 🛭	STAGE3_INTO[31:0]	9555008		х		X	0	$\supset$	5	( 6 )	38	40	520	522	6422	12988	263						9.	555008	

### Screenshot of the console output

```
| Jaunch_assimilation | Jaunch_assim_lattion | Jaunch_assim_lattion in 'D:/files/uni/digitalengineering/lab3/2-1-7_2-1-9/Performance_Check_On_Algorithm_Circuit/Performance_Check_On_Algorithm_Circuit.sim/sim_l/behav/xsim' | Jaunch_assim_lattion_object_is 'sim_l' | Jaunch_assim_l' | Jaunc
                                          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 ?
    # run 1000ns
IMFO: [USF-XSim-96] XSim completed. Design snapshot 'two_process_TB_behav' loaded.
IMFO: [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
```

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  $(\frac{1}{2000})$ .

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:
 Path Type:
                        Setup (Max at Slow Process Corner)
 Requirement:
                       90.000ns (clk rise@90.000ns - clk rise@0.000ns)
                       89.478ns (logic 42.152ns (47.109%) route 47.326ns (52.891%))
 Data Path Delav:
 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?

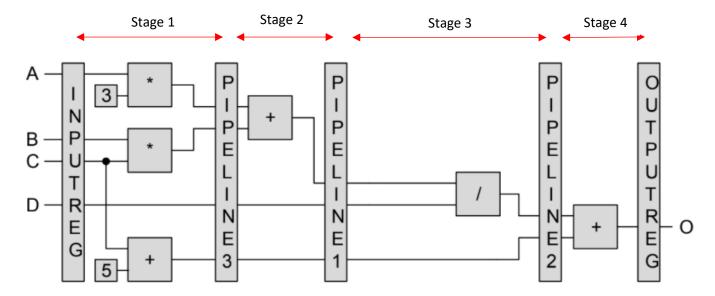
The clock period should reduce since the new pipeline we added was between the two operators in the previous critical path, meaning that that path can now be taken in two clock cycles rather than 1.

# 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

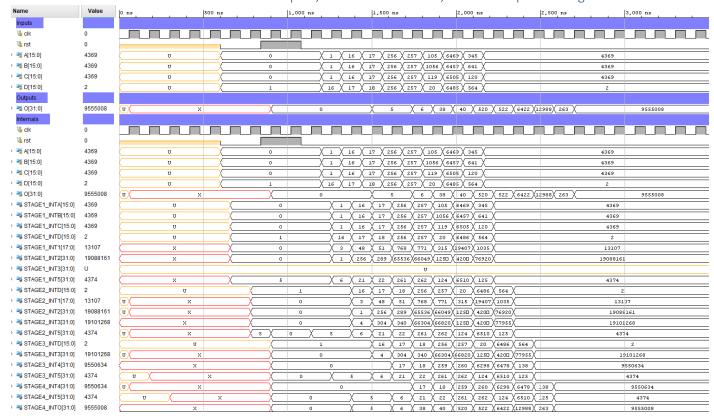
## A note on variable names

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

```
INFO: [Visuation object is 'sim_l' INFO: [SIM-utils-51] Simulation object is 'sim_l' INFO: [SIM-utils-51] Simulation design source files for 'two_process_TB' in fileset 'sim_l'...
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] vcmpile' step finished in '2' seconds

INFO: [USF-XSim-6] 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-64] 'Sim::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
 # 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
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
```

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

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

1	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)
                       STAGE3_INTD_reg[5]/C
 Source:
                         (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
                      Setup (Max at Slow Process Corner)
 Path Type:
 Requirement:
                       94.000ns (clk rise@94.000ns - clk rise@0.000ns)
                       93.229ns (logic 42.318ns (45.392%) route 50.911ns (54.608%))
 Data Path Delay:
                       185 (CARRY4=154 LUT1=1 LUT3=30)
  Logic Levels:
```

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

Name	Value	0 ns	500 ns	1,000 ns	1,500 ns	2,000 ns	2,500 ns	3,000 ns
Inputs								
¼ clk	0							
¼ rst	0							
™ A[15:0]	4369	U	X	0	16   17   256   257   105	X 6469 X 345 X		4369
₩ B[15:0]	4369	U				6 X 6457 X 641 X		4369
™ C[15:0]	4369	U	- V		16 × 17 × 256 × 257 × 119			4369
₫ D[15:0]	2	Ū	$\rightarrow$		17 \ 18 \ 256 \ 257 \ 20			2
Outputs					/ / / /	X 222 X 221 X		
₫ O[31:0]	9555008	U	x			0		
nternals	5555666	\ <u>\</u>	^					
STAGE1_INTA[15:0]	4369	U		0	16 17 256 257	105 V C4C0 V C45 V		4369
STAGE1_INTB[15:0]	4369		$\longrightarrow$					
		U	+-	0 1		1056 X 6457 X 641 X		4369
STAGE1_INTC[15:0]	4369	U	<u> </u>	0 1		119 (6505 ) 120 )		4369
STAGE1_INTD[15:0]		ū	<u> </u>	1 16	17 18 256 257	20 X 6486 X 564 X		2
STAGE1_INT1[17:0]	13107	х	X	0 X 3		315 (19407) 1035 (		13107
STAGE1_INT2[31:0]	19088161	х	X	0 1	X 256 X 289 X65536X66049X			19088161
STAGE1_INT3[31:0]	19101268	Х	X	0 4	X 304 X 340 X66304 X66820 X			19101268
STAGE1_INT5[31:0]	4374	х	X	5 X 6		124 X 6510 X 125 X		4374
STAGE2_INTD[15:0]		Ū	Х	1		257 X 20 X 6486 X 564 X		2
STAGE2_INT3[31:0]	19101268	n x	X	0	4 304 340 66304	66820 1250 4200 77955		9101268
STAGE2_INT5[31:0]	4374	n x	X	5 0 5	6 21 22 261	262   124   6510   125		4374
STAGE3_INT4[31:0]	9550634					0		
STAGE3_INT5[31:0]	4374	U		X		0		
STAGE4_INT4[31:0]	9550634	U X				0		
	9550634 4374	n (	•	X		0 0		
STAGE4_INT5[31:0] STAGE4_INTO[31:0]		K K		5,000 ns	[5,500 ns	0	500 ns	7,000 ns
STAGE4_INT5[31:0] STAGE4_INTO[31:0] ame	4374 9555008	K K		S,000 ns	5,500 ns	0	500 ns	7,000 ns
STAGE4_INT5[31:0] STAGE4_INTO[31:0]  mme  nputs	4374 9555008 <b>Value</b>	K K		[5,000 ns	5,500 ns	0	500 ns	7,000 ns
STAGE4_INT5[31:0] STAGE4_INT0[31:0]  mme inputs clk	4374 9555008 <b>Value</b> 0 0	K K		[5,000 ns	5,500 ns	0	500 ns	7,000 ns
STAGE4_INT5[31:0] STAGE4_INTO[31:0]  me  mputs Lick state first 14 (15:0)	4374 9555008 Value 0 0 4369	K K		5,000 ns	5,500 ns	0	500 ns	7,000 ns
© STAGE4_INT5(31:0) © STAGE4_INTO(31:0)  mane  nputs 0	4374 9555008 Value 0 0 4369 4369	K K		5,000 ns		0	500 ns	7,000 ns
\$ STAGE4_INT5(31:0) \$ STAGE4_INTO(31:0)  ame  nputs  the circle for st  4 (415:0)  \$ 19(15:0)	4374 9555008 Value 0 0 4369	K K		5,000 ns	4369	0	500 ns	7,000 ns
\$ STAGE4_INT5(31:0) \$ STAGE4_INTO(31:0)  \$ STAGE4_INTO(31:0)  \$ sme  \$ clk \$ clk \$ rst \$ 4(15:0) \$ gl(15:0) \$ (15:0)	4374 9555008 Value 0 0 4369 4369	K K		5,000 ns	4369 4369	0	500 ns	7,000 ns
\$\$ STAGE4_INT5(31:0] \$\$ STAGE4_INTO(31:0] \$\$ STAGE4_INTO(31:0] \$\$ since the stage of the stage o	4374 9555008 Value 0 0 4369 4369 4369 2	K K	4,500 ns		4369 4369 4369 2	0 0		
\$ STAGE4_INT5(31:0] \$ STAGE4_INTO(31:0]  \$ STAGE4_INTO(31:0]  \$ STAGE4_INTO(31:0]  \$ GIX \$	4374 9555008 Value 0 0 4369 4369 4369	K K		5,000 ns	4369 4369 4369 2	0		7,000 ns
\$ STAGE4_INT5(31:0) \$ STAGE4_INTO(31:0) \$ STAGE4_INTO(31:0) \$ STAGE4_INTO(31:0) \$ Clk \$ Interpretation of the content of the c	4374 9555008 Value 0 0 4369 4369 2 9555008	K K	4,500 ns	[5,000 ns	4369 4369 4369 2	0 0		
## STAGE4_INT5(31:0] ## STAGE4_INTO(31:0] ## STAGE4_INTO(31:0] ## Interpretation	4374 9555008 Value 0 0 4369 4369 4369 2 9555008	K K	4,500 ns	5,000 ns	4369 4369 4369 2	0 0		
## STAGE4_INT5(31:0] ## STAGE4_INTO(31:0] ### ### #############################	4374 9555008 Value 0 0 4369 4369 2 9555008 4369 4369	K K	4,500 ns	[5,000 ns	4369 4369 2 X S X 6 X 38 X 4369 4369	0 0		
\$ STAGE4_INT5(31:0] \$ STAGE4_INTO(31:0]  sime  simputs  dict dirst  simputs dict dirst dist(15:0) d	4374 9555008 Value 0 0 4369 4369 4369 2 9555008	K K	4,500 ns		4369 4369 4369 2 X 5 X 6 X 38 X	0 0		
### STAGE4_INT5(31:0] #### ### ### ########################	4374 9555008 Value 0 0 4369 4369 4369 2 9555008 4369 4369 4369 2	K K	4,500 ns	[5,000 ns	14369 4369 4369 2 X 5 X 6 X 38 X 4369 4369 4369 2	0 0		
# STAGE4_INT5(31:0) # STAGE4_INTO(31:0) # STAGE4_INTO(31:0) # Disputs # STAGE1_INTA(15:0) # STAGE1_INTA(15:0) # STAGE1_INTD(15:0)	Value  Value  0 0 4369 4369 4369 2 9555008  4369 4369 2 13107	K K	4,500 ns	5,000 ns	4369 4369 4369 2 2 3 6 38 4369 4369 4369 4369 2 13107	0 0 0   6,000 ns   6,		
### STAGE4_INT5(31:0] #### INTO(31:0] ##### INTO(31:0] ####################################	4374 9555008 Value 0 0 0 4369 4369 2 9555008  4369 4369 4369 4369 2 13107 19088161	K K	4,500 ns	[5,000 ns	14369 4369 4369 2 X 5 X 6 X 38 X 4369 4369 4369 2	0 0 0   6,000 ns   6,		
### STAGE4_INT5(31:0] ### INTO(31:0] ### INTO(31:0] ### INTO(31:0] ### INTO(31:0] ### INTO(31:0) ### INTO(31:0) ### INTO(31:0) ### INTO(31:0) ### STAGE1_INT4(15:0) ### STAGE1_INT5(15:0) ### STAGE1_INT1(15:0) ### STAGE1_INT1(15:0) ### STAGE1_INT1(17:0) ### STAGE1_INT1(17:0) ### STAGE1_INT1(17:0) ### STAGE1_INT1(17:0) #### STAGE1_INT2(31:0) #### STAGE1_INT3(31:0)	4374 9555008 Value 0 0 4369 4369 4369 2 9555008 4369 4369 4369 4369 13107 19088161 19101268	K K	4,500 ns	5,000, ns	4369 4369 4369 2 2 3 6 38 4369 4369 4369 4369 2 13107	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0		
### STAGE4_INT5(31:0] ### STAGE4_INTO(31:0] ### INTO(31:0] ### INTO(31:0] ### INTO(31:0] ### INTO(31:0] ### INTO(31:0) ### INTO(31:0) ### INTO(31:0) ### INTO(31:0) ### STAGE1_INT4(15:0) ### STAGE1_INTD(15:0) ### STAGE1_INTD(15:0) ### STAGE1_INTD(15:0) ### STAGE1_INTD(15:0) ### STAGE1_INTD(15:0) ### STAGE1_INTT(17:0) ### STAGE1_INT2(31:0) ### STAGE1_INT3(31:0) ### STAGE1_INT3(31:0) #### STAGE1_INT3(31:0)	4374 9555008 Value 0 0 4369 4369 4369 2 9555008 4369 4369 4369 2 13107 19088161 19101268 4374	K K	4,500 ns		4369 4369 2 2 3 6 39 4369 4369 4369 4369 2 12107 19088161	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0		
### STAGE4_INT5(31:0) ### STAGE4_INTO(31:0) ### INTO(31:0) ### INT	4374 9555008 Value 0 0 4369 4369 4369 2 9555008 4369 2 13107 19088161 19101268 4374 2	K K	4,500 ns	[5,000 ns ]	4369 4369 2 2 3 5 6 30 X 4369 4369 4369 2 13107 19088161	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0		
STAGE4_INT5(31:0] STAGE4_INTO(31:0] STAGE4_INTO(31:0] STAGE4_INTO(31:0] STAGE1_INT5(31:0) STAGE1_INT5(3:0)	Value  Value  0 0 4369 4369 4369 2 9555008  4369 2 13107 19088161 19101268 4374 2 19101268	K K	4,500 ns	[5,000 ns	4369 4369 4369 2 2 3 5 6 38 X 4369 4369 4369 2 13107 1908161 19101264 4374	0 0 0 0   6,000 ns   6,		
STAGE4_INT5[31:0]  STAGE4_INTO[31:0]  ame  Imputs  Inputs  Inp	4374 9555008 Value 0 0 4369 4369 4369 2 9555008 4369 2 13107 19088161 19101268 4374 2	K K	4,500 ns	5,000 ns	4369 4369 2 2 3 5 6 38 2 4369 4369 4369 2 2 13107 19088161 19101266 4374	0 0 0 0   6,000 ns   6,		
STAGE4_INT5(31:0) STAGE4_INTO(31:0) STAGE4_INTO(31:0) STAGE4_INTO(31:0) STAGE1_INTA(15:0) STAGE2_INTA(15:0) STAGE2_INTA(15:0)	Value  Value  0 0 4369 4369 4369 2 9555008  4369 2 13107 19088161 19101268 4374 2 19101268	K K	4,500 ns		4369 4369 2 3 4369 4369 2 4369 4369 4369 4369 4369 4374 4274 4374 4374	0 0 0 0   6,000 ns   6,		
STAGE4_INT5(31.0) STAGE4_INTO(31.0) STAGE4_INTO(31.0)  Imputs Continue Stage Stage1_INT6(15.0) Continue Stage1_INT6(15.0) STAGE2_INT6(15.0) STAGE2_INT6(15.0) STAGE2_INT6(15.0) STAGE2_INT6(15.0) STAGE2_INT6(15.0) STAGE2_INT6(15.0) STAGE2_INT6(15.0) STAGE2_INT6(15.0) STAGE2_INT6(15.0)	Value  0 0 4369 4369 4369 2 9555008  4369 4369 2 13107 19088161 19101268 4374 2 19101268 4374 4374	K K	4,500 ns		4369 4369 2 3 4369 4369 2 4369 4369 4369 4369 4369 4374 4274 4374 4374	0 0 0 (6,000 ns		9555008
STAGE4_INT5(31.0)  STAGE4_INTO(31.0)  STAGE4_INTO(31.0)  STAGE4_INTO(31.0)  STAGE1_INTO(31.0)  STAGE1_INTA(15.0)  STAGE1_INTA(15.0)  STAGE1_INTD(15.0)  STAGE1_INTD(15.0)  STAGE1_INTD(15.0)  STAGE1_INT1(17.0)  STAGE1_INT3(31.0)  STAGE1_INT5(31.0)  STAGE2_INT5(31.0)  STAGE2_INT5(31.0)  STAGE2_INT5(31.0)  STAGE2_INT5(31.0)  STAGE2_INT5(31.0)  STAGE2_INT3(31.0)  STAGE2_INT3(31.0)  STAGE2_INT3(31.0)  STAGE3_INT4(31.0)	4374 9555008 Value 0 0 4369 4369 4369 2 9555008  4369 2 13107 19088161 19101268 4374 2 19101268 4374 9550634	K K	0 0		4369 4369 2 2 3 5 6 38 X 4369 4369 4369 4369 4369 4369 2 13107 1908161 19101261 4374 2 19101264 4374 4374 4374 4374 4374 4374 4374 43	0 0 0 (6,000 ns		9555008 9555008
** STAGE4_INT4(31:0)  ** STAGE4_INT6(31:0)  ** STAGE4_INT6(31:0)  ** STAGE4_INT6(31:0)  ** STAGE4_INT6(31:0)  ** STAGE4_INT6(31:0)  ** STAGE4_INT6(31:0)  ** STAGE1_INT6(15:0)  ** STAGE1_INT6(31:0)  ** STAGE2_INT6(31:0)  ** STAGE2_INT6(31:0)  ** STAGE3_INT6(31:0)  ** STAGE3_INT4(31:0)  ** STAGE3_INT4(31:0)  ** STAGE3_INT4(31:0)  ** STAGE3_INT4(31:0)  ** STAGE3_INT4(31:0)  ** STAGE3_INT6(31:0)  ** STAGE3_INT6(31:0)  ** STAGE3_INT6(31:0)  ** STAGE3_INT6(31:0)  ** STAGE3_INT6(31:0)  ** STAGE3_INT6(31:0)	Value  0 0 4369 4369 4369 2 9555008  4369 4369 2 13107 19088161 19101268 4374 2 19101268 4374 4374	K K	0 0 0		4369 4369 2 3 5 6 38 4369 4369 4369 4369 4369 2 13107 13088161 4374 2 2 19101266 4374 2 4374 2 17 18 259 32 6 21 22 261 32	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0		9555008 9555008 9550634 4374



```
in 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..
         [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 comm
                                    onent 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 lps
   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 lps
   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 luts_v of entity unisim.LUT3 [\LUT3(init="01101010")(0,7)\]
Compiling architecture luts_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 lutl_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 th
   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_l: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

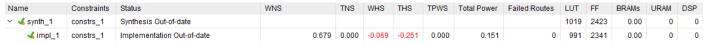
  source two process TB.tcl
    set curr_wave [current_wave_config]
  # if { [string length $curr_wave] == 0
# if { [llength [get_objects]] > 0} {
        set_property needs_save false [current_wave_config]
     } else {
        send_msg_id Add_Wave-1 WARNING "No top level signals found. Simulator will 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
```

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".



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) STAGE1\_INTB\_reg[7]/C Source: (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) 14.000ns (clk rise@14.000ns - clk rise@0.000ns) Requirement: Data Path Delay: 13.337ns (logic 5.20lns (38.995%) route 8.136ns (61.005%)) 13 (CARRY4=7 LUT3=1 LUT4=1 LUT6=4) Logic Levels:

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.

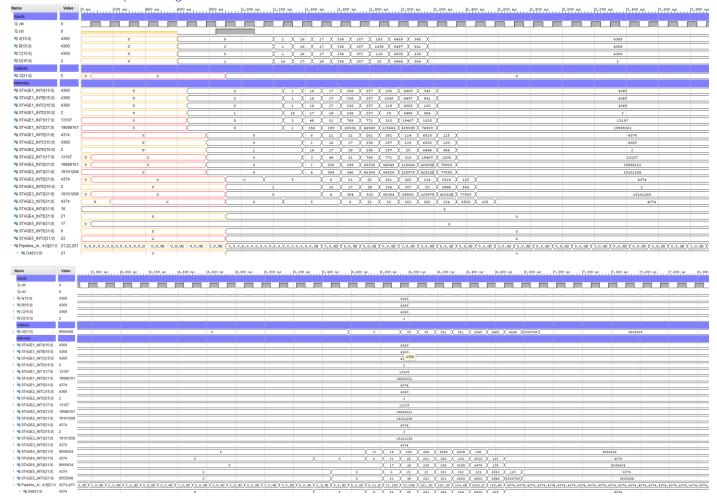
Does the practice match the theory (compare the new clock period to the one you obtained in step 2.1.8)?

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 Algorithm
      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 Algori
      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_Circuit/Performance_Check_On_Algorithm_Circuit/Performance_Check_On_Algorithm_Circuit/Performance_Check_On_Algorithm_Circuit/Performance_Check_On_Algorithm_Circuit/Performance_Check_On_Algorithm_Circuit/Performance_Check_On_Algorithm_Circuit/Performance_Check_On_Algorithm_Circuit/Performance_Check_On_Algorithm_Circuit/Performance_Check_On_Algorithm_Circuit/Performance_Check_On_Algorithm_Circuit/Performance_Check_On_Algorithm_Circuit/Performance_Check_On_Algorithm_Circuit/Performance_Check_On_Algorithm_Circuit/Performance_Check_On_Algorithm_Circuit/Performance_Check_On_Algorithm_Circuit/Performance_Check_On_Algorithm_Circuit/Performance_Check_On_Algorithm_Circuit/Performance_Check_On_Algorithm_Circuit/Performance_Check_On_Algorithm_Circuit/Performance_Check_On_Algorithm_Circuit/Performance_Check_On_Algorithm_Circuit/Performance_Check_On_Algorithm_Circuit/Performance_Check_On_Algorithm_Circuit/Performance_Check_On_Algorithm_Circuit/Performance_Check_On_Algorithm_Circuit/Performance_Check_On_Algorithm_Circuit/Performance_Check_On_Algorithm_Circuit/Performance_Check_On_Algorithm_Circuit/Performance_Check_On_Algorithm_Circuit/Performance_Check_On_Algorithm_Circuit/Performance_Check_On_Algorithm_Circuit/Performance_Check_On_Algorithm_Circuit/Performance_Check_On_Algorithm_Circuit/Performance_Check_On_Algorithm_Circuit/Performance_Check_On_Algorithm_Circuit/Performance_Check_On_Algorithm_Circuit/Performance_Check_On_Algorithm_Circuit/Performance_Check_On_Algorithm_Circuit/Performance_Check_On_Algorithm_Circuit/Performance_Check_On_Algorithm_Circuit/Performance_Check_On_Algorithm_Circuit/Performance_Check_On_Algorithm_Circuit/Performance_Check_On_Algorithm_Circuit/Performance_Check_On_Algorithm_Circuit/Performance_Check_On_Algorithm_Circuit/Performance_Check_On_Algorithm_Circuit/Performance_Check_On_Algorithm_Circuit/Performance_Check_On_Algorithm_Circuit/Performance_Check_On_Algorit
      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
```

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 DSP48El clocked by clk {ri
  Destination:
                         STAGE3 INT3 reg/PCIN[0]
                           (rising edge-triggered cell DSP48El clocked by clk {ri
  Path Group:
                         clk
  Path Type:
                       Setup (Max at Slow Process Corner)
                        4.000ns (clk rise@4.000ns - clk rise@0.000ns)
  Requirement:
                       2.413ns (logic 2.411ns (99.917%) route 0.002ns (0.083%)
  Data Path Delay:
  Logic Levels:
```

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 {\tt 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);</pre>
      STAGE1_INTC <= unsigned(C);</pre>
      STAGE1_INTD <= unsigned(D);</pre>
    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);</pre>
 -- Pipeline O 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);</pre>
      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;</pre>
      STAGE2_INT2 <= STAGE1_INT2;</pre>
      STAGE2 INT5 <= STAGE1 INT5;
    end if:
  end if;
end process pipeline 0 regs;
 - Stage2
STAGE2 INT3 <= STAGE2 INT1 + Stage2 INT2;</pre>
-- 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);</pre>
STAGE4 INT5 <= unsigned(Pipeline Array Internal(LATENCY));</pre>
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');
                Pipeline_Array_Internal(I+1) <= Pipeline_Array_Internal(I);</pre>
            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');
      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
      0 <= (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 0 : 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, \overline{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 \Rightarrow A
         B => B,
         C \Rightarrow C
         D \Rightarrow D
         0 => 0
     );
     -- 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;</pre>
              C <= test_vectors(i).C;</pre>
              D <= test_vectors(i).D;</pre>
              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 (0 = test_vectors(i).0)
    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
```