Agenda

- Timing Analysis Basics
- Timing Analysis — TimeQuest
- Programmable I/O delay (Stratix III)
- Cyclone III External Memory Interfaces
How does timing verification work?

- Every device path in design must be analyzed with respect to timing specifications/requirements
  - Catch timing-related errors faster and easier than gate-level simulation & board testing
- Designer must enter timing requirements & exceptions
  - Used to guide fitter during placement & routing
  - Used to compare against actual results
Timing Analysis Basics

- Launch vs. latch edges
- Setup & hold times
- Data & clock arrival time
- Data required time
- Setup & hold slack analysis
- I/O analysis
- Recovery & removal
- Timing models
Path & Analysis Types

Three types of Paths:
1. Clock Paths
2. Data Path
3. Asynchronous Paths*

Two types of Analysis:
1. Synchronous  – clock & data paths
2. Asynchronous*  – clock & async paths

*Asynchronous refers to signals feeding the asynchronous control ports of the registers
Launch & Latch Edges

Launch Edge: the edge which “launches” the data from source register
Latch Edge: the edge which “latches” the data at destination register (with respect to the launch edge, typically 1 cycle)
**Setup & Hold**

Setup: The minimum time data signal must be stable BEFORE clock edge

Hold: The minimum time data signal must be stable AFTER clock edge

*Together, the setup time and hold time form a Data Required Window, the time around a clock edge in which data must be stable.*
Data Arrival Time

The time for data to arrive at destination register’s D input

\[
\text{Data Arrival Time} = \text{launch edge} + T_{\text{clk1}} + T_{\text{co}} + T_{\text{data}}
\]
Clock Arrival Time

- The time for clock to arrive at destination register’s clock input

\[
\text{Clock Arrival Time} = \text{latch edge} + T_{\text{clk2}}
\]
Data Required Time - Setup

- The minimum time required for the data to get latched into the destination register.

\[
\text{Data Required Time} = \text{Clock Arrival Time} - T_{su} - \text{Setup Uncertainty}
\]
Data Required Time - Hold

- The minimum time required for the data to get latched into the destination register

\[ \text{Data Required Time} = \text{Clock Arrival Time} + T_h + \text{Hold Uncertainty} \]
Setup Slack

- The margin by which the setup timing requirement is met. It ensures launched data arrives in time to meet the latching requirement.

\[ \text{Tsetup} = \text{TCLK2} - \text{Tdata} \]

- **Launched Data:**
  - **Launch Edge:** Tclk1
  - **Data Valid:** REG1.Q

- **Latch Edge:**
  - **Latch Edge:** Tclk2
  - **Data Valid:** REG2.D

\[ \text{Tdata} = \text{TCLK1} + \text{Tco} + \text{Tclk1} \]

\[ \text{Tsetup} = \text{Tclk1} + \text{Tco} + \text{Tclk2} - \text{Tdata} \]

\[ \text{Setup Slack} = \text{Data Valid} - \text{Data Valid} \]

- **Comb. Logic:**
  - REG1.D
  - REG1.Q
  - REG2.D
  - REG2.Q

- **Timing Parameters:**
  - Tclk1
  - Tclk2
  - Tco
  - Tdata
  - Tsetup
  - Tsu
Setup Slack (cont’d)

Setup Slack = Data Required Time
– Data Arrival Time

Positive slack
– Timing requirement met

Negative slack
– Timing requirement not met
Hold Slack

- The margin by which the hold timing requirement is met. It ensures latch data is not corrupted by data from another launch edge. It also prevents “double-clocking”.

![Diagram of Hold Slack]

**Definitions:**
- **Tclk1**: Margin by which the hold timing requirement is met.
- **Tclk2**: Margin by which the hold timing requirement is met.
- **Tco**: Margin by which the hold timing requirement is met.
- **Tdata**: Margin by which the hold timing requirement is met.
- **TCO**: Margin by which the hold timing requirement is met.

**Key Components:**
- **REG1**: First register with Tclk1 and Tco margins.
- **REG2**: Second register with Tclk2 and Th margin.
- **Comb. Logic**: Logic that operates on the data between REG1 and REG2.

**Timelines:**
- **CLK**: Input clock signal.
- **REG1.CLK**: Clock signal for REG1.
- **REG1.Q**: Output of REG1.
- **REG2.D**: Input to REG2.
- **REG2.CLK**: Clock signal for REG2.
- **Hold Slack**: Margin between the latches to ensure data integrity.

© 2008 Altera Corporation—Public

Altera, Stratix, Arria, Cyclone, MAX, HardCopy, Nios, Quartus, and MegaCore are trademarks of Altera Corporation.
Hold Slack (cont’d)

Hold Slack = Data Arrival Time – Data Required Time

Positive slack
– Timing requirement met

Negative slack
– Timing requirement not met
I/O Analysis

- Analyzing I/O performance in a synchronous design uses the same slack equations
  - Must include external device & PCB timing parameters

* Represents delay due to capacitive loading
Recovery & Removal

Recovery: The minimum time an asynchronous signal must be stable BEFORE clock edge

Removal: The minimum time an asynchronous signal must be stable AFTER clock edge
Asynchronous = Synchronous?

- Asynchronous control signal source is assumed synchronous
  - Slack equations still apply
    - data arrival path = asynchronous control path
    - $T_{su} \approx T_{rec}$; $T_{h} \approx T_{rem}$
  - External device & board timing parameters may be needed (Ex. 1)

Example 1

Example 2
Why Are These Calculations Important?

- Calculations are important when timing violations occur
  - Need to be able to understand cause of violation
- Example causes
  - Data path too long
  - Requirement too short (incorrect analysis)
  - Large clock skew signifying a gated clock, etc.
- TimeQuest uses them
  - Equations to calculate slack
  - Terminology (launch and latch edges, Data Arrival Path, Data Required Path, etc.) in timing reports
Timing Models

- Quartus II software models device timing at two PVT conditions by default
  - **Slow Corner** Model
    - Indicates slowest possible performance for any single path
    - Timing at 85 °C junction temp. and VCC_{MIN}
  - **Fast Corner** Model
    - Indicates fastest possible performance for any single path
    - Timing at 0 °C junction temp. and VCC_{MAX}

- Why two corner timing models?
  - Ensure **setup** timing is met in **slow model**
  - Ensure **hold** timing is met in **fast model**
    - Essential for source synchronous interfaces

- Third model (slow, 0 °C) available only for Stratix III and Cyclone III devices to support 65 nm and smaller technology (temperature inversion phenomenon)
Generating Fast/Slow Netlist

- Specify the timing model to be used when creating your netlist
- Default is the slow timing netlist
- To specify fast timing netlist
  - Use `-fast_model` option with `create_timing_netlist` Command
  - Choose Fast corner in GUI when executing Create Timing Netlist from Netlist menu
  - CANNOT run from Tasks Pane
Specifying Operating Conditions

- Perform timing analysis for different delay models without recreating the existing timing netlist
- Takes precedence over already generated netlist
- Required for selecting slow, 0 °C model for Stratix III & Cyclone III
- Use `get_available_operating_conditions` to see available conditions for target device
Quartus II Software Design Series: Foundation

Timing Analysis
Timing Analysis Agenda

- TimeQuest GUI
- Using TimeQuest
- Using TimeQuest in the Quartus II flow

Note: For more details on verifying designs for timing, please attend the course “Quartus II Software Design Series: Verification”
Online training also available: Validating Performance with the TimeQuest Static Timing Analyzer
TimeQuest Timing Analyzer

- Timing engine in Quartus II software
- Provides timing analysis solution for all levels of experience
- Features
  - Synopsys Design Constraints (SDC) support
    - Standardized constraint methodology
  - Easy-to-use interface
    - Constraint entry
    - Standard reporting
  - Scripting emphasis
    - Presentation focuses on using GUI

© 2008 Altera Corporation—Public
Altera, Stratix, Arria, Cyclone, MAX, HardCopy, Nios, Quartus, and MegaCore are trademarks of Altera Corporation
Steps to Using TimeQuest

1. Generate timing netlist
2. Enter SDC constraints
   a. Create and/or read in SDC file (recommended method)
   or
   b. Constrain design directly in console
3. Update timing netlist
4. Generate timing reports
5. Save timing constraints (optional)
Opening TimeQuest

- Toolbar button
- Tools menu
- Stand-alone mode
  - quartus_staw
- Command line
Tasks Pane

- Provides quick access to common operations
  - Command execution
  - Report generation
- Executes most commands with default settings
- Use menus for non-default settings

Double-click to execute any command
Report Pane

- Displays list of generated reports currently available for viewing
  - Reports generated by Tasks pane
  - Reports generated using report commands

Highlight report to see detail in View window.
View Pane

- Main viewing area that displays report table contents & graphical results
Viewing Multiple Reports

Click & drag ‘+’ sign to divide view pane into multiple windows
Viewing Multiple Reports Example

- View pane split into two windows
- Use Target Pane button to force a selected report to appear in a pane
- Drag bars to edges to remove splits
- Highlight window, then highlight report in Reports pane you would like to appear there
**Console pane**

- Allows direct entry and execution of SDC & Tcl commands
  - Displays equivalent of command executed by GUI
- Displays TimeQuest output messages
- History tab records all executed SDC & Tcl commands
  - Copy & paste to create scripts or SDC files
SDC File Editor = Quartus II Text Editor

- Use Quartus II editor to create and/or edit SDC
- SDC editing unique features (for .sdc files)
  - Access to GUI dialog boxes for constraint entry (Edit ⇒ Insert Constraint)
  - Syntax coloring
  - Tooltip syntax help

TimeQuest File menu ⇒ New/Open SDC File
Quartus II File menu ⇒ New ⇒ Other Files tab

Place cursor over command to see tooltip
SDC File Editor (cont.)

Construct an SDC file using TimeQuest graphical constraint creation tools.

Create Clock

- Period: 10.000 ns
- Target: [get_ports (clk)]

SDC command:
```
create_clock -period 10.000 -name clk [get_ports (clk)]
```
Steps to Using TimeQuest

1. Generate timing netlist
2. Enter SDC constraints
   a. Create and/or read in SDC file (recommended method)
   b. Constrain design directly in console
3. Update timing netlist
4. Generate timing reports
5. Save timing constraints (optional)
1) Generate Timing Netlist

- Creates timing netlist (i.e. database) based on compilation results
  - Post-synthesis (mapping) or post-fit
  - Worst-case (slow), best-case (fast) timing models, or set operating conditions
    (needed for Stratix III and Cyclone III devices only)

- To execute:

  **Netlist menu**

  **Tasks pane**

  **Tcl equivalent of command**

  ```
  Tcl: create_timing_netlist
  ```
2a) Create and/or Read in SDC File

- Create SDC file using SDC file editor
- Read in constraints & exceptions from existing SDC file
  - Skip if no SDC file
- Execution
  - Read SDC File (Tasks pane or Constraints menu)
- File precedence (if no filename specified)
  - Files specifically added to Quartus II project
  - <current_revision>.sdc (if it exists)

*Tcl: read_sdc [<filename>]
2b) Constrain Directly in Console

- Add new constraints directly
  - Not automatically added to SDC file
  - Use GUI or Console pane
  - Not needed if all constraints in SDC file

- Recommend using SDC file (step 2a) instead to ease management and storage of constraints

- Example constraints
  - create_clock
  - create_generated_clock
  - set_input_delay
  - set_output_delay
Using GUI to Enter Constraints Directly

- Most common constraints can be accessed from the Constraints menu.
- Same as Edit menu ⇒ Insert Constraints in SDC file editor.
- Use if unfamiliar with SDC syntax.
Constraining

- User MUST enter constraints for all paths to fully analyze design
  - TimeQuest only performs slack analysis on constrained design paths
  - Recommendation: Constrain all paths (clocks & I/O)

- Not as difficult a task as it may sound
  - Wildcards
  - Single, generalized constraints cover many paths, even all paths in an entire clock domain
3) Update Timing Netlist

- Applies SDC constraints/exceptions to current timing netlist
- Generates warnings
  - Undefined clocks
  - Partially defined I/O delays
  - Combinatorial loops
- Update timing netlist after adding any new constraint
- Execution
  - Update Timing Netlist (Tasks pane or Netlist menu)

*Tcl: update_timing_netlist*
4) Generate Timing Reports

- Verify timing requirements and locate violations
- Check for fully constrained design or ignored timing constraints
- Two Methods
  - Tasks pane
    - Automatically creates/updates netlist & reads default SDC file if needed
  - Reports menu
    - Must have valid netlist to access
    - Tasks pane or Reports menu

Double-click on individual report
“Out of Date” Reports

- Adding new constraints causes current reports to be “out of date”
- Update timing netlist & regenerate reports (report right-click menu)
Reset Design Command

- Located in Tasks pane
- Flushes all timing constraints from current timing netlist
  - Functional Tcl equivalent: `delete_timing_netlist` command followed by `create_timing_netlist`
- Uses
  - “Re-starting” timing analysis on same timing netlist applying different constraints or SDC file
  - Starting analysis over if results seem to be unexpected
5) Save Timing Constraints (Optional)

- **write_sdc command**
  - Saves all constraints & exceptions applied to current netlist into SDC file
  - Use if constraints added during TimeQuest session in console instead of SDC file

- **Notes**
  - SDC files generated by TimeQuest only if requested
  - Converts Altera-specific SDC commands into standard SDC
  - Run `report_sdc` command (console, Tasks pane, or Report menu) to see what will get written to SDC file
Steps to Using TimeQuest (Review)

1. Generate timing netlist
2. Enter SDC constraints
   a. Create and/or read in SDC file (recommended method)

   or

   b. Constrain design directly in console

3. Update timing netlist
4. Generate timing reports
5. Save timing constraints (optional)
Using TimeQuest in Quartus II Flow

1. Synthesize Quartus II project
2. Use TimeQuest to specify timing requirements
3. Enable TimeQuest in Quartus II project
4. Perform full compilation (run Fitter)
5. Verify timing in TimeQuest
Specifying Timing Requirements

- Follow TimeQuest steps
- Use `-post_map` argument for synthesis (mapping) only netlist
  - If the design is already fully compiled, then choose `-post_fit` (default)

- Enter constraints via Constraints menu or Console pane
## SDC Netlist Terminology

<table>
<thead>
<tr>
<th>Term</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cell</td>
<td><strong>Device building blocks</strong> (e.g. look-up tables, registers, embedded multipliers, memory blocks, I/O elements, PLLs, etc.)</td>
</tr>
<tr>
<td>Pin</td>
<td>Input or outputs of cells</td>
</tr>
<tr>
<td>Net</td>
<td>Connections between pins</td>
</tr>
<tr>
<td>Port</td>
<td>Top-level inputs and outputs (e.g. device pins)</td>
</tr>
</tbody>
</table>
**SDC Netlist Example**

- **Pathways defined in constraints by targeted endpoints (pins or ports)**

Sample Pin Names:
- `ina|combout`
- `inreg|datin`
- `inreg|clk`
- `inreg|regout`
- `ab|combout`
- `ab|dific`

Sample Net Names:
- `ina~combout`
- `ab`
- `clk~clkctrl`
- `inreg`
Collections

- Searches and returns from the design netlist with a list of names meeting criteria
- Used in SDC commands
  - Some collections searched automatically during commands usage and may not need to be specified
- Examples
  - get_ports
  - get_pins
  - get_clocks
  - all_clocks
  - all_registers
  - all_inputs
  - all_outputs

See “TimeQuest Timing Analyzer” chapter of the Quartus II Software Handbook for a complete list & description of each
Create Clock

Create Clock fields:
- **Clock Name** – Assign name to clock setting; defaults to target node name
- **Period** – Clock period in nanoseconds
- **Waveform edges** – Use for non-50% duty cycle clocks
- **Targets** – Port or pin to which clock setting is being applied

**Important Note**: Unlike the Classic Timing Analyzer, all design clocks are related by default. This means TimeQuest will analyze paths between clock domains whether you have specifically related them or not.

**SDC**: `create_clock`

© 2008 Altera Corporation—Public
Altera, Stratix, Arria, Cyclone, MAX, HardCopy, Nios, Quartus, and MegaCore are trademarks of Altera Corporation
Name Finder

Clicking on Browse button opens Name Finder allowing you to search netlist for node names (similar to Quartus II Node Finder)

Select collection to search

Edit command here or final command to use wildcards

Generated Clocks

Generated clocks are clock signals derived from a previously created clock
- E.g. clock dividers, ripple clocks, PLLs
- Must be defined by a constraint

Create Generated Clock fields:
- Clock Name – Assign name to clock setting
- Relationship to source – Specify how generated clock is related to base clock. The Based on waveform section allows for more complexity in the relationship to the base clock (not discussed)
- Targets – Port or pin to which clock setting is being applied

SDC: create_generated_clock
PLL Clocks (Altera SDC Extension)

- **Command:** `derive_pll_clocks`
  - `[−use_tan_name]`: names clock after design net name instead of the default PLL output SDC pin name
- Create generated clocks on all PLL outputs
  - Based on input clock & PLL settings
- Requires defining PLL input as clock
- Automatically updates generated clocks on PLL outputs as changes made to PLL design
- `write_sdc -expand` expands constraint into standard `create_generated_clock` commands

- **Not in GUI; must be entered in SDC manually**
**derive_pll_clocks Example**

```vhdl
create_clock -period 10.0 [get_ports in_clk]
create_generated_clock -name c100
    -source [get_pins {inst|altpll_component|pll|inclk[0]}]
    -divide_by 1
    [get_pins {inst|altpll_component|pll|clk[0]}]
create_generated_clock -name c200
    -source [get_pins {inst|altpll_component|pll|inclk[0]}]
    -multiply_by 2
    [get_pins {inst|altpll_component|pll|clk[1]}]
create_generated_clock -name c200_shift
    -source [get_pins {inst|altpll_component|pll|inclk[0]}]
    -multiply_by 2
    -phase 90
    [get_pins {inst|altpll_component|pll|clk[2]}]
```

**Using generated clock commands**

**Using derive pll command**

```vhdl
create_clock -period 10.0 [get_ports in_clk]
derive_pll_clocks
# Note the clock names for the generated clocks will be the names of the PLL output pins
```

© 2008 Altera Corporation—Public
Altera, Stratix, Arria, Cyclone, MAX, HardCopy, Nios, Quartus, and MegaCore are trademarks of Altera Corporation
I/O Constraining

- Specify **system-level** timing constraints
- **Settings**
  - Input maximum delay
  - Input minimum delay
  - Output maximum delay
  - Output minimum delay
Input Maximum/Minimum Delay

- Maximum/minimum delay from external device to Altera I/O
  - Represents external device max & min $t_{co}$ + PCB delay - PCB clock skew
- Constrains registered input path ($t_{su}/t_h$)

$$t_{su_A} \leq t_{CLK} - \text{Input Maximum Delay}$$
$$t_{h_A} \leq \text{Input Minimum Delay}$$
Output Maximum/Minimum Delay

- Maximum/minimum delay from Altera I/O to external device
  - Represents external device $t_{su}/t_{h} +$ PCB delay - PCB clock skew
- Constrains registered output path (max & min $t_{co}$)

$$t_{co_B} \leq t_{CLK} - \text{Output Maximum Delay}$$

$$t_{co_B} \geq \text{Output Minimum Delay}$$
Set Input/Output Delay

Set Input/Output Delay fields:
- **Clock Name** – Specify source clock
- **Input delay options** – Choose max or min constraint. Rise/Fall indicate if the constraint applies particularly to a rising or falling edge transition (advanced).
- **Delay value** – Total off chip delay
- **Add delay** – Must use if applying multiple sets of input/output delays to the same port (e.g. input ports feeding multiple internal registers)
- **Targets** – Port to which setting is being applied

**SDC:** `set_input_delay`
**SDC:** `set_output_delay`

© 2008 Altera Corporation—Public
Altera, Stratix, Arria, Cyclone, MAX, HardCopy, Nios, Quartus, and MegaCore are trademarks of Altera Corporation
I/O Constraints

- Combinatorial I/O interface
- Synchronous I/O interface
- Source synchronous interface
**Combinatorial Interface**

- All paths from IN to OUT need to be constrained

- **Use** `set_max_delay` & `set_min_delay` commands
  - Specify an absolute maximum & minimum delay between points

- **Options**
  
  ```
  [-from <names>]
  [-to <names>]
  [-fall_from <clocks>]
  [-rise_from <clocks>]
  [-fall_to <clocks>]
  [-rise_to <clocks>]
  [-through]
  <delay>
  ```

---

© 2008 Altera Corporation—Public
Altera, Stratix, Arria, Cyclone, MAX, HardCopy, Nios, Quartus, and MegaCore are trademarks of Altera Corporation
**set_max_delay & set_min_delay Notes**

- **-from & -to:** Use to indicate source & destination nodes for constraints
- **-through:** Use to indicate the constraint should only be applied to path(s) going through a particular node name
- **-fall_from, -rise_from:** delay begins at selected clock’s edge; *not in GUI*
- **-fall_to, -rise_to:** delay ends at selected clock’s edge; *not in GUI*
set_max_delay & set_min_delay (GUI)
Combinatorial Interface Example

FPGA/CPLD

in1

in2

in3

Combinatorial Logic

out1

out2

set_max_delay -from [get_ports in1] -to [get_ports out*] 5.0
set_max_delay -from [get_ports in2] -to [get_ports out*] 7.5
set_max_delay -from [get_ports in3] -to [get_ports out*] 9.0

set_min_delay -from [get_ports in1] -to [get_ports out*] 1.0
set_min_delay -from [get_ports in2] -to [get_ports out*] 2.0
set_min_delay -from [get_ports in3] -to [get_ports out*] 3.0
Synchronous Inputs

- Need to specify timing relationship from ASSP to FPGA/CPLD to guarantee setup/hold in FPGA/CPLD

\[ T_{co} \] represents total clock-to-output time of ASSP (i.e. datasheet spec)

\[ T_{co} \] = total clock-to-output time of ASSP

\[ T_{clk1} \] = clock to clock 1

\[ T_{clk2int} \] = clock to clock 2 internal

\[ T_{clk2ext} \] = clock to clock 2 external

\[ T_{data_{PCB}} \] = data on PCB

\[ T_{data_{int}} \] = data internal

\[ C_L \] = capacitive loading

\[ T_{su}/T_{su} \] = setup time

\[ T_{sh} \] = hold time

* Represents delay due to capacitive loading
Constraining Synchronous Inputs

- Use `set_input_delay (-max option)` command to constrain input setup time (maximum time to arrive and still meet $T_{su}$)
  - Calculated input delay value represents all delays external to device

  \[
  \text{input delay max} = \text{Board Delay (max)} - \text{Board clock skew (min)} + T_{co(max)} \\
  = (T_{data\_PCB(max)} + T_{CL}) - (T_{clk2ext(min)} - T_{clk1(max)}) + T_{co(max)} \\
  \]

  \[
  \text{data arrival time} = \text{launch edge} + \text{input delay max} + T_{dataint} \\
  \]

  \[
  \text{data required time} = \text{latch edge} + T_{clk2int} - T_{su} \\
  \]

  \[
  \text{slack} = \text{required time} - \text{data arrival time} \\
  \]

- Use `set_input_delay (-min option)` command to constrain input hold time (minimum time to stay active and still meet $T_{h}$)
  - Calculated input delay value represents all delays external to device

  \[
  \text{input delay min} = \text{Board Delay (min)} - \text{Board clock skew (max)} + T_{co(min)} \\
  = (T_{data\_PCB(min)} + T_{CL}) - (T_{clk2ext(max)} - T_{clk1(min)}) + T_{co(min)} \\
  \]

  \[
  \text{data arrival time} = \text{launch edge} + \text{input delay min} + T_{dataint} \\
  \]

  \[
  \text{required time} = \text{latch edge} + T_{clk2int} + T_{h} \\
  \]

  \[
  \text{slack} = \text{data arrival time} - \text{data required time} \\
  \]
set_input_delay Command

- Constrains input pins by specifying external device timing parameters

- Options
  -clock <clock_name>
  [-clock_fall]
  [-rise | -fall]
  [-max | -min]
  [-add_delay]
  [-reference_pin <target>]
  [-source_latency_included]
  <delay value>
  <targets>
set_input_delay Notes

- **clock**: Specifies the clock driving the source (external) register
  - Used to determine launch edge vs. latch edge relationship

- **clock_fall**: Use to specify input signal was launched by a falling edge clock transition

- **-rise | -fall**: Use to indicate whether input delay value is for a rising or falling edge transaction

- **-add_delay**: Use to specify multiple constraints on single input
  - Only one set of max/min & rise/fall constraints allowed on an input pin
    - Ex. Constraining one input port driving two registers in different clock domains would require the `-add_delay` option
**set_input_delay** Notes

- **-reference_pin**: Use to specify that delays are with respect to some other port or pin in the design
  - Example: Feedback clock: Input delay is relative to an output port being fed by a clock

- **-source_latency_included**: input delay value specified includes clock source latency normally added automatically
  - Tells TimeQuest to ignore any clock latency constraints applied to source clock

To fully constrain, must specify both **-max** & **-min**
- Each will default to the value of the other setting if only one assigned (same with rise/fall)
- Warning message if one or the other not specified
Synchronous Outputs

- Need to specify timing relationship from FPGA/CPLD to ASSP to guarantee clock-to-output times in FPGA/CPLD

*T represents delay due to capacitive loading*
**Constraining Synchronous Outputs**

- **Use** `set_output_delay` (-max option) **command** to **constrain maximum clock-to-output** (maximum time to arrive and still meet ASSP’s $T_{su}$)
  - Calculated output delay value represents all delays external to device

  \[
  \text{output delay max} = \text{Board Delay (max)} - \text{Board clock skew (min)} + T_{su}
  \]
  \[
  = (T_{\text{data PCB(max)}} + T_{CL}) - (T_{\text{clk2(min)}} - T_{\text{clk1ext(max)}}) + T_{su}
  \]

  \[
  \text{data arrival time} = \text{launch edge} + T_{\text{clk1int}} + T_{\text{co(max)}} + T_{\text{dataint}}
  \]
  \[
  \text{data required time} = \text{latch edge} - \text{output delay max}
  \]
  \[
  \text{slack} = \text{data required time} - \text{data arrival time}
  \]

- **Use** `set_output_delay` (-min option) **command** to **constrain minimum clock-to-output** (minimum time to stay active and still meet ASSP’s $T_{h}$)
  - Calculated output delay value represents all delays external to device

  \[
  \text{output delay min} = \text{Board Delay (min)} - \text{Board clock skew (max)} - T_{h}
  \]
  \[
  = (T_{\text{data PCB(min)}} + T_{CL}) - (T_{\text{clk2(max)}} - T_{\text{clk1ext(min)}}) - T_{h}
  \]

  \[
  \text{data arrival time} = \text{launch edge} + T_{\text{clk1int}} + T_{\text{co(min)}} + T_{\text{dataint}}
  \]
  \[
  \text{data required time} = \text{latch edge} - \text{output delay min}
  \]
  \[
  \text{slack} = \text{data arrival time} - \text{data required time}
  \]
set_output_delay Command

- Constrains output pins by specifying external device timing parameters

- Options
  - `clock <clock_name>`
  - `[-clock_fall]`
  - `[-rise | -fall]`
  - `[-max | -min]`
  - `[-add_delay]`
  - `[-reference_pin <target>]`
  - `<delay value>`
  - `<targets>`
set_output_delay Notes

- Same notes as set_input_delay command
Input/Output Delays (GUI)

Set Input Delay

- Clock name: clk
- Use falling clock edge
- Input delay options:
  - Minimum
  - Maximum
  - Both
  - Rise
  - Fall
  - Both
- Delay value: 5 ns
- Add delay
- Targets: [get_ports d^*]
- SDC command:
  - set_input_delay -clock { clk } -max 5 [get_ports d^*]
Source-Synchronous Interfaces

- Both data & clock transmitted by host device with designated phase relationship (e.g. edge or center-aligned)
  - No clock tree skew included in calculation
  - Target device uses transmitted clock to sample incoming data
- Data & clock routed identically to maintain phase relationship at destination device
  - Board delay not included in external delay calculations
    - Clock trace delay (data required time) & Data trace delay (data arrival time) are equal and offset
  - Enables higher interface speeds (compared to using system clock)

*The PLL in this example, represented by a single symbol, is actually generating multiple outputs clocks*
**SDR Source-Synchronous Input (Center-Aligned)**

- Total setup/hold relationship of FPGA to clock (clkin) already defined by output waveform of external device
  - $T_{su}$ is start of DVW
  - $T_h$ is end of DVW
- Must derive `set_input_delay` values from $T_{su}$ & $T_h$

* The PLL in this example is used to maintain the input clock to data relationship

© 2008 Altera Corporation—Public

Altera, Stratix, Arria, Cyclone, MAX, HardCopy, Nios, Quartus, and MegaCore are trademarks of Altera Corporation
input delay max = board delay (max) - clock delay (min) + $T_{co(max)}$

setup slack = data required time - data arrival time

If setup slack = 0 (start of DVW):

data arrival time = data required time

launch edge + input delay max = latch edge - $T_{su}$

SO

input delay max = (latch edge - launch edge)$^*$ - $T_{su}$

*Typically 1 clock period for SDR

Note: In reality for high-speed designs, there would be some max/min board & clock delay that would need to be figured into the analysis.
input delay min = \( \text{board delay (min)} - \text{clock delay (max)} + T_{\text{co(min)}} \)

hold slack = \( \text{data arrival time} - \text{data required time} \)

If hold slack = 0 (end of DVW):
- data required time = data arrival time
- latch edge + \( T_h \) = launch edge + input delay min

For hold analysis, latch and launch edges cancel out, so
input delay min = \( T_h \)

Note: In reality for high-speed designs, there would be some max/min board & clock delay that would need to be figured into the analysis.
Using SDC with Source-Sync Input

- Create clock on clock input port
- **Use** `set_input_delay` command with reference to clock input
  - Same as with synchronous input
  - Do not include board delay parameters in value
output delay max = board delay (max) – clock delay (min) + T_s
= T_s

output delay min = board delay (min) – clock delay (max) – T_h
= -T_h

Notes:
1) In reality for high-speed designs, there would be some max/min board & clock delay that would need to be figured into the analysis.
2) The PLL in this example is used to shift output clock to establish an output clock to data relationship
Using SDC with Source-Synch Output

- Must tell TimeQuest to analyze path from clock source to output clock port during analysis
- Use `set_output_delay` command on `dataout` with reference to generated clock on output port
  - Create generated clock on output clock port (source is PLL output pin)
  - Use `-clock` argument in output delay assignment to associate output clock to output data bus
- Path from PLL output pin to output port still considered unconstrained by TimeQuest
  - Constrain path from PLL pin to output port with false path (described later), `set_min/max_delay`, or `set_output_delay`
Constraining Source-Sync Output Example

```vhdl
create_clock 5 -name clkin \[get_ports clkin]
create_generated_clock -name pllclk divide_by 1 \-source [get_ports clkin] [get_pins inst|altpll_component|p1l|clk[0]]

# Place clock on external clock output
create_generated_clock -name clkout \-source [get_pins inst|altpll_component|p1l|clk[0]] \-divide_by 1 \[get_ports clkout]

# Constrain dataout with an external tsu of 0.5 ns
# and th of 0.5 ns using clkout as clock
set_output_delay -clock [get_clocks clkout] 
  -max 0.500 \[get_ports dataout]
set_output_delay -clock [get_clocks clkout] 
  -min -0.500 \[get_ports dataout]
```
Source Synchronous Summary (Center-Aligned)

Waveform @ input to external device
Waveform @ output to external device

<table>
<thead>
<tr>
<th></th>
<th>Maximum</th>
<th>Minimum</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input delay setting (ns)</td>
<td>(latch edge – launch edge) - $T_{su}$</td>
<td>$T_h$</td>
</tr>
<tr>
<td>Output delay setting (ns)</td>
<td>$T_{su}$</td>
<td>$-T_h$</td>
</tr>
</tbody>
</table>
Source Synchronous (Edge-Aligned)

Waveform @ input to external device
Waveform @ output to external device

**Launch edge**

**Latch edge**

<table>
<thead>
<tr>
<th></th>
<th>Maximum</th>
<th>Minimum</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input delay setting (ns)</td>
<td>(latch edge – launch edge) - $T_{su}$</td>
<td>$-T_h$</td>
</tr>
<tr>
<td>Output delay setting (ns)</td>
<td>$T_{su}$</td>
<td>$T_h$</td>
</tr>
</tbody>
</table>

$INCLK$

$DVW$

$T_{su}$

$T_h$
Useful Reports for Design Constraining

- **Report Clocks**
  - Use to ensure all clocks have been defined correctly

- **Report Unconstrained Paths**
  - Use to determine if any constraints are missing

- **Report SDC**
  - Use to review what constraints have currently been applied to the netlist

- **Check Timing**
  - Use to check issues with design or applied constraints

- **Report Clock Transfers**
  - Use to determine nets crossing clock domains
  - Remember, by default, all clock domains are related and analyzed with respect to one another
    - Paths between domains might need to be set as false paths

- **Report Ignored Constraints**
  - Use to determine if any constraints are being ignored
    - Intentionally ignored (false paths)
    - Unintentionally ignored (possible typos or other errors in constraints)
Using TimeQuest in Quartus II Flow

1. Synthesize Quartus II project
2. Use TimeQuest to specify timing requirements
3. Enable TimeQuest in Quartus II project
4. Perform Full Compilation (run Fitter)
5. Verify timing in TimeQuest
Enabling TimeQuest in Quartus II

- Tells Quartus II to use SDC constraints during fitting
- File order precedence
  1. Any SDC files added to Quartus II project (in order)
  2. `<current_revision>.SDC`
Enabling TimeQuest in Quartus II Software

Notes:
- Arria GX only supports TimeQuest.
- TimeQuest is enabled by default for new Stratix III and Cyclone III designs.
Adding SDC File to Quartus II Project

- Add SDC files to TimeQuest Timing Analyzer page of Settings dialog box
- Multicorner analysis checks all process corners in one analysis
  - On by default for Cyclone II & III, Stratix II & III devices

![Image of Quartus II Settings dialog box with SDC file addition options]
Quartus Settings File (QSF)

- SDC constraints are not stored in QSF
- TimeQuest uses script to convert QSF timing assignments to SDC
  - Constraints menu
  - Done automatically if no SDC file exists when first opening TimeQuest

See Quartus II Handbook Chapter, “Switching to the TimeQuest Timing Analyzer” for details
  - Differences between Classic Timing Analyzer and TimeQuest
  - Details on conversion utility

Online training also available
  - Switching to the TimeQuest Timing Analyzer

© 2008 Altera Corporation—Public
Altera, Stratix, Arria, Cyclone, MAX, HardCopy, Nios, Quartus, and MegaCore are trademarks of Altera Corporation
Timing Driven Compilation (TDC)

- Directs fitter to place & route logic to meet timing assignments
  - Optimize timing (on by default)
    - Placing nodes in critical paths closer together
    - Located in Fitter settings “More Settings…” box
  - Optimize fast-corner timing
    - Off: optimize for slow corner only
    - On: add optimization for fast process as well (minimum timing models)
    - Can add up to 10% to compile time
Optimize Hold Timing

- Modifies place & route to meet hold or minimum timing requirements
  - May add additional routing in path

- Settings
  - Any I/O & minimum $t_{pd}$ paths (default)
  - All paths (I/O & internal)
Using TimeQuest in Quartus II Flow

1. Synthesize Quartus II project
2. Use TimeQuest to specify timing requirements
3. Enable TimeQuest in Quartus II project
4. Perform full compilation (run Fitter)
5. Verify timing in TimeQuest
Verifying Timing Requirements

- View TimeQuest summary information directly in Quartus II Compilation Report
- Open TimeQuest for more thorough analysis
  - Follow TimeQuest flow
  - Run TimeQuest easy-to-use reporting capabilities (Tasks pane)
  - Place Tcl reporting commands into script file
    - Easy repetition
### TimeQuest Summary Reports

#### SDC files used during fitting
- Clocks generated
- Timing violations
- Unconstrained paths

#### Setup Summary

<table>
<thead>
<tr>
<th>Clock</th>
<th>Slack</th>
<th>End Point TNS</th>
</tr>
</thead>
<tbody>
<tr>
<td>clk_core</td>
<td>-2.230</td>
<td>-1.684</td>
</tr>
<tr>
<td>clk2_core</td>
<td>3.595</td>
<td>0.000</td>
</tr>
</tbody>
</table>

**Quartus II - D:\altera\71\qdesigns\fir_filter\fir_filter - filtref - [Compilation Report - Setup Summary]**
Generating Detailed Reports

Choose Report Timing (Reports menu) or double-click on Report Timing (Tasks pane).

Select level of detail.

Select where to send output report.

Tcl: report_timing
Custom Reports: report_timing

CLK: 9.572 ns

Path Summary

<table>
<thead>
<tr>
<th>Property</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>From Node</td>
<td>src_reg</td>
</tr>
<tr>
<td>To Node</td>
<td>dst_reg</td>
</tr>
<tr>
<td>Launch Clock</td>
<td>clk_a</td>
</tr>
<tr>
<td>Latch Clock</td>
<td>clk_b</td>
</tr>
<tr>
<td>Data Arrival Time</td>
<td>2.824</td>
</tr>
<tr>
<td>Data Required Time</td>
<td>12.386</td>
</tr>
<tr>
<td>Slack</td>
<td>9.572</td>
</tr>
</tbody>
</table>

Data Arrival Path

<table>
<thead>
<tr>
<th>Total</th>
<th>Incr</th>
<th>RF</th>
<th>Type</th>
<th>Fanout</th>
<th>Location</th>
<th>Element</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.000</td>
<td>0.000</td>
<td>R</td>
<td></td>
<td></td>
<td></td>
<td>launch edge line</td>
</tr>
<tr>
<td>1.995</td>
<td>1.995</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>clock network</td>
</tr>
<tr>
<td>2.083</td>
<td>0.094</td>
<td></td>
<td>UTOC</td>
<td>1</td>
<td>LCFF X38 Y18 N17</td>
<td>src_reg</td>
</tr>
<tr>
<td>2.181</td>
<td>0.099</td>
<td>RR</td>
<td>CELL</td>
<td>1</td>
<td>LCFF X38 Y18 N17</td>
<td>src_regDFFOUT</td>
</tr>
<tr>
<td>2.616</td>
<td>0.527</td>
<td>RR</td>
<td>IC</td>
<td>1</td>
<td>LCCOMB X38 Y20 N16</td>
<td>dst_regFlipFlop</td>
</tr>
<tr>
<td>2.689</td>
<td>0.053</td>
<td>RR</td>
<td>CELL</td>
<td>1</td>
<td>LCCOMB X38 Y20 N16</td>
<td>dst_regDFFOUT</td>
</tr>
<tr>
<td>2.689</td>
<td>0.000</td>
<td>RR</td>
<td>IC</td>
<td>1</td>
<td>LCFF X38 Y20 N17</td>
<td>dst_regDFFOUT</td>
</tr>
<tr>
<td>2.824</td>
<td>0.155</td>
<td>RR</td>
<td>CELL</td>
<td>1</td>
<td>LCFF X38 Y20 N17</td>
<td>dst_reg</td>
</tr>
</tbody>
</table>

Data Required Path

<table>
<thead>
<tr>
<th>Total</th>
<th>Incr</th>
<th>RF</th>
<th>Type</th>
<th>Fanout</th>
<th>Location</th>
<th>Element</th>
</tr>
</thead>
<tbody>
<tr>
<td>10.000</td>
<td>10.000</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>latch edge line</td>
</tr>
<tr>
<td>12.406</td>
<td>2.406</td>
<td>R</td>
<td></td>
<td></td>
<td></td>
<td>clock network</td>
</tr>
<tr>
<td>15.916</td>
<td>0.090</td>
<td></td>
<td>UTOC</td>
<td>1</td>
<td>LCFF X38 Y20 N17</td>
<td>dst_reg</td>
</tr>
</tbody>
</table>
Summary

- TimeQuest provides an easy-to-use tool to verify timing
  - Entering timing constraints
  - Run various timing reports
TimeQuest Support Resources

- Quartus II Handbook Chapters
  - “The TimeQuest Timing Analyzer” (Volume 3)
  - “Switching to the TimeQuest Timing Analyzer” (Volume 3)

- Training & Demonstrations
  - “Validating Performance with the TimeQuest Static Timing Analyzer” (online recording)
Altera Technical Support

- Reference Quartus II software on-line help
- **Quartus II Handbook**
- Consult Altera applications (factory applications engineers)
  - Hotline: (800) 800-EPLD (7:00 a.m. - 5:00 p.m. PST)
- Field applications engineers: contact your local Altera sales office
- Receive literature by mail: (888) 3-ALTERA
- FTP: [ftp.altera.com](ftp.altera.com)
- World-wide web: [http://www.altera.com](http://www.altera.com)
  - Use solutions to search for answers to technical problems
  - View design examples
Programmable I/O delay (Stratix III)

- Stratix III FPGAs offers the Programmable I/O delay feature with 50 ps resolution.
- In addition, this feature is dynamically programmable.
- The user guide and a design example have been published.
## Programmable I/O delay (Stratix III)

### Table 1–1. Dynamic Delay Chain Valid Values  (Part 1 of 2)

<table>
<thead>
<tr>
<th>Delay Chain Type (1)</th>
<th>Available Settings</th>
<th>Step Value (ps) (4)</th>
<th>Minimum Settings Delay Value (ps) (5)</th>
<th>Maximum Settings Delay Value (ps) (5)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input Delay Chain (D1) (7)</td>
<td>16 (2)</td>
<td>50 ps</td>
<td>0 ps</td>
<td>750 ps</td>
</tr>
<tr>
<td>Output Delay Chain 1 (D5) (6), (7)</td>
<td>16 (2)</td>
<td>50 ps</td>
<td>0 ps</td>
<td>750 ps</td>
</tr>
</tbody>
</table>

### Table 1–1. Dynamic Delay Chain Valid Values  (Part 2 of 2)

<table>
<thead>
<tr>
<th>Delay Chain Type (1)</th>
<th>Available Settings</th>
<th>Step Value (ps) (4)</th>
<th>Minimum Settings Delay Value (ps) (5)</th>
<th>Maximum Settings Delay Value (ps) (5)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Output Delay Chain 2 (D6) (6), (7)</td>
<td>8 (3)</td>
<td>50 ps</td>
<td>0 ps</td>
<td>350 ps</td>
</tr>
</tbody>
</table>
## I/O Performance (Stratix III)

LVDS and external memory interfaces support on all I/O banks

<table>
<thead>
<tr>
<th>Interconnect</th>
<th>Stratix III FPGA Definition</th>
<th>Stratix II FPGA</th>
<th>% increase</th>
</tr>
</thead>
<tbody>
<tr>
<td>DDR3</td>
<td>400 MHz</td>
<td>N/A</td>
<td>New!</td>
</tr>
<tr>
<td>DDR2</td>
<td>400 MHz</td>
<td>333 MHz</td>
<td>20%</td>
</tr>
<tr>
<td>QDR II</td>
<td>350 MHz</td>
<td>300 MHz</td>
<td>16%</td>
</tr>
<tr>
<td>QDR II+</td>
<td>350 MHz</td>
<td>N/A</td>
<td>New!</td>
</tr>
<tr>
<td>RLDRAM II</td>
<td>400 MHz</td>
<td>300 MHz</td>
<td>33%</td>
</tr>
<tr>
<td>LVDS</td>
<td>1.25 Gbps</td>
<td>1.0 Gbps</td>
<td>25%</td>
</tr>
<tr>
<td>Dynamic Phase Alignment (DPA)</td>
<td></td>
<td>DPA</td>
<td></td>
</tr>
<tr>
<td>PCI, PCI-X</td>
<td>3.3-V Compatible</td>
<td>3.3-V Compliant</td>
<td></td>
</tr>
</tbody>
</table>

*Stratix III FPGA left and right banks support 300-MHz double data rate (DDR). Stratix III FPGA top and bottom banks support 800-Mbps LVDS input and 500-Mbps output using a 3-resistor network. Pending Characterization
# Cyclone III Benefits

<table>
<thead>
<tr>
<th>Goal</th>
<th>Cyclone III Solution</th>
</tr>
</thead>
<tbody>
<tr>
<td>Robust</td>
<td>Self calibrating for process, voltage, and temperature changes</td>
</tr>
<tr>
<td>Easy Implementation</td>
<td>Push button timing closure</td>
</tr>
<tr>
<td></td>
<td>Half rate solution lowers FPGA side $f_{\text{MAX}}$ requirements</td>
</tr>
<tr>
<td>Flexibility</td>
<td>Altera, third party, or custom controller functions</td>
</tr>
<tr>
<td>Resource efficient</td>
<td>Uses just 5 global clocks from a single PLL</td>
</tr>
<tr>
<td></td>
<td>Cyclone II required 12 global clocks for x72</td>
</tr>
<tr>
<td>High Performance</td>
<td>200 MHz DDR2 on fastest speed grade</td>
</tr>
</tbody>
</table>

*Performance preliminary pending characterization*
Complete System Support

- **Soft IP Manages Self Calibrating Interface to Memory Devices**

- **Altera or Custom Memory Controller**
  - Altera or Custom Memory Controller
  - Autocalibrating PHY Minimizes Effort for Timing Closure

- **External Memory**
  - Memory support & performance
  - Flexible PCB Layout
  - Signal Integrity

Supporting tools & information
- TimeQuest Timing Analyzer Documentation Other

© 2008 Altera Corporation—Public
Altera, Stratix, Arria, Cyclone, MAX, HardCopy, Nios, Quartus, and MegaCore are trademarks of Altera Corporation
109
### Cyclone III External Memory Support

<table>
<thead>
<tr>
<th>Memory Standards</th>
<th>C6 (MHz)</th>
<th>C7 (MHz)</th>
<th>C8 (MHz)</th>
<th>Availability</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Col I/O</td>
<td>Row I/O</td>
<td>Col I/O</td>
<td>Row I/O</td>
</tr>
<tr>
<td>DDR1 SDRAM</td>
<td>167</td>
<td>150</td>
<td>150</td>
<td>133</td>
</tr>
<tr>
<td>DDR2 SDRAM</td>
<td>200</td>
<td>167</td>
<td>167</td>
<td>150</td>
</tr>
<tr>
<td>QDRII SRAM</td>
<td>167</td>
<td>150</td>
<td>150</td>
<td>133</td>
</tr>
</tbody>
</table>

- All numbers are minimum frequencies achievable; maximum frequencies pending characterization.
ALTMEMPHY Physical Interface

- Soft megafunction included with all versions of the Quartus II software
- Self-calibrating at startup:
  - FPGA and memory device process changes
  - System uncertainties
- Periodic calibration during operation
  - Voltage and temperature changes
- Push button timing closure
- Better performance for fast and slow speed grade devices
Timing Margins Are Shrinking

Align Clock
Capture Phase Here

DQS

DQ (First Data Valid)

DQ (Last Data Valid)

Data Valid at Memory

Data Valid at FPGA

Board Uncertainties

Setup & Hold Time

Internal Skew between DQS & DQ

Total Timing Margin

© 2008 Altera Corporation—Public
Altera, Stratix, Arria, Cyclone, MAX, HardCopy, Nios, Quartus, and MegaCore are trademarks of Altera Corporation
Additional Uncertainties

- Temperature and voltage changes
- Process variations over time
- Memory vendor changes
- Board layout changes

Data Valid at FPGA

Setup & Hold Time

Internal Skew between DQS & DQ

Total Timing Margin
PLL Dynamic Phase Adjustment

- Dynamic adjustment of PLL phase setting
- Increase/decrease 1 step at a time
  - Step increments depend on PLL configuration

In clk0

Phasecounter[3:0]

Phaseupdown

Phasestep

PLL

C0

Phasedone

Locked
Calibration at Startup

Write Training Pattern

Set PLL Phase

Read DQ and Compare to Training Pattern (repeat for all pins)

Set Optimum Clock Phase

PHY adapts to your system!
Periodic Calibration

- Data capture and measurement of a representative mimic path delay every 128 ms
  - Path delay may change due to voltage and temperature changes
  - Assumption: Data Valid window drift due to temperature and voltage similar to delay change of representative path
- Non-Intrusive dynamic phase adjustment
Less Uncertainty = Higher Performance

- Mimic path compensates for clock delay changes inside of FPGA due to voltage and temperature changes
- Benefit:
  - Less uncertainty in timing margin analysis
Altera Memory Controllers

![DDR2 SDRAM High Performance Controller](image)

- **Device family**: Cyclone III
- **Speed grade**: 8
- **FLL reference clock frequency**: 200 MHz (5000 ps)
- **Memory clock frequency**: 200 MHz (5000 ps)
- **Local interface clock frequency**: 100 MHz (10000 ps)
- **Local interface width**: 32 bits

**Memory Presets**

- JEDEC DDR2-400 256Mb x8
- JEDEC DDR2-400 256Mb x16
- JEDEC DDR2-533 256Mb x8
- JEDEC DDR2-533 256Mb x16
- Infineon HYB18TS1250AC-3.7
- Infineon HYB18TS1250AC-3.7

Selected memory preset: JEDEC DDR2-400 256Mb x8

© 2008 Altera Corporation—Public

Altera, Stratix, Arria, Cyclone, MAX, HardCopy, Nios, Quartus, and MegaCore are trademarks of Altera Corporation
MegaCore Ease of Use

- Configures Altera controller and physical interface megafunction PHY

External memory simulation models available from individual vendors
Full Rate Controller

Cyclone III FPGA

<table>
<thead>
<tr>
<th>Memory Controller IP</th>
<th>ALTMEMPHY</th>
<th>FIFO</th>
<th>External Memory</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>16</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>8</td>
<td></td>
</tr>
</tbody>
</table>

PLL

SDR
200 MHz

DDR
200 MHz
Half Rate Controller

- Simplify design requirements by halving application side frequency and doubling data width
- Example: 75MHz Nios II core operating with 150 MHz DDR2 memory

Cyclone III FPGA

- Memory Controller IP
- ALTMEMPHY
- PLL
- HDR 100 MHz
- DDR 200 MHz

External Memory

Example: 75MHz Nios II core operating with 150 MHz DDR2 memory
Flexibility: Separate PHY & Controller

- Customers and third-party intellectual property (IP) providers can design specialized controllers
- Altera ALTMEMPHY Physical Interface retained
TimeQuest – Timing Analysis Tool

- Native support for industry-standard SDC timing constraints
  - Synopsys Design Constraints
- Easily constrain source synchronous interface
  - Constrain data with respect to clock
Signal Integrity

- **Series On-Chip Termination**
  - Match output driver impedance to trace impedance
  - Calibrated on power up for process, temperature, and voltage variations

- **Adjustable Slew Rates**
  - Choose slower slew rates to lower simultaneously switching output (SSO) effects

- **IBIS models for board simulation**
  - Pending characterization
### Flexible PCB Layout

- Interfaces available on all sides

<table>
<thead>
<tr>
<th>Device</th>
<th>Package</th>
<th>Side</th>
<th>Number of ×8 Groups</th>
<th>Number of ×9 Groups</th>
<th>Number of ×16 Groups</th>
<th>Number of ×18 Groups</th>
<th>Number of ×32 Groups</th>
<th>Number of ×36 Groups</th>
</tr>
</thead>
<tbody>
<tr>
<td>EP3C40</td>
<td>484-pin FineLine BGA</td>
<td>Left</td>
<td>4</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Right</td>
<td>4</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Top</td>
<td>4</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Bottom</td>
<td>4</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
## Documentation & Resources

<table>
<thead>
<tr>
<th>Document Type</th>
<th>Title</th>
</tr>
</thead>
<tbody>
<tr>
<td>Handbook Chapter</td>
<td>External Memory Interfaces in Cyclone III Devices</td>
</tr>
<tr>
<td>Application Notes</td>
<td>- Designing for DDR/2 in Cyclone III Devices</td>
</tr>
<tr>
<td></td>
<td>- Timing Analysis for External Memories</td>
</tr>
<tr>
<td>User Guide</td>
<td>High Performance Memory Controllers ALTMEMPHY</td>
</tr>
<tr>
<td>Development Kits</td>
<td>Cyclone III FPGA Starter Kit (DDR)</td>
</tr>
<tr>
<td></td>
<td>Cyclone III Development Kit (DDR2)</td>
</tr>
</tbody>
</table>
Any question?