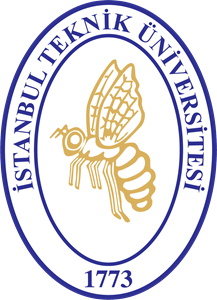
**DIGITAL SYSTEM DESIGN APPLICATIONS**

**(CRN: 11275)**

**THE REPORT OF EXPERIMENT – 6**



Faculty of Electrical and Electronics Engineering

Electronics and Communication Engineering

Yusuf Tekin – 040200043

1. **Clock Generation**

Clock Generation Verilog Code

`timescale 1ns **/** 1ps

**module** top\_module**(**

**input** clk**,**

**input** rst**,**

**output** **reg** **[**6**:**0**]** cnt100**,**

**output** **reg** **[**6**:**0**]** cnt80**,**

**output** **reg** **[**6**:**0**]** cnt60

**);**

**wire** clk80**,** clk60**;**

// Clocking Wizard instantiation

clk\_wiz\_0 clknetwork **(**

**.**clk\_out1**(**clk80**),**

**.**clk\_out2**(**clk60**),**

**.**reset**(**rst**),**

**.**locked**(),**

**.**clk\_in1**(**clk**)**

**);**

// Counter for 100 MHz clock

**always** **@(posedge** clk **or** **posedge** rst**)** **begin**

**if** **(**rst**)** **begin**

cnt100 **<=** 7'b0**;**

**end** **else** **if** **(**cnt100 **<** 100**)** **begin**

cnt100 **<=** cnt100 **+** 1**;**

**end**

**end**

// Counter for 80 MHz clock

**always** **@(posedge** clk80 **or** **posedge** rst**)** **begin**

**if** **(**rst**)** **begin**

cnt80 **<=** 7'b0**;**

**end** **else** **if** **(**cnt80 **<** 80**)** **begin**

cnt80 **<=** cnt80 **+** 1**;**

**end**

**end**

// Counter for 60 MHz clock

**always** **@(posedge** clk60 **or** **posedge** rst**)** **begin**

**if** **(**rst**)** **begin**

cnt60 **<=** 7'b0**;**

**end** **else** **if** **(**cnt60 **<** 60**)** **begin**

cnt60 **<=** cnt60 **+** 1**;**

**end**

**end**

**endmodule**

Clock Generation Testbench Code

`timescale 1ns **/** 1ps

**module** top\_module\_tb**;**

**reg** clk**,** rst**;**

**wire** **[**6**:**0**]** cnt100**,** cnt80**,** cnt60**;**

top\_module dut **(**

**.**clk**(**clk**),**

**.**rst**(**rst**),**

**.**cnt100**(**cnt100**),**

**.**cnt80**(**cnt80**),**

**.**cnt60**(**cnt60**)**

**);**

**always** **#**5 clk **=** **~**clk**;**

**initial** **begin**

clk **=** 0**;** rst **=** 1**;** **#**15**;**

rst **=** 0**;** **#**10**;** rst **=** 1**;**

**#**10**;** rst **=** 0**;**

**#**4200**;**

$finish**();**

**end**

**endmodule**

A black and green line

Description automatically generated

Clock Generation Behavioral Simulation

A screenshot of a computer

Description automatically generated

Clock Generation Primitive Usage

A screenshot of a computer program

Description automatically generated

Clock Generation Clock Networks Report

1. **Clock Gating**

Clock Gating Verilog Code

`timescale 1ns **/** 1ps

**module** clock\_gating**(**

**input** clk**,**

**input** rst**,**

**output** **reg** **[**6**:**0**]** cnt50**,**

**output** **reg** **[**6**:**0**]** cnt25

**);**

**reg** clk50\_en**,** clk25\_en**;**

**wire** clk50**,** clk25**;**

**integer** i**;**

BUFR **#(**

**.**BUFR\_DIVIDE**(**"2"**),**

**.**SIM\_DEVICE**(**"7SERIES"**)**

**)**

BUFR\_ins\_1 **(**

**.**O**(**clk50**),**

**.**CE**(**clk50\_en**),**

**.**CLR**(**1'b0**),**

**.**I**(**clk**)**

**);**

BUFR **#(**

**.**BUFR\_DIVIDE**(**"4"**),**

**.**SIM\_DEVICE**(**"7SERIES"**)**

**)**

BUFR\_inst\_2 **(**

**.**O**(**clk25**),**

**.**CE**(**clk25\_en**),**

**.**CLR**(**1'b0**),**

**.**I**(**clk**)**

**);**

**always** **@(posedge** clk50 **or** **posedge** rst**)** **begin**

**if(**rst**)** **begin**

cnt50 **<=** 7'b0**;**

**end** **else** **if(**cnt50 **<** 50**)** **begin**

cnt50 **<=** cnt50 **+** 1**;**

**end**

**end**

**always** **@(posedge** clk25 **or** **posedge** rst**)** **begin**

**if(**rst**)** **begin**

cnt25 **<=** 7'b0**;**

**end** **else** **if(**cnt25 **<** 25**)** **begin**

cnt25 **<=** cnt25 **+** 1**;**

**end**

**end**

**always** **@(posedge** clk **or** **posedge** rst**)** **begin**

**if** **(**rst**)** **begin**

clk50\_en **<=** 1'b0**;**

clk25\_en **<=** 1'b0**;**

**end** **else** **begin**

**if** **(**cnt50 **<** 50**)**

clk50\_en **<=** 1'b1**;**

**else**

clk50\_en **<=** 1'b0**;**

**if** **(**cnt25 **<** 25**)**

clk25\_en **<=** 1'b1**;**

**else**

clk25\_en **<=** 1'b0**;**

**end**

**end**

**endmodule**

Clock Gating Testbench Code

`timescale 1ns **/** 1ps

**module** clock\_gating\_tb**;**

**wire** **[**6**:**0**]** cnt50**,** cnt25**;**

**reg** clk**,** rst**;**

clock\_gating dut**(**

**.**clk**(**clk**),**

**.**rst**(**rst**),**

**.**cnt50**(**cnt50**),**

**.**cnt25**(**cnt25**)**

**);**

**always** **#**5 clk **=** **~**clk**;**

**initial** **begin**

clk **=** 1'b0**;** rst **=** 1'b0**;** **#**100**;**

rst **=** 1'b1**;** **#**10**;** rst **=** 1'b0**;**

**#**4200**;**

$finish**();**

**end**

**endmodule**

A screenshot of a computer

Description automatically generated

Clock Gating Behavioral Simulation

A screenshot of a computer screen

Description automatically generated

Clock Gatin Primitive Usage

A screenshot of a computer

Description automatically generated

Clock Gating Clock Network Report

1. **Block RAM**

Block RAM Verilog Code

//Copyright 1986-2022 Xilinx, Inc. All Rights Reserved.

//Copyright 2022-2024 Advanced Micro Devices, Inc. All Rights Reserved.

//--------------------------------------------------------------------------------

//Tool Version: Vivado v.2024.1 (win64) Build 5076996 Wed May 22 18:37:14 MDT 2024

//Date : Mon Dec 9 11:36:38 2024

//Host : DESKTOP-KMJ9JAC running 64-bit major release (build 9200)

//Command : generate\_target block\_ram\_50MHz\_wrapper.bd

//Design : block\_ram\_50MHz\_wrapper

//Purpose : IP block netlist

//--------------------------------------------------------------------------------

`timescale 1 ps **/** 1 ps

**module** block\_ram\_50MHz\_wrapper

**(**addra\_0**,**

clk\_100MHz**,**

dina\_0**,**

douta\_0**,**

wea\_0**);**

**input** **[**3**:**0**]**addra\_0**;**

**input** clk\_100MHz**;**

**input** **[**7**:**0**]**dina\_0**;**

**output** **[**7**:**0**]**douta\_0**;**

**input** **[**0**:**0**]**wea\_0**;**

**wire** **[**3**:**0**]**addra\_0**;**

**wire** clk\_100MHz**;**

**wire** **[**7**:**0**]**dina\_0**;**

**wire** **[**7**:**0**]**douta\_0**;**

**wire** **[**0**:**0**]**wea\_0**;**

block\_ram\_50MHz block\_ram\_50MHz\_i

**(.**addra\_0**(**addra\_0**),**

**.**clk\_100MHz**(**clk\_100MHz**),**

**.**dina\_0**(**dina\_0**),**

**.**douta\_0**(**douta\_0**),**

**.**wea\_0**(**wea\_0**));**

**endmodule**

Block RAM Testbench Code

`timescale 1ns **/** 1ps

**module** block\_ram\_50MHz\_wrapper\_tb**;**

**wire** **[**7**:**0**]** douta\_0**;**

**reg** clk\_100MHz**,** wea\_0**;**

**reg** **[**3**:**0**]** addra\_0**;**

**reg** **[**7**:**0**]** dina\_0**;**

**integer** i**;**

block\_ram\_50MHz\_wrapper dut**(**

**.**addra\_0**(**addra\_0**),**

**.**clk\_100MHz**(**clk\_100MHz**),**

**.**dina\_0**(**dina\_0**),**

**.**douta\_0**(**douta\_0**),**

**.**wea\_0**(**wea\_0**)**

**);**

**always** **#**5 clk\_100MHz **=** **~**clk\_100MHz**;**

**initial** **begin**

clk\_100MHz **=** 1'b0**;** wea\_0 **=** 1'b0**;** **#**500**;**

$display**(**"\n--- Reading Initial Data ---"**);**

**for** **(**i **=** 0**;** i **<** 16**;** i **=** i **+** 1**)** **begin**

addra\_0 **=** i**;**

**#**40**;**

$display**(**"Address %0d: Data = %0d"**,** addra\_0**,** douta\_0**);**

**end**

$display**(**"\n--- Writing School Number in reverse to the RAM ---"**);**

wea\_0 **=** 1'b1**;** **#**90**;**

addra\_0 **=** 4'b0000**;** dina\_0 **=** 8'b00000011**;** **#**60**;**

addra\_0 **=** 4'b0001**;** dina\_0 **=** 8'b00000100**;** **#**60**;**

addra\_0 **=** 4'b0010**;** dina\_0 **=** 8'b00000000**;** **#**60**;**

addra\_0 **=** 4'b0011**;** dina\_0 **=** 8'b00000000**;** **#**60**;**

addra\_0 **=** 4'b0100**;** dina\_0 **=** 8'b00000000**;** **#**60**;**

addra\_0 **=** 4'b0101**;** dina\_0 **=** 8'b00000010**;** **#**60**;**

addra\_0 **=** 4'b0110**;** dina\_0 **=** 8'b00000000**;** **#**60**;**

addra\_0 **=** 4'b0111**;** dina\_0 **=** 8'b00000100**;** **#**60**;**

addra\_0 **=** 4'b1000**;** dina\_0 **=** 8'b00000011**;** **#**60**;**

addra\_0 **=** 4'b1001**;** dina\_0 **=** 8'b00000100**;** **#**60**;**

addra\_0 **=** 4'b1010**;** dina\_0 **=** 8'b00000000**;** **#**60**;**

addra\_0 **=** 4'b1011**;** dina\_0 **=** 8'b00000000**;** **#**60**;**

addra\_0 **=** 4'b1100**;** dina\_0 **=** 8'b00000000**;** **#**60**;**

addra\_0 **=** 4'b1101**;** dina\_0 **=** 8'b00000010**;** **#**60**;**

addra\_0 **=** 4'b1110**;** dina\_0 **=** 8'b00000000**;** **#**60**;**

addra\_0 **=** 4'b1111**;** dina\_0 **=** 8'b00000100**;** **#**60**;**

wea\_0 **=** 1'b0**;** **#**90**;**

addra\_0 **=** 4'b0000**;** **#**80**;**

addra\_0 **=** 4'b0001**;** **#**80**;**

addra\_0 **=** 4'b0010**;** **#**80**;**

addra\_0 **=** 4'b0011**;** **#**80**;**

addra\_0 **=** 4'b0100**;** **#**80**;**

addra\_0 **=** 4'b0101**;** **#**80**;**

addra\_0 **=** 4'b0110**;** **#**80**;**

addra\_0 **=** 4'b0111**;** **#**80**;**

addra\_0 **=** 4'b1000**;** **#**80**;**

addra\_0 **=** 4'b1001**;** **#**80**;**

addra\_0 **=** 4'b1010**;** **#**80**;**

addra\_0 **=** 4'b1011**;** **#**80**;**

addra\_0 **=** 4'b1100**;** **#**80**;**

addra\_0 **=** 4'b1101**;** **#**80**;**

addra\_0 **=** 4'b1110**;** **#**80**;**

addra\_0 **=** 4'b1111**;** **#**80**;**

$display**(**"\n--- Writing Complete ---"**);**

$display**(**"\n--- Reading data from RAM ---"**);**

**for** **(**i **=** 0**;** i **<** 16**;** i **=** i **+** 1**)** **begin**

addra\_0 **=** i**;**

**#**40**;**

$display**(**"Address %0d: Data = %0d"**,** addra\_0**,** douta\_0**);**

**end**

$display**(**"\nBenchmark finished.\n"**);**

$finish**();**

**end**

**endmodule**

A screenshot of a computer

Description automatically generated

Block RAM Simulation TCL Console Log

A screenshot of a computer

Description automatically generated

Block RAM Behavioral Simulation

A screenshot of a computer program

Description automatically generated

Block RAM Primitive Usage

A computer diagram of a computer

Description automatically generated with medium confidence

Block RAM Block Design

1. **FIFO for Clock Domain Crossing**

FIFO Verilog Code

`timescale 1ns **/** 1ps

**module** top\_module**(**

**input** clk**,**

**input** rst**,**

**input** wr\_en**,**

**input** rd\_en**,**

**input** **[**7**:**0**]** din**,**

**output** **[**7**:**0**]** dout**,**

**output** empty**,**

**output** full**,**

**output** overflow**,**

**output** underflow

**);**

**reg** **[**2**:**0**]** counter **=** 3'b0**;**

**wire** wr\_clk**,** rd\_clk**;**

**assign** wr\_clk **=** counter**[**0**];**

**assign** rd\_clk **=** counter**[**1**];**

**always** **@(posedge** clk **or** **posedge** rst**)** **begin**

**if(**rst**)** counter **<=** 3'b0**;**

**else** counter **<=** counter **+** 1**;**

**end**

fifo\_generator\_0 memory1**(**

**.**wr\_clk**(**wr\_clk**),**

**.**wr\_rst **(**rst**),**

**.**rd\_clk**(**rd\_clk**),**

**.**rd\_rst**(**rst**),**

**.**din**(**din**),**

**.**wr\_en**(**wr\_en**),**

**.**rd\_en**(**rd\_en**),**

**.**dout**(**dout**),**

**.**full**(**full**),**

**.**overflow**(**overflow**),**

**.**empty**(**empty**),**

**.**underflow**(**underflow**));**

**endmodule**

FIFO Testbench Code

`timescale 1ns **/** 1ps

**module** top\_module\_tb**;**

**wire** full**,** overflow**,** underflow**,** empty**;**

**wire** **[**7**:**0**]** dout**;**

**reg** **[**7**:**0**]** din**;**

**reg** clk**,** rst**,** wr\_en**,** rd\_en**;**

**integer** i**;**

top\_module dut**(**

**.**clk**(**clk**),**

**.**rst**(**rst**),**

**.**wr\_en**(**wr\_en**),**

**.**rd\_en**(**rd\_en**),**

**.**din**(**din**),**

**.**dout**(**dout**),**

**.**empty**(**empty**),**

**.**full**(**full**),**

**.**overflow**(**overlow**),**

**.**underflow**(**underflow**));**

**always** **#**5 clk **=** **~**clk**;**

**initial** **begin**

clk **=** 1'b0**;** rst **=** 1'b1**;** wr\_en **=** 1'b0**;** rd\_en **=** 1'b0**;** **#**40**;**

rst **=** 1'b0**;** wr\_en **=** 1'b1**;** rd\_en **=** 1'b1**;** **#**80**;**

**for** **(**i **=** 0**;** i **<** 64**;** i **=** i **+** 1**)** **begin**

din **<=** i**;** **#**40**;**

**end**

**#**1200**;**

$finish**();**

**end**

**endmodule**

A screenshot of a computer

Description automatically generated

FIFO Behavioral Simulation

A table with numbers and letters

Description automatically generated

FIFO Primitive Usage

A screenshot of a computer program

Description automatically generated

Clock Network Reports

A screenshot of a computer

Description automatically generated

Clock Interaction Report

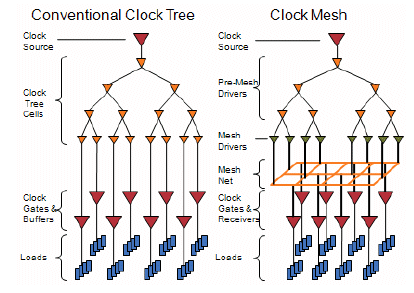
A screenshot of a computer

Description automatically generated

FIFO Post-Implementation Timing Simulation

1. **Research**

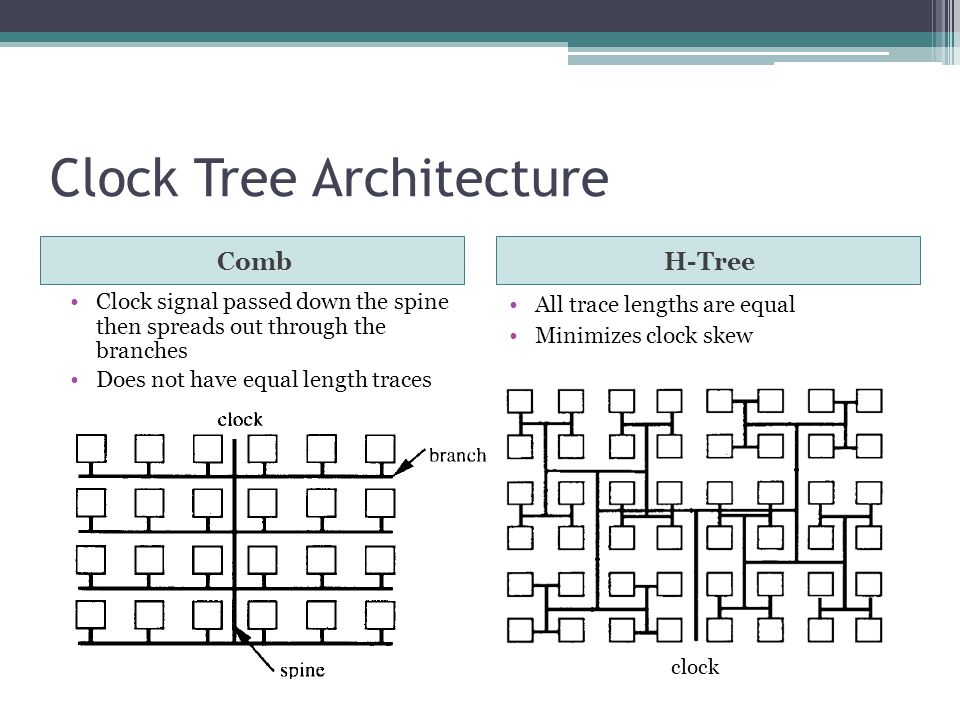
**Clock Tree Synthesis (CTS)** aims to distribute the clock signal from a single clock source to all clocked elements in a design while minimizing skew and maintaining signal integrity.



CTS Figure1

A clock tree is a hierarchical network of buffers, inverters, and interconnect wires that distribute the clock signal. It starts at the clock source and branches out to clock sinks. The topology can be:

* H-Tree: Balanced tree structure for equal latency.
* Spine or Mesh: Grid-based clock distribution, typically used in high-frequency designs.
* Buffered Tree: Uses buffers at branching points to strengthen the clock signal.



Clock Tree Architecture2

**Dual FF Synchronizer:**

Clock Domain Crossing (CDC) occurs when signals pass from one clock domain to another domain with a different clock frequency or phase relationship. CDC introduces the risk of metastability, where a flip-flop enters an unstable state and delays the signal propagation. The **dual flip-flop synchronizer** is a common technique used to address metastability. It ensures that data crossing clock domains stabilizes before it is used in the new clock domain.

A diagram of a circuit

Description automatically generated

Dual FF Figure with Clocks & Outputs3

* **First Flip-Flop:** Captures the signal from the source clock domain and allows metastability to resolve over one clock cycle.
* **Second Flip-Flop:** Further stabilizes the signal, ensuring that it is safe to use in the destination clock domain.
* The two flip-flops act as a buffer, allowing metastability to settle while ensuring signal integrity.

**References**

1. <https://www.design-reuse.com/articles/21019/clock-mesh-benefits-analysis.html>
2. <https://slideplayer.com/slide/4442868/>
3. <https://commons.wikimedia.org/wiki/File:2FF_synchronizer.gif>