Advanced Vlsi Project

Bitcoin

Stage 2 - STA

שמות המגישים	אור שאול	עמרי אורן	סטאניסלאב קוגן	נועה פרקש

Workarea path for Stage2: /project/advvlsi/users/orshaul/ws/bitcoin/stage2/scripts

Warmup Questions

1. Explain what STA is and why it is important in digital circuit design:

Static Timing Analysis (STA) is a method used in digital circuit design to validate that the timing requirements of a system are satisfied, by checking the worst case propagation of all possible paths for min/max delays and it should report if any path violates the max/min delay constraints. It involves examining the timing delays and relationships of signals within a circuit to confirm that the system operates correctly and reliably under specified conditions. In digital circuits, signals travel through various components, each introducing a certain delay. Precise timing is essential for the circuit's proper functioning, as it ensures data reaches its destination within the required timeframe.

STA is crucial in digital circuit design for several reasons:

- **Achieving Timing Closure:** STA helps meet all timing requirements of the circuit.
- **Performance Optimization:** STA can identify bottlenecks, allowing for design improvements that enhance circuit speed.
- **Ensuring Correct Operation:** It helps confirm that the circuit behaves as intended by verifying signal propagation timing.
- **Verification Before Manufacturing:** It serves as a critical step in verifying the design's accuracy before production.
- **Manufacturability:** It aids in ensuring the design can be reliably manufactured by identifying potential issues such as process variations and signal integrity problems that could affect timing.

2. Discuss the limitations of STA and situations where it may not be applicable:

STA has several limitations and may not be suitable in all situations. Some of these limitations include:

- **Static Focus:** It primarily analyses the static behaviour of circuits, overlooking dynamic effects such as glitches, noise, and crosstalk, which can impact signal integrity and timing.
- **Complex Interconnects:** it can be challenging to model delays accurately in complex designs which have complex network connections. STA often uses idealized models that may not fully capture the impact of parasitic capacitance, resistance, and inductance, leading to timing inaccuracies.
- **Idealized Conditions:** STA assumes ideal conditions and doesn't taking into consideration any possible process variations that can happen during manufacturing, which might lead to timing violations in real-world operation.
- **Power and Temperature Considerations:** STA does not consider power consumption or temperature effects, which can introduce additional delays or timing violations due to variations in supply voltage and thermal conditions.

Static Timing Analysis (STA) may not be applicable in certain scenarios, such as:

- **Inability to Handle Asynchronous Designs**: STA is not well-suited for analysing asynchronous circuits, where timing is not solely governed by a global clock. In such designs, the lack of a fixed clock period makes it difficult to apply traditional STA methods effectively.
- **Cannot analyse Combinatorial Feedback Loops**: STA cannot effectively analyse combinatorial feedback loops, such as flip-flops created out of basic logic gates.
- **Inadequate Analysis of Asynchronous Timing Issues**: STA cannot analyse asynchronous timing issues, such as clock domain crossing. These scenarios involve complex interactions between different clock domains that STA cannot accurately assess.
- **Lack of Glitch Detection on Asynchronous Pins**: STA does not check for glitching effects on asynchronous pins. For instance, it cannot detect glitches in combinatorial logic driving asynchronous set/reset pins of sequential elements, potentially leading to overlooked timing issues.

Theoretical Warmup

3.

t_{setup}

Setup time is the time required for the data to arrive to the entrance of the capture (second) flip-flop before the next clock rise to ensure correct sampling. Setup violation occurs when the setup constraint is not met:

Setup equation: $t_{latency,l} + t_{cq} + t_{logic} + t_{setup} < T + t_{latency,c}$

t_{hold}

Hold time is the time required for the data in the entrance of the capture (second) flip-flop to be stable after the clock edge to ensure correct output of the circuit.

Hold equation: $t_{latency,l} + t_{cq} + t_{logic} < t_{hold} + t_{latency,c}$

4. In VLSI timing analysis, slack is the difference between the required time (RAT) and the arrival time (AT) of a signal. Positive slack indicates that the design meets the timing requirements, while negative slack indicates a timing violation. Slack can be calculated for both setup and hold conditions. In Static Timing Analysis (STA), slack is used to determine the timing performance of a circuit. It helps in identifying critical paths, optimizing circuit timing, and ensuring that all timing constraints are met.

5.

The STA algorithm is used to analyse the timing performance of digital circuits. It helps in identifying critical paths, optimizing circuit timing, and ensuring that all timing constraints are met. Here are the inputs and outputs of this algorithm:

Inputs:

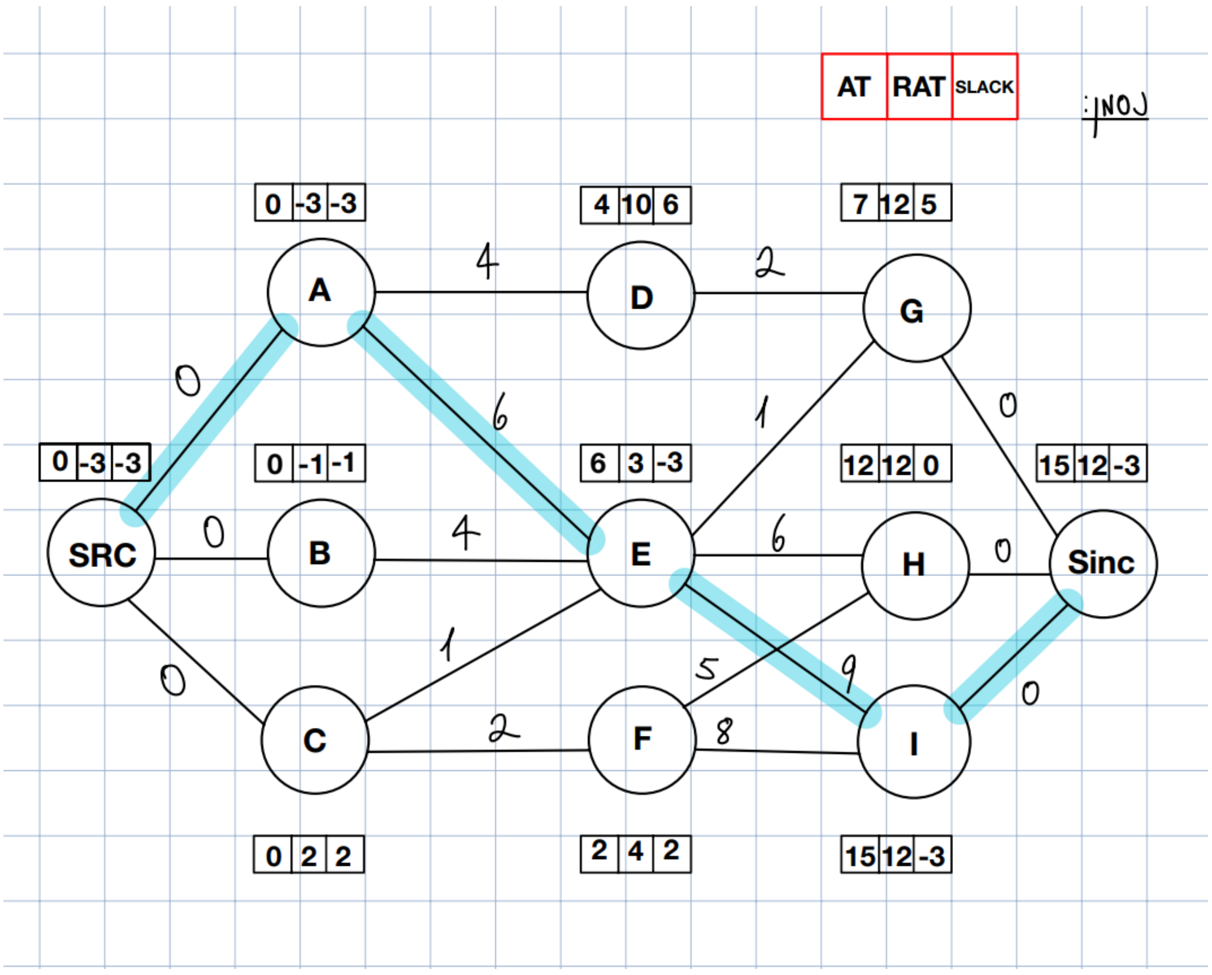
- Circuit netlist: description of the circuit's components and connections
- Timing constraints: such as clock period, setup and hold times
- Delays: interconnect delays (values on arrows between nodes in the graph)

Outputs:

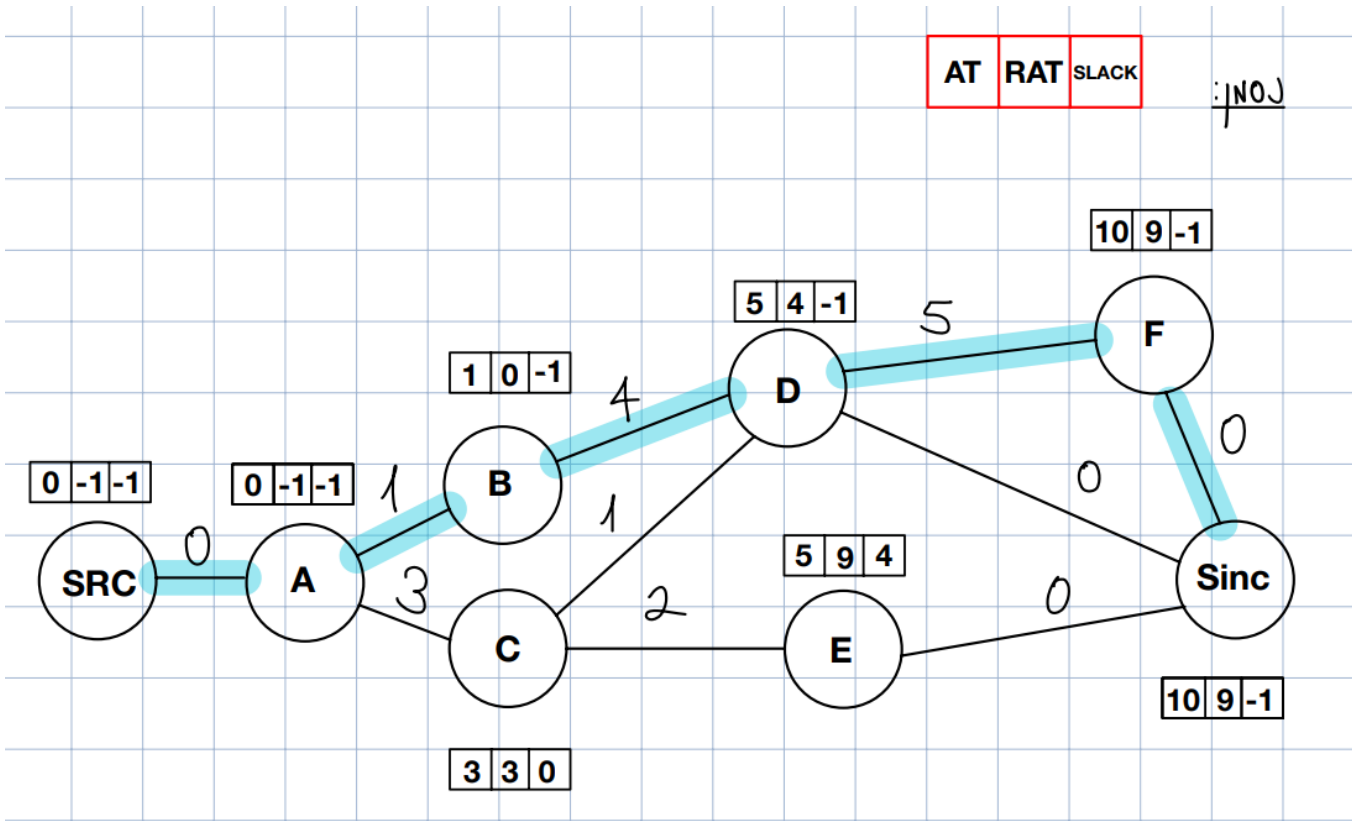
1. Arrival Times: The computed times at which each signal reaches its destination within the circuit.
2. Required Arrival Times: The calculated time windows within which signals must arrive at their destinations to meet the timing constraints.
3. Slack Values: The difference between the required arrival times and the actual arrival times for signals or paths. Slack values indicate the timing margin; negative slack values suggest potential timing violations.
4. Critical Paths: Paths with the smallest slack values, indicating the most critical paths for timing performance. These paths are crucial for determining the overall speed and efficiency of the circuit.

Implementation

Graph 1

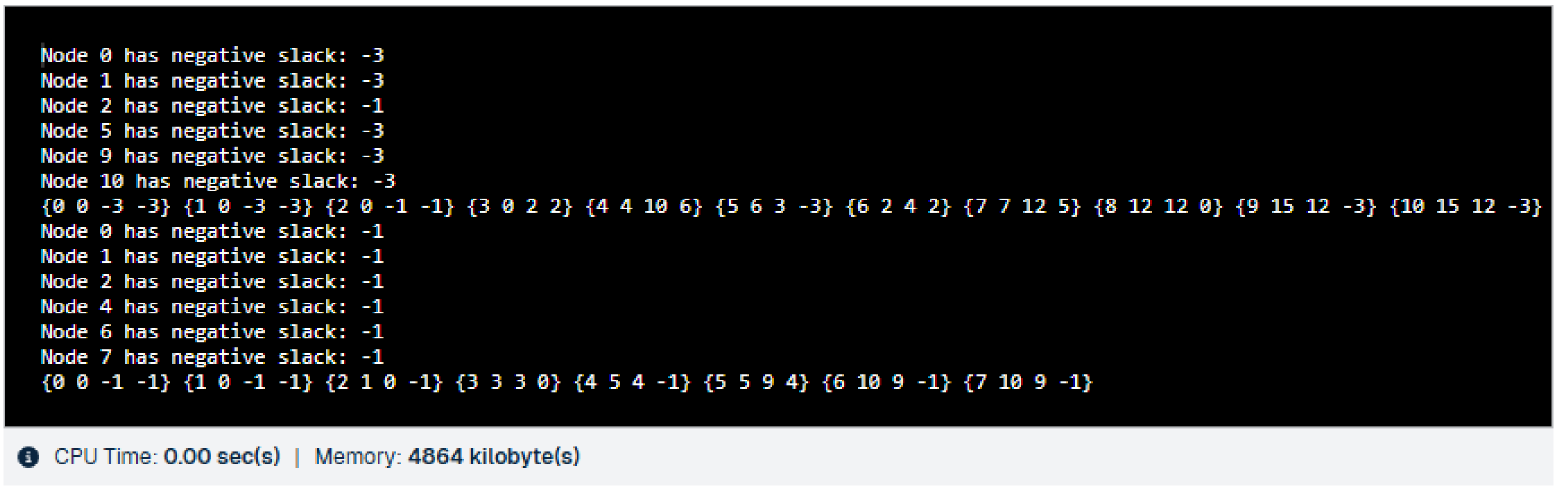


Graph 2



We used the online TCL compiler from the website was mentioned in the PDF instructions: **https://www.jdoodle.com/execute-tcl-online/**

We got identical results between the online compiler output window and the given outputs files OUTPUT_1 And OUTPUT_2. In addition, as seen below we reported on any node which has negative slack as required and we updated the node list with the correct INDEX, AT, RAT, SLACK values.



```
Node 0 has negative slack: -3
Node 1 has negative slack: -3
Node 2 has negative slack: -1
Node 5 has negative slack: -3
Node 9 has negative slack: -3
Node 10 has negative slack: -3
{0 0 -3 -3} {1 0 -3 -3} {2 0 -1 -1} {3 0 2 2} {4 4 10 6} {5 6 3 -3} {6 2 4 2} {7 7 12 5} {8 12 12 0} {9 15 12 -3} {10 15 12 -3}
Node 0 has negative slack: -1
Node 1 has negative slack: -1
Node 2 has negative slack: -1
Node 4 has negative slack: -1
Node 6 has negative slack: -1
Node 7 has negative slack: -1
{0 0 -1 -1} {1 0 -1 -1} {2 1 0 -1} {3 3 3 0} {4 5 4 -1} {5 5 9 4} {6 10 9 -1} {7 10 9 -1}
```

CPU Time: 0.00 sec(s) | Memory: 4864 kilobyte(s)

TCL Code Documentation

Function: calcAT

Purpose:

Calculates the Arrival Time (AT) for each node in the graph. It computes the latest time at which data can arrive at each node, based on the propagation delays.

Parameters:

- adjMat: Adjacency matrix of the graph indicating connections between nodes.
- wMat: Weight matrix of the graph indicating the delay (weight) of edges.
- nodes: List of nodes in the graph, where each node is a list containing [id AT RAT SLACK].

Returns:

The updated list of nodes with calculated AT values.

Function: calcRAT

Purpose:

Calculates the Required Arrival Time (RAT) for each node in the graph. It determines the required latest time data can arrive at each node without violating the timing constraints.

Parameters:

- adjMat: Adjacency matrix of the graph.
- wMat: Weight matrix of the graph indicating the delay (weight) of edges.
- nodes: List of nodes in the graph, where each node is a list containing [id AT RAT SLACK].
- T: The total time budget or deadline for the timing analysis.

Returns:

The updated list of nodes with calculated RAT values.

Function: calcSlack

Purpose:

Calculates the Slack for each node in the graph. Slack is the difference between the Required Arrival Time (RAT) and Arrival Time (AT), indicating the timing margin. It also checks for negative slack, which indicates timing violations.

Parameters:

- adjMat: Adjacency matrix of the graph.
- wMat: Weight matrix of the graph.
- nodes: List of nodes in the graph, where each node is a list containing [id AT RAT SLACK].
- T: The total time budget or deadline for the timing analysis (cycle time).

Returns:

The updated list of nodes with calculated Slack, AT, RAT values.

Function: topologicalSort

Purpose:

Performs a topological sort on the graph, ordering the nodes such that for every directed edge u -> v, node u comes before node v. This is essential for correctly calculating timing properties in acyclic graphs.

Parameters:

- nodes: List of nodes in the graph, each represented as [id AT RAT SLACK].
- adjMat: Adjacency matrix representing the graph structure.

Returns:

List of nodes sorted in topological order, or an error message if a cycle is detected.

Helper Functions for Queue Operations

These functions assist in managing the nodes during the topological sort process.

Function: init_queue

Initializes an empty queue.

Function: enqueue

Adds an element to the queue.

Function: dequeue

Removes and returns the first element from the queue.

Function: is_empty

Checks if the queue is empty.