FPGA Development for Radar, Radio-Astronomy and Communications

THE RADAR MASTERS COURSE

Dept. Electrical Engineering, University of Cape Town
Private Bag, Rondebosch, 7701, South Africa

http://www.rrsg.uct.ac.za

Presented by Dr John-Philip Taylor

Convened by Dr Stephen Paine

Day 2 – 10 September 2024

Outline

Finite State Machines

Simulation

JTAG

Timing Constraints

Advanced Timing Constraints

Outline

Finite State Machines

Simulation

JTAG

Timing Constraints

Advanced Timing Constraints

Finite State Machines

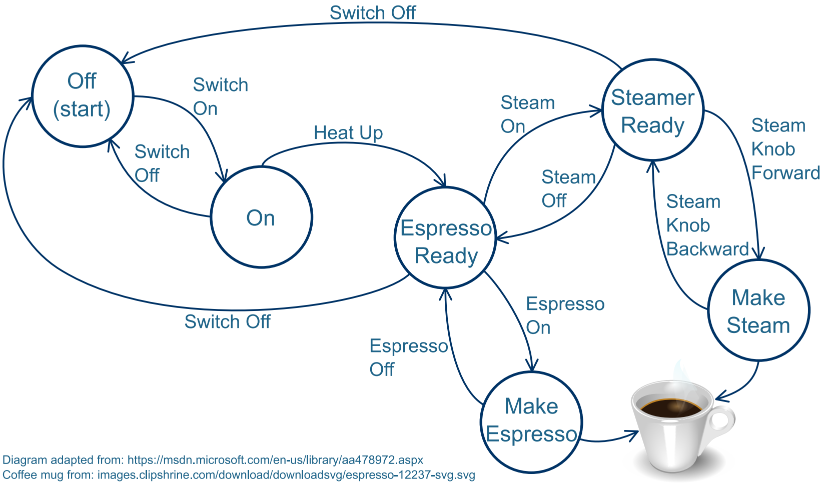


Diagram adapted from: https://msdn.microsoft.com/en-us/library/aa478972.aspx
Coffee mug from: images.clipshrine.com/download/downloadsvg/espresso-12237-svg.svg

Named States

- Example using Gray code:

```
reg [2:0]State;

localparam Off           = 3'b000;
localparam On            = 3'b001;
localparam EspressoReady = 3'b011;
localparam SteamerReady  = 3'b010;
localparam MakeEspresso  = 3'b110;
localparam MakeSteam     = 3'b111;
```

Named States

- Example using one-hot encoding:

```
reg [5:0]State;

localparam Off           = 6'b000001;
localparam On            = 6'b000010;
localparam EspressoReady = 6'b000100;
localparam SteamerReady  = 6'b001000;
localparam MakeEspresso  = 6'b010000;
localparam MakeSteam     = 6'b100000;
```

Named States

- Or let the compiler select the encoding:

```
typedef enum{ // SystemVerilog only
  Off,
  On,
  EspressoReady,
  SteamerReady,
  MakeEspresso,
  MakeSteam
} STATE;
STATE State;
```

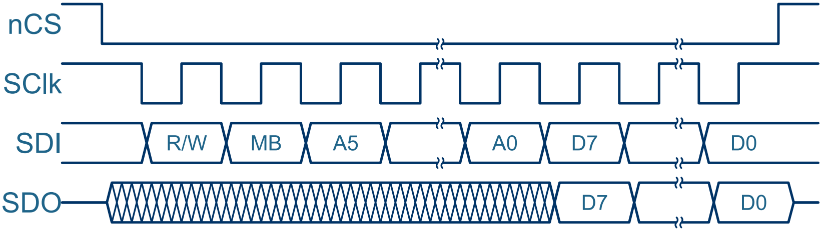
FSM Template

```
always @(posedge ipClk) begin
  if(Reset) begin
    State <= Off;

  end else begin
    case(State)
      Off:           begin ... end
      On:            begin ... end
      EspressoReady: begin ... end
      SteamerReady:  begin ... end
      MakeEspresso:  begin ... end
      MakeSteam:     begin ... end
      default:;
    endcase
  end
end
```

FSM Example

```
              ...
case(State)
  Off: begin
    if(Switch_On) State <= On;
  end

  On: begin
         if(Switch_Off) State <= Off;
    else if(Heat_Up   ) State <= EspressoReady;
  end

  EspressoReady: begin
         if(Switch_Off ) State <= Off;
    else if(Steam_On    ) State <= SteamerReady;
    else if(Espresso_On) State <= MakeEspresso;
  end
              ...
```

SPI Interface



- For the ADXL345 Digital Accelerometer:
  - Maximum SClk frequency is 5 MHz (200 ns period)
  - SDI setup and hold is 5 ns (sampled on rising edge)
  - SClk falling edge to SDO delay is 40 ns

The Abstraction

```
module ADXL345 #(
  parameter Clock_kHz,
  parameter Baud_kHz = 5_000
)(
  input ipClk, ipReset,

  // 2's Compliment Output
  output reg [15:0]opX,
  output reg [15:0]opY,
  output reg [15:0]opZ,

  // Physical device interface
  output reg opnCS, opSClk, opSDI,
  input      ipSDO
);
localparam ClockDiv = Clock_kHz / Baud_kHz / 2;
```

General Structure

```verilog
reg       Reset;
reg [3:0]ClockCount  = 0;
wire      ClockEnable = (ClockCount == ClockDiv);

always @(posedge ipClk) begin
  Reset <= ipReset;

  if(ClockEnable) ClockCount <= 4'd1;
  else            ClockCount <= ClockCount + 1'b1;

  if(Reset) begin
    // Reset the machine here

  end else if(ClockEnable) begin
    // State machine goes here
  end
end
```

State Machine

```
reg [ 4:0]Count;
reg [15:0]Data; // (R/W, MB, Address, Byte) or (2 Bytes)

typedef enum {
  Setup,
  ReadX, ReadY, ReadZ,
  Transaction
} STATE;

STATE State;
STATE RetState; // Used for function calls
```

State Machine

```
if(Reset) begin
  opnCS   <= 1'b1;
  opSClk <= 1'b1;
  opSDI   <= 1'b1;
  State   <= Setup;

end else if(ClockEnable) begin
  case(State)
    Setup: begin
      // SPI 4-wire; Full-res; Right-justify; 4g Range
      Data     <= {2'b00, 6'h31, 8'b0000_1001};
      Count    <= 5'd16;
      State    <= Transaction;
      RetState <= ReadX;
    end
```

THE RADAR MASTERS COURSE

State Machine

```
ReadX: begin
  opZ      <= {Data[7:0], Data[15:8]};
  Data     <= {2'b11, 6'h32, 8'd0};
  Count    <= 5'd24;
  State    <= Transaction;
  RetState <= ReadY;
end

ReadY: begin
  opX      <= {Data[7:0], Data[15:8]};
  Data     <= {2'b11, 6'h34, 8'd0};
  Count    <= 5'd24;
  State    <= Transaction;
  RetState <= ReadZ;
end
```

State Machine

```
ReadZ: begin
  opY      <= {Data[7:0], Data[15:8]};
  Data     <= {2'b11, 6'h36, 8'd0};
  Count    <= 5'd24;
  State    <= Transaction;
  RetState <= ReadX;
end
```

Reading Data

```
Transaction: begin
  if(opnCS) begin
    opnCS <= 1'b0;

  end else begin
    if(opSClk) begin
      if(Count == 0) begin
        opnCS <= 1'b1;
        State <= RetState;
      end else begin
        opSClk <= 1'b0;
      end
      Count <= Count - 1'b1;
      {opSDI, Data} <= {Data, ipSDO};

    end else begin
      opSClk <= 1'b1;
end end end
```

Outline

Finite State Machines

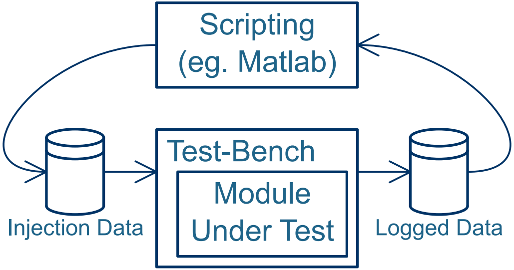
Simulation

JTAG

Timing Constraints

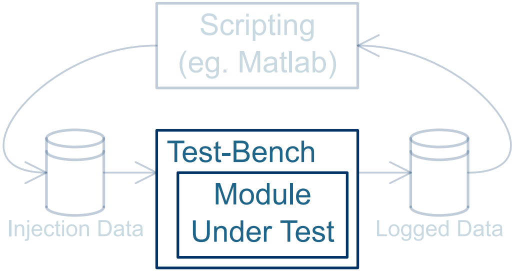
Advanced Timing Constraints

Simulation Basics



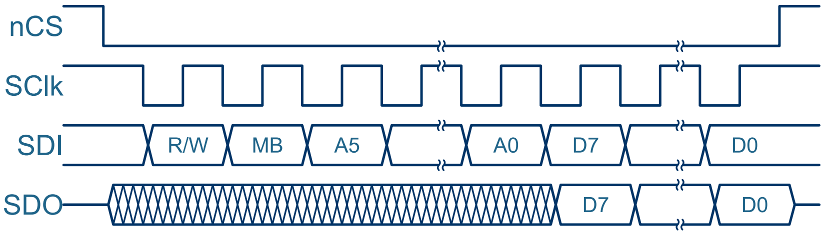
Today we only focus on the test-bench and simulation tool

Simulation Basics



Today we only focus on the test-bench and simulation tool

ADXL345 SPI Interface



Test Bench Basics

```
`timescale 1ns/1ps
module ADXL345_TB;

// Clock
reg ipClk = 0;
always #10 ipClk <= ~ipClk;

// Reset
reg ipReset = 1;
initial #50 ipReset <= 0;
```

Test Bench Basics

```
// DUT
wire [15:0]X, Y, Z;
wire nCS, SClk, SDI;
reg  SDO = 0;

ADXL345 #(50_000) Accelerometer( // Set parameter for 50 MHz clock
  ipClk, ipReset,
  opX, opY, opZ,
  opnCS, opSClk, opSDI, ipSDO
);
```

Test Bench Basics

```
reg [ 7:0]DataIn;
reg [15:0]DataOut = 0;

integer n;
always begin
  @(negedge nCS);

  // Instruction word
  for(n = 7; n >= 0; n--) begin
    @(negedge SClk);
    DataIn[n] <= SDI;
  end
```

Test Bench Basics

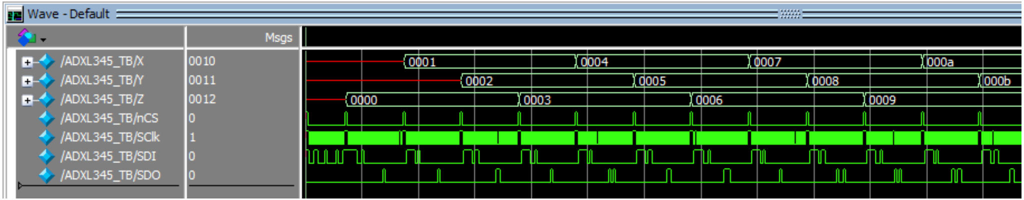
```
// The first data word
for(n = 7; n >= 0; n--) begin
  @(negedge SClk); #40 // Output delay;
  SDO <= DataOut[n];
end

// The optional second data word
if(DataIn[6]) begin // More bits
  for(n = 15; n >= 8; n--) begin
    @(negedge SClk); #40 // Output delay;
    SDO <= DataOut[n];
  end
end
```

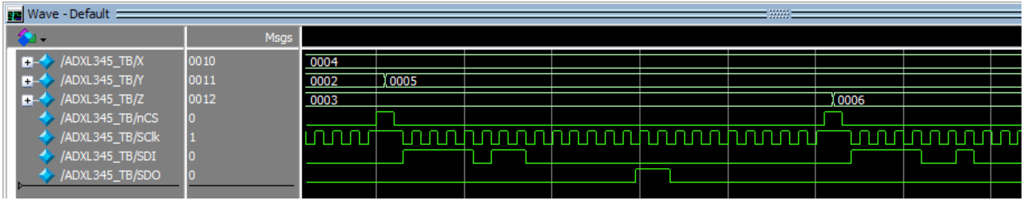
Test Bench Basics

```
    @(posedge nCS);

    DataOut <= DataOut + 1;
end

endmodule
```

Questa

The resulting Questa waveforms:



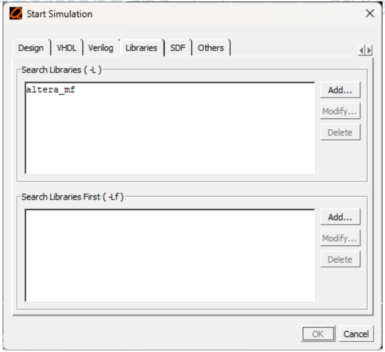


Simulation

- Questa can simulate IP modules
- Add the `altera_mf` library in the "Start Simulation" dialogue box
- Also remember to compile the IP block
- Questa cannot understand the `defparam` style of parameters.

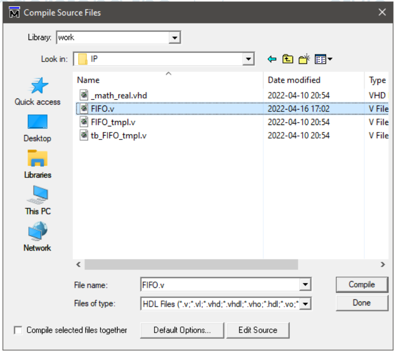
Simulation

- Questa can simulate IP modules
- Add the `altera_mf` library in the “Start Simulation” dialogue box
- Also remember to compile the IP block
- Questa cannot un… style of parameters.



Simulation

- Questa can simulate IP modules
- Add the `altera_mf` library in the “Start Simulation” dialogue box
- Also remember to compile the IP block
- Questa cannot u… of parameters.



Simulation

- Questa can simulate IP modules
- Add the `altera_mf` library in the "Start Simulation" dialogue box
- Also remember to compile the IP block
- Questa cannot understand the `defparam` style of parameters.

```
Mod Inst(
  // Port assignments
);
defparam Inst.Param1 = 8'h12;
defparam Inst.Param2 = 8'h23;
```

Simulation

- Questa can simulate IP modules
- Add the `altera_mf` library in the “Start Simulation” dialogue box
- Also remember to compile the IP block
- Questa cannot understand the `defparam` style of parameters.

```
Mod #(
  .Param1(8'h12),
  .Param2(8'h23)
) Inst(
  // Port assignments
);
```

Outline

Finite State Machines

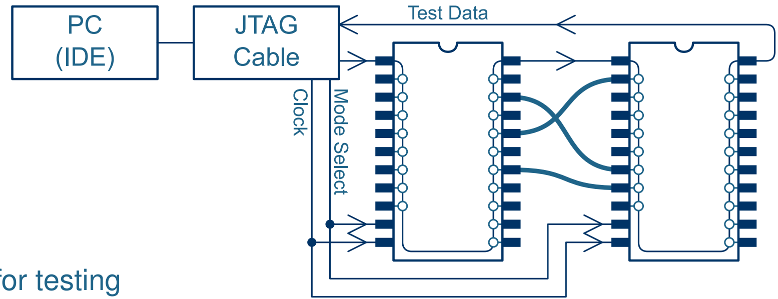
Simulation

JTAG

Timing Constraints

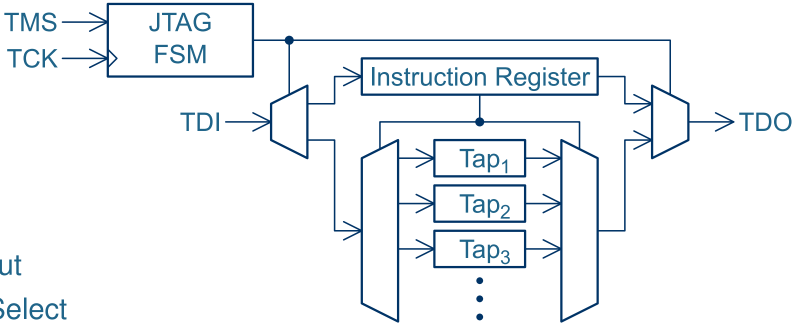
Advanced Timing Constraints

JTAG – Joint Test Action Group



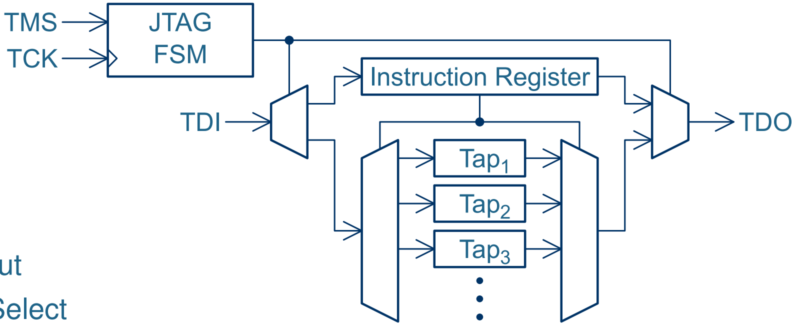
- Originally intended for testing soldering quality on populated PCBs
- Connects to the PC over USB / Ethernet / etc.
- Connected devices form a long chain of shift-registers

JTAG – Joint Test Action Group



- TCK: Test Clock
- TDI: Test Data In
- TDO: Test Data Out
- TMS: Test Mode Select
- Taps could include: Device pins; Internal flash; Status registers; Debug registers; Virtual JTAG interface; etc.

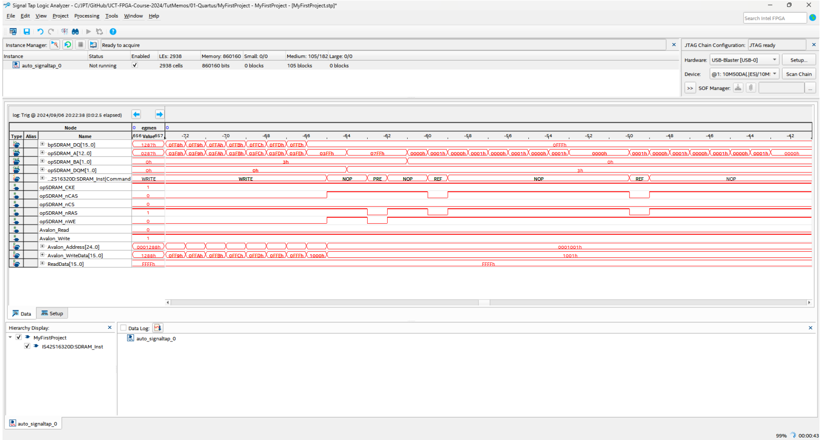
JTAG – Joint Test Action Group



- TCK: Test Clock
- TDI: Test Data In
- TDO: Test Data Out
- TMS: Test Mode Select
- Taps could include: Device pins; Internal flash; Status registers; Debug registers; Virtual JTAG interface; etc.

JTAG Debugging

FPGA IDEs includes powerful JTAG-based debugging tools.



THE RADAR MASTERS COURSE

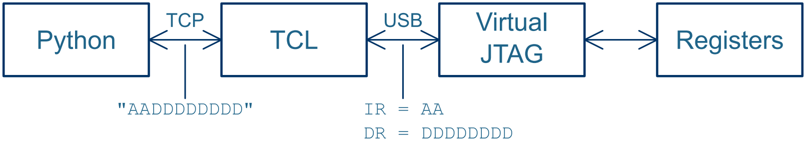
JTAG Debugging

FPGA IDEs includes powerful JTAG-based debugging tools.

- Sources and Probes
  https://www.youtube.com/watch?v=Mftgi318Nrc
- In-system memory editor
  https://www.youtube.com/watch?v=_VcVtFvJnuY
- Signal-tap logic analyser
  https://www.youtube.com/watch?v=vhkzxCEXuaA

JTAG Debugging

You can do some really interesting things using TCL scripting and the JTAG interface.



Idle Logic Labs: Talking to the DE0-Nano using the Virtual JTAG interface

Coffee Break...

Outline

Finite State Machines

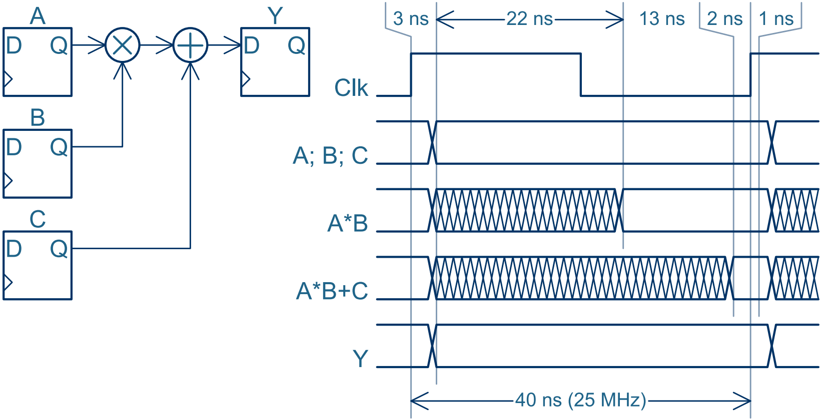
Simulation

JTAG

Timing Constraints

Advanced Timing Constraints

Internal Timing



Synopsis Design Constraints

- De facto industry standard
- TCL based, so one can use TCL scripting within the SDC file
- Only specify what the compiler does not already know:
  - Clock frequencies
  - Asynchronous paths
  - PCB trace delays
  - External device parameters
  - Multi-cycle paths
  - etc.

Synopsis Design Constraints

- De facto industry standard
- TCL based, so one can use TCL scripting within the SDC file
- Only specify what the compiler does not already know:
  - Clock frequencies
  - Asynchronous paths
  - PCB trace delays
  - External device parameters
  - Multi-cycle paths
  - etc.

Asynchronous Pins

- Pins such as LEDs, buttons, RS-232 signals, etc. does not belong to a clock domain
- Quartus must not try to meet timing on these:

```
set_false_path -from * -to   [get_ports opLED*]
set_false_path -to   * -from [get_ports ipSwitch*]
set_false_path -to   * -from [get_ports ipButton*]
```

THE RADAR MASTERS COURSE

Asynchronous Pins

- Pins such as LEDs, buttons, RS-232 signals, etc. does not belong to a clock domain
- Quartus must not try to meet timing on these:

```
set_false_path -from * -to   [get_ports opLED*]
set_false_path -to   * -from [get_ports ipSwitch*]
set_false_path -to   * -from [get_ports ipButton*]
```

Clock Specification

```
create_clock -period 100 [get_ports ADC_CLK_10]
create_clock -period  20 [get_ports MAX10_CLK1_50]
create_clock -period  20 [get_ports MAX10_CLK2_50]

derive_pll_clocks -create_base_clocks -use_net_name
derive_clock_uncertainty

create_generated_clock \
  -source [get_pins { SDRAM_PLL_Inst|*|clk[1] } ] \
  -name opClk_SDRAM [get_ports opClk_SDRAM]
```

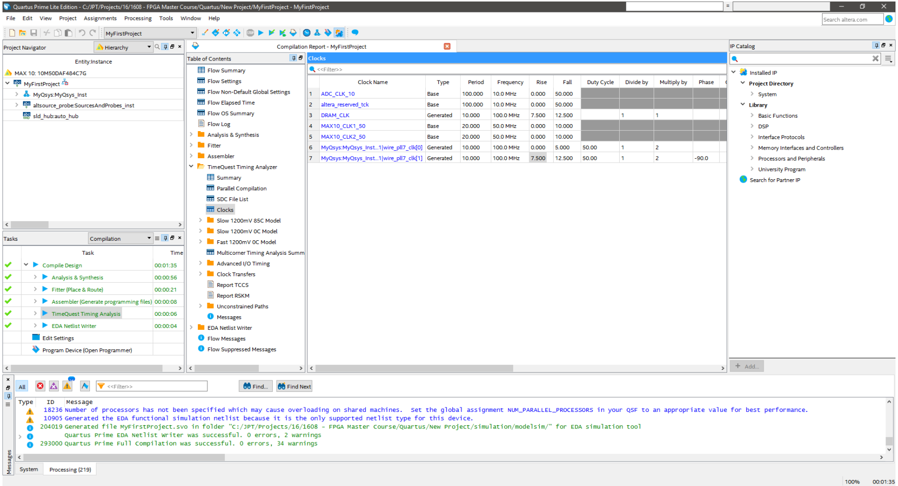
To get the node names, use the "Clocks" report of the TimeQuest Timing Analyser and copy the name, or use the TimeQuest GUI to search for it.

Clock Specification

```
create_clock -period 100 [get_ports ADC_CLK_10]
create_clock -period  20 [get_ports MAX10_CLK1_50]
create_clock -period  20 [get_ports MAX10_CLK2_50]

derive_pll_clocks -create_base_clocks -use_net_name
derive_clock_uncertainty

create_generated_clock \
  -source [get_pins { SDRAM_PLL_Inst|*|clk[1] } ] \
  -name opClk_SDRAM [get_ports opClk_SDRAM]
```

To get the node names, use the "Clocks" report of the TimeQuest Timing Analyser and copy the name, or use the TimeQuest GUI to search for it.

Clock Report



THE RADAR MASTERS COURSE

Clock Groups

- Unless specified otherwise, Quartus assumes that all clocks are related and in the same clock domain
- When clocks are unrelated, all paths between them must be marked as “false paths”
- Do this with clock groups:

```
set_clock_groups -logically_exclusive \
  -group [get_clocks ADC_CLK_10]       \
  -group [get_clocks MAX10_CLK1_50]    \
  -group [get_clocks MAX10_CLK2_50]    \
  -group [get_clocks {opClk_SDRAM *altpll_component*}]
```

Clock Groups

- Unless specified otherwise, Quartus assumes that all clocks are related and in the same clock domain
- When clocks are unrelated, all paths between them must be marked as "false paths"
- Do this with clock groups:

```
set_clock_groups -logically_exclusive \
  -group [get_clocks ADC_CLK_10]        \
  -group [get_clocks MAX10_CLK1_50]     \
  -group [get_clocks MAX10_CLK2_50]     \
  -group [get_clocks {opClk_SDRAM *altpll_component*}]
```

Clock Groups

- Unless specified otherwise, Quartus assumes that all clocks are related and in the same clock domain
- When clocks are unrelated, all paths between them must be marked as “false paths”
- Do this with clock groups:

```
set_clock_groups -logically_exclusive \
  -group [get_clocks ADC_CLK_10]       \
  -group [get_clocks MAX10_CLK1_50]    \
  -group [get_clocks MAX10_CLK2_50]    \
  -group [get_clocks {opClk_SDRAM *altpll_component*}]
```

JTAG Timing Specification

- Standard across all Altera FPGAs
- Copied from Altera examples:

```
# Don't specify the altera_reserved_tck frequency,
# the compiler knows what it is, but put it in its own group
set_clock_groups -exclusive -group [get_clocks {altera_reserved_tck}]

set_input_delay  -clock altera_reserved_tck \
  -clock_fall 3 [get_ports {altera_reserved_tdi}]
set_input_delay  -clock altera_reserved_tck \
  -clock_fall 3 [get_ports {altera_reserved_tms}]
set_output_delay -clock altera_reserved_tck \
  -clock_fall 3 [get_ports {altera_reserved_tdo}]
```

Outline

Finite State Machines

Simulation

JTAG

Timing Constraints

Advanced Timing Constraints

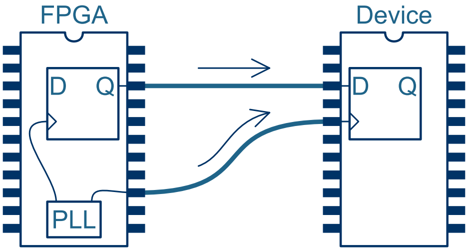
External Timing Requirements

- Synopsis Design Constraints start with the ideal case:
  - There are no PCB trace delays
  - The external setup requirement is zero
  - The external hold requirement is zero
  - The external propagation delay is zero
- Increase the maximum output delay for external setup timing
- Decrease the minimum output delay for external hold timing
- Specify the minimum and maximum input delays according to the external propagation delay parameters
- Worsen the situation with PCB trace delays and uncertainties (clock jitter, manufacturing tolerances, etc.)

External Timing Requirements

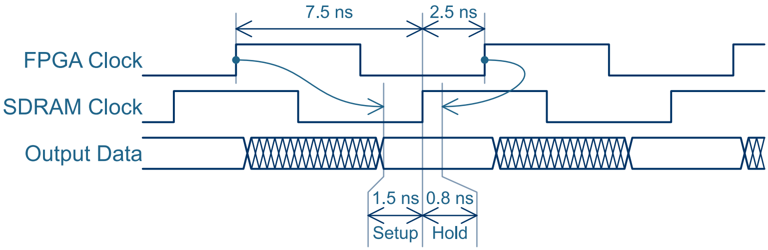
- Synopsis Design Constraints start with the ideal case:
  - There are no PCB trace delays
  - The external setup requirement is zero
  - The external hold requirement is zero
  - The external propagation delay is zero
- Increase the maximum output delay for external setup timing
- Decrease the minimum output delay for external hold timing
- Specify the minimum and maximum input delays according to the external propagation delay parameters
- Worsen the situation with PCB trace delays and uncertainties (clock jitter, manufacturing tolerances, etc.)

External Timing – Output



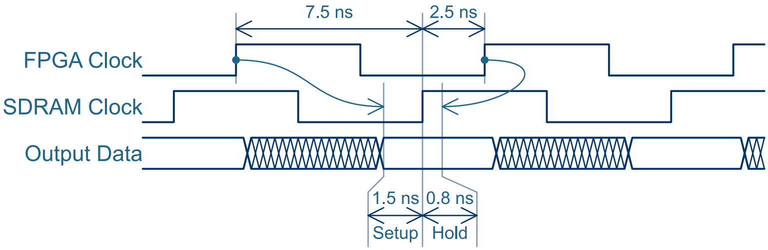
- FPGA internal delays and PCB trace delays cancel
- The important parameters are:
  - Setup and hold times of the external device
  - Clock jitter and other uncertainties
- Shift the external clock to ease the hold margin

External Timing – Output



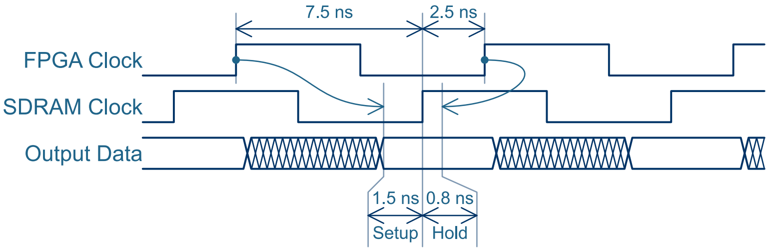
- FPGA internal delays and PCB trace delays cancel
- The important parameters are:
  - Setup and hold times of the external device
  - Clock jitter and other uncertainties
- Shift the external clock to ease the hold margin

External Timing – Output



- FPGA internal delays and PCB trace delays cancel
- The important parameters are:
  - Setup and hold times of the external device
  - Clock jitter and other uncertainties
- Shift the external clock to ease the hold margin

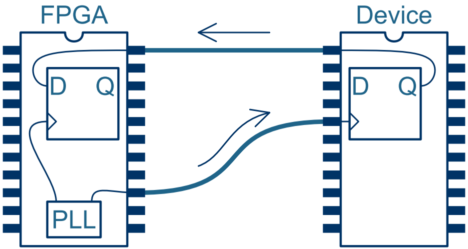
External Timing – Output




```
# Suppose 100 ps uncertainty
set_output_delay -max -clock opClk_SDRAM  1.6 [get_ports opSDRAM*]
set_output_delay -min -clock opClk_SDRAM -0.9 [get_ports opSDRAM*]
set_output_delay -max -clock opClk_SDRAM  1.6 [get_ports bpSDRAM*]
set_output_delay -min -clock opClk_SDRAM -0.9 [get_ports bpSDRAM*]
```

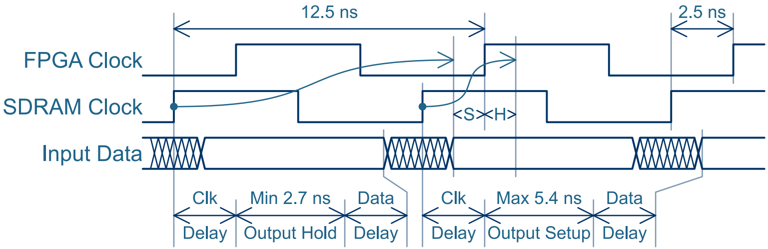
THE RADAR MASTERS COURSE

External Timing – Input



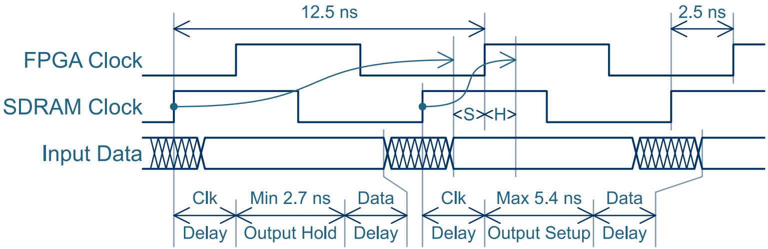
- Large FPGA internal delays and PCB trace delays
- Shift the external clock to ease the setup margin
- Multi-cycle requirement on the setup path (otherwise the compiler uses the 2.5 ns path) – the minimum propagation delay of 2.7 ns makes the 2.5 ns clock shift safe

External Timing – Input



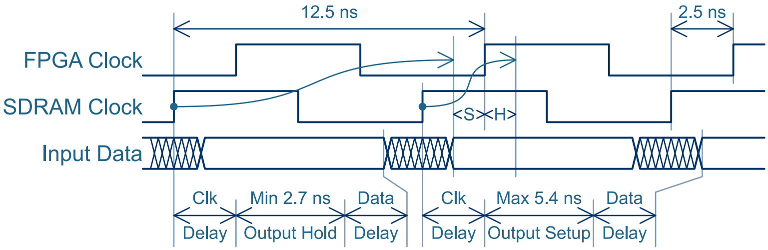
- Large FPGA internal delays and PCB trace delays
- Shift the external clock to ease the setup margin
- Multi-cycle requirement on the setup path (otherwise the compiler uses the 2.5 ns path) – the minimum propagation delay of 2.7 ns makes the 2.5 ns clock shift safe

External Timing – Input



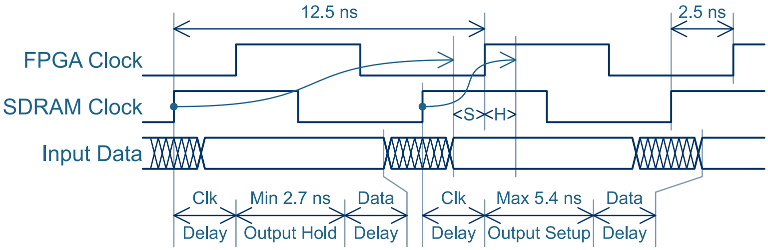
- Large FPGA internal delays and PCB trace delays
- Shift the external clock to ease the setup margin
- Multi-cycle requirement on the setup path (otherwise the compiler uses the 2.5 ns path) – the minimum propagation delay of 2.7 ns makes the 2.5 ns clock shift safe

External Timing – Input



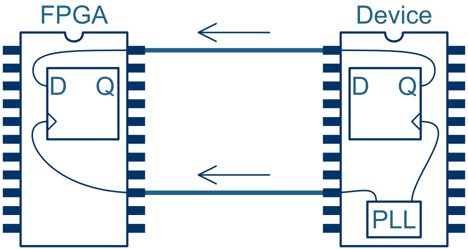

```
set_multicycle_path \
  -from [get_clocks opClk_SDRAM] \
  -to   [get_clocks SDRAM_PLL:SDRAM_PLL_Inst|*|wire_pll1_clk[0] ] \
  -setup 2
```

External Timing – Input



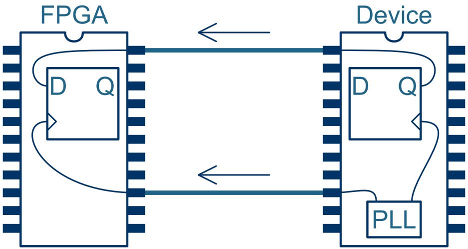

```
# Suppose 100 ps uncertainty and 200 ps PCB delay (each way)
set_input_delay -max -clock opClk_SDRAM 5.9 [get_ports bpSDRAM*]
set_input_delay -min -clock opClk_SDRAM 3.0 [get_ports bpSDRAM*]
```

External Timing – DDR ADC



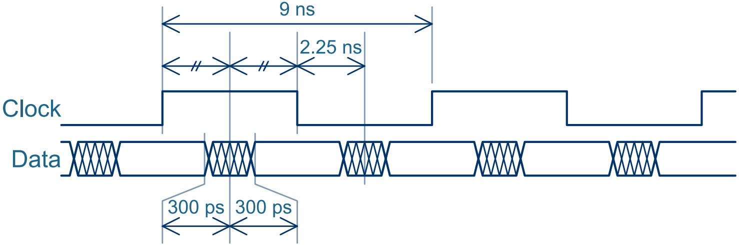
- The clock is sourced by the external device
- The PCB trace delays cancel
- The data is centre-aligned
- Use the dedicated serial LVDS receivers (on most FPGAs)

External Timing – DDR ADC



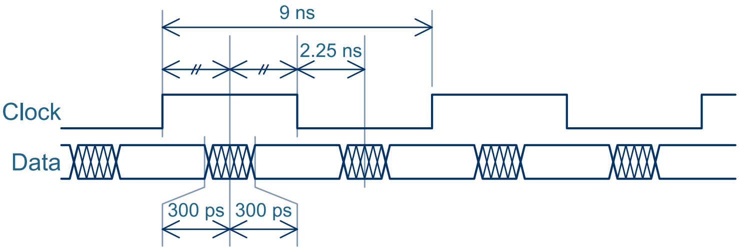
- The clock is sourced by the external device
- The PCB trace delays cancel
- The data is centre-aligned
- Use the dedicated serial LVDS receivers (on most FPGAs)

External Timing – DDR ADC



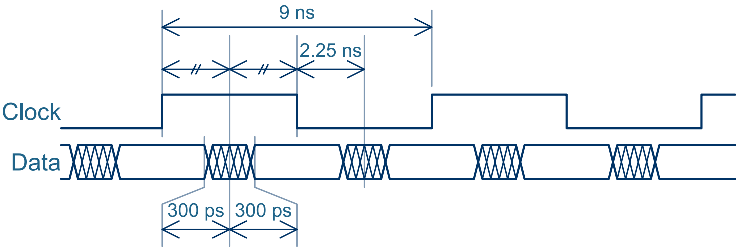
- The clock is sourced by the external device
- The PCB trace delays cancel
- The data is centre-aligned
- Use the dedicated serial LVDS receivers (on most FPGAs)

External Timing – DDR ADC



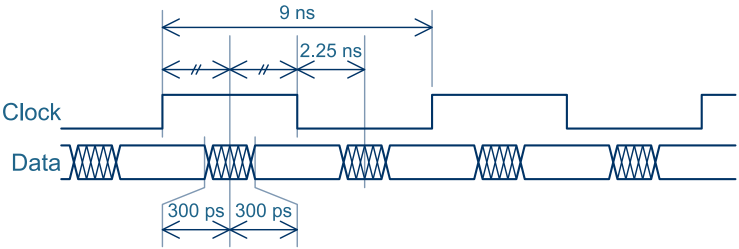
- The clock is sourced by the external device
- The PCB trace delays cancel
- The data is centre-aligned
- Use the dedicated serial LVDS receivers (on most FPGAs)

External Timing – DDR ADC




```
# Suppose 100 ps uncertainty
create_clock    -name  ADC_DCO -period 9 [get_ports ipADC_DCLK_P]
set_input_delay -clock ADC_DCO -min 1.85 [get_ports ipADC_CH*]
set_input_delay -clock ADC_DCO -max 2.65 [get_ports ipADC_CH*]
```

External Timing – DDR ADC




```
set_input_delay -clock ADC_DCO -clock_fall \
  -min 1.85 [get_ports ipADC_CH*] -add_delay
set_input_delay -clock ADC_DCO -clock_fall \
  -max 2.65 [get_ports ipADC_CH*] -add_delay
```

Timing-related Pin Attributes

- Use the correct I/O standard
- Set the correct output current
- Set the pin capacitance

```
set_instance_assignment -name IO_STANDARD "3.3-V LVTTL" -to opClk_SDRAM
set_instance_assignment -name IO_STANDARD "3.3-V LVTTL" -to opSDRAM*
set_instance_assignment -name IO_STANDARD "3.3-V LVTTL" -to bpSDRAM*
```

Timing-related Pin Attributes

- Use the correct I/O standard
- Set the correct output current
- Set the pin capacitance

```
set_instance_assignment -name CURRENT_STRENGTH_NEW 8MA -to opClk_SDRAM
set_instance_assignment -name CURRENT_STRENGTH_NEW 8MA -to opSDRAM*
set_instance_assignment -name CURRENT_STRENGTH_NEW 8MA -to bpSDRAM*
```

Timing-related Pin Attributes

- Use the correct I/O standard
- Set the correct output current
- Set the pin capacitance

```
set_instance_assignment -name BOARD_MODEL_NEAR_C 3.5P -to opClk_SDRAM
set_instance_assignment -name BOARD_MODEL_NEAR_C 3.8P -to opSDRAM*
set_instance_assignment -name BOARD_MODEL_NEAR_C 6.0P -to bpSDRAM*
```

Select References

- Stephen Brown and Zvonko Vranesic
  Fundamentals of Digital Logic with Verilog Design, 2nd Edition
  ISBN 978-0-07-721164-6
- Merrill L Skolnik
  Introduction to RADAR Systems
  ISBN 978-0-07-288138-7
- Mark A. Richards and James A. Scheer
  Principles of Modern Radar: Basic Principles
  ISBN 978-1-89-112152-4
- Deepak Kumar Tala
  World of ASIC
  http://www.asic-world.com/
- Jean P. Nicolle
  FPGA 4 Fun
  http://www.fpga4fun.com/

FPGA Development for Radar, Radio-Astronomy and Communications

THE RADAR MASTERS COURSE

Dept. Electrical Engineering, University of Cape Town
Private Bag, Rondebosch, 7701, South Africa
http://www.rrsg.uct.ac.za

Presented by Dr John-Philip Taylor

Convened by Dr Stephen Paine

Day 2 – 10 September 2024