Assignment 1: Tennis Score Board

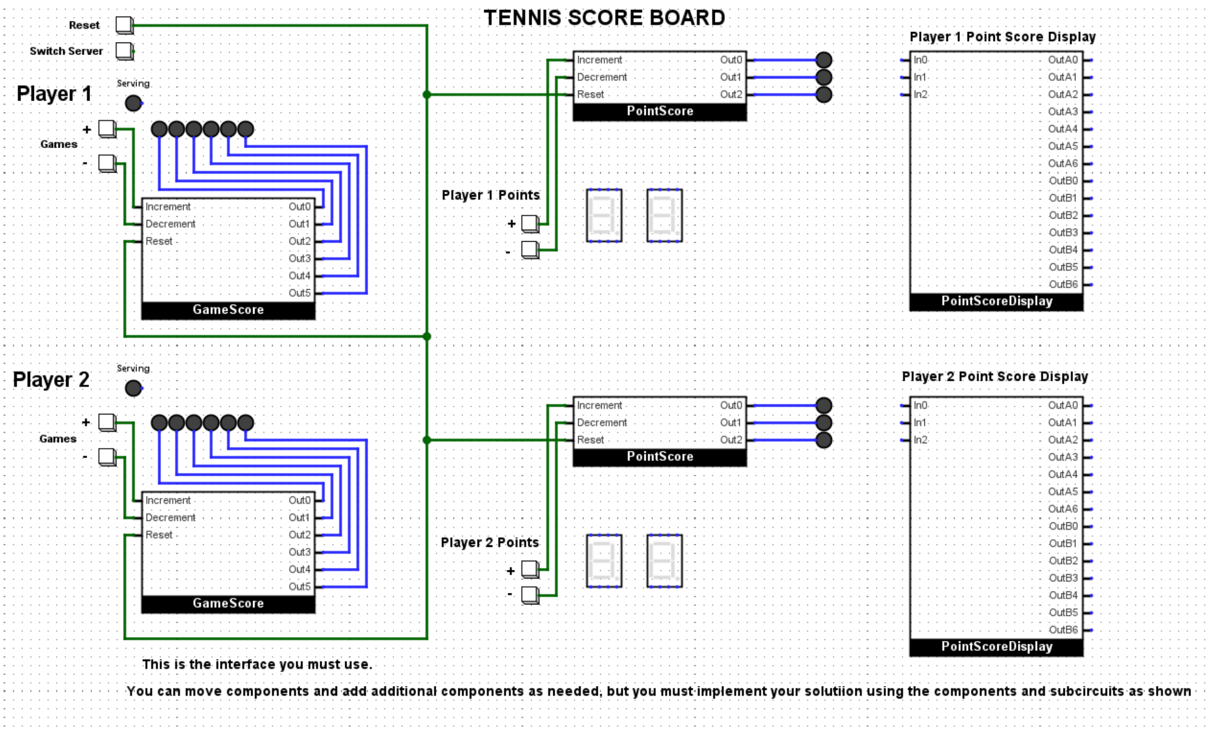
This document outlines all the details for Assignment 1 of COS10004 Computer Systems. It is important that you read it carefully and follow all instructions. If anything is unclear, ask your tutor, your lecturer or post your question on the Assignment 1 Discussion Board.

For this assignment, you are going to implement the functionality for a simple scoreboard for tennis as shown below. For this we are going to simplify things, and not deal with all facets of tennis scoring (e.g., sets, tie-breakers, etc). The scoreboard you will implement (if you complete all stages) will include the following features:

- An indicator keeping track of who is serving
- A game score display that keeps track of how many games have been won by each player (up to 6)
- A point score display that displays the score for the current game (ie., "0", "15", "30", "40", "Ad")
- Functionality to switch the server, and increment and decrement the game and point score.

Your solution will be developed in stages, each described in detail further on in this document.

The interface and subcircuits you will be using to implement your solution are all provided as a Logisim file named TennisScoreBoard.circ and shown below:


TENNIS SCORE BOARD
Reset
Switch Server
Player 1
Serving
Games
Increment
Decrement
Reset
GameScore
Player 1 Points
PointScore
Player 1 Point Score Display
PointScoreDisplay
Player 2
Player 2 Points
Player 2 Point Score Display
This is the interface you must use.
You can move components and add additional components as needed, but you must implement your solutiion using the components and subcircuits as shown

Your assignment must be built using this file, with circuits added inside the provided subcircuits (where applicable). While you are free to add components to the interface, and move things as you need, *<u>all functionality must be controllable using the above interface components only</u>*. ***Failure to do so will result in no marks for Completeness of Solution.***

Logisim Version

Your assignment must be implemented using Logisim Evolution 3.8.0, which can be downloaded from: https://github.com/logisim-evolution/logisim-evolution/releases

This is the version we will test with, and we will not be using any other version, or making special accommodations. If your solution is incompatible with ours*, it will not be able to be tested and thus will be ineligible for most marks on offe*r.

You can verify that your version is correct by loading the provided TennisScoreBoard.circ file with the interface as shown above.

Allowable Logisim Components

Only the following components may be used to develop your solution (but may or may not be required):

- Logic Gates: any
- Flip Flops: JK, D, S-R, T
- LEDs
- Clock
- 7-segment Display (already provided in interface)
- Buttons (already provided in interface)
- Pins (for connecting circuits)
- Constants (for setting inputs that will not change)

The use of any other components *will be penalised* – in particular, you must not use any pre-built circuits such as registers, shift registers, multiplexers etc).

Implementation Stages

To break the problem down, you will implement the functionality of the scoreboard in stages – in many cases within subcircuits already provided for you. Each stage has a percentage weighting of marks contributing to the overall total of 70% for functionality. You should implement each stage in order, and upon completion of each stage, save your file using the naming convention: stageX.circ.

Evidence of progressive development: Each stage *must obviously build on your previous stages*. If the solution for a later stage does not have obvious links with your previous stages (i.e. the solution to a later stage X contains a different implementation of a previous Stage Y to what was provided as that stage's solution, *then you will be awarded no marks for Completeness of Solution*.

Use the subcircuits: Where a subcircuit is provided and you are instructed to implement your solution within it – you must do this! Solutions that do not conform to the required structure will not receive marks.

The reset button: The interface provides a reset button in the top left corner which is connected to a number of components. Most stages of the assignment, in addition to describing the functionality required, also detail the state the circuit for that stage should default back to when the reset button is clicked. It is therefore important that you handle this for each relevant stage, such that all parts of the system reset correctly when the reset button is clicked.

Simplified score board: Note that the scoreboard is simplified for the purposes of this assignment and does not incorporate all facets of tennis scoring (e.g., sets, tie breakers etc). You just need to implement the solution as per the stages described below. No bonus marks are available for extra functionality.

Give yourself the time you need: This assignment will take a long time to complete in full. You should start early and work methodically through each stage if you are seeking to implement everything. Note however that a good mark for this assignment does not require all stages being complete. The break down of marks for each stage and criteria are given further down in this document.

Getting help: All tutors have designated Help Desk times, which are advertised here. There will also be a dedicated Discussion Board on Canvas, however it is important that you do not share your solutions when asking questions or responding to posts in this forum (or others!).

Stage 1 – Switching Servers:

In tennis, the server changes every game. To keep track of who is serving, your scoreboard will show a single LED indicator next to each player's name (labelled "Serving" on the provided interface) which, when turned on, indicates that the corresponding player is serving.

Your task is to implement the "Switch Server" button, which when clicked, swaps which LED is turned on. By default, Player 1 should be the server (i.e. when the scoreboard is first turned on).

Reset Condition: The server should also return to Player 1 if the Reset button is clicked

Stage 2 - Game Score:

Each player on the scoreboard has a 6-LED display indicating how many games they have won so far in the set. Each time a player wins a game, a new LED (from left-to-right) is turned on such that if the player wins 6 games (i.e., a Set), all 6 LEDs would be turned on. The 6-LED game score is controlled by the "+" and "-" buttons to the left of each display. In this stage we are only concerned with the "+" button, which awards the corresponding player another game (we deal with the "-" button in Stage 5).

Your task is to implement the "+" button using the subcircuit "GameScore" for both players such that the game score works as described above. If the user presses "+" after all the LEDs are turn on, then the display should wrap back to all LEDs being turned off and continue to work as described above.

Reset Condition: When the reset button is pressed, the game score for both players should return to 0 (i.e.., all LEDs off)

Stage 3 – Points Score Index:

In tennis, points in a game are awarded in the following sequence "0, 15, 30, 40, Ad". That is, there are 5 possible scores that may be displayed for a given player (we are ignoring "Deuce" in this assignment). As an intermediate step, you are going to implement a simple counter that increments from 0 to 4 in binary each time the associated "+" button is pressed. This number will be used as an index in Stage 4.

Your task: <u>In the provided subcircuit, “PointScore”,</u> implement a counter that increments values between 0 and 4 every time the associated “+” button is clicked on the main interface (don’t worry about the “-“ button – we will deal with that in Stage 5). The “+” button is already connected to the subcircuit for each player via the “Increment” input pin in the “PointScore” subcircuit. Your task is to implement the circuit that takes this input and keeps track of the count. Your counter should wrap around back to 0 after 4.

The binary output showing the current value of your counter should be communicated back to the main interface via the output pins: “Out0”, “Out1” and “Out2” of the PointScore subcircuit, with “Out0” being the least significant bit.

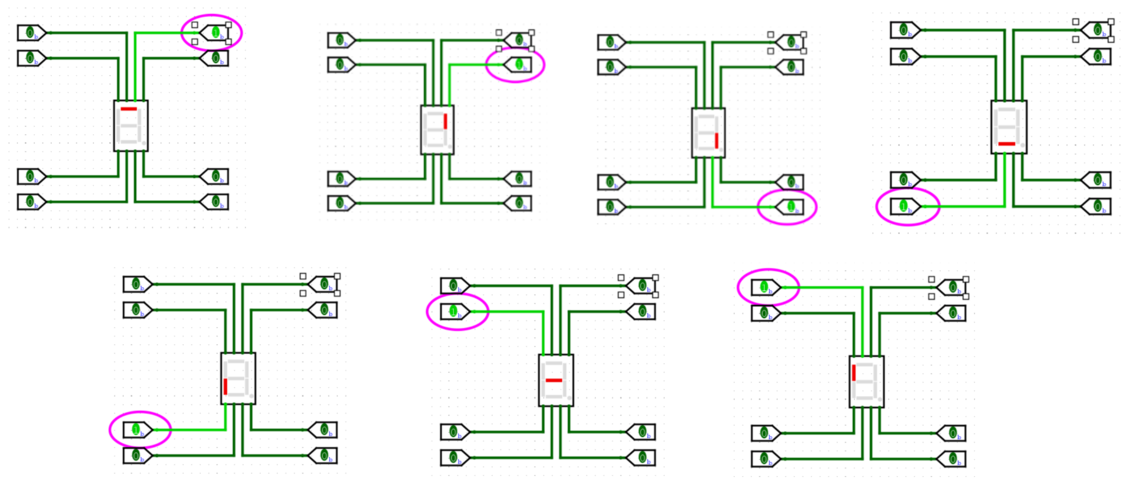
The main interface provides LEDs for each output pin of the subcircuit for you (and us) to see the output of your counter visually - do not remove them!

Reset condition: When the reset button is pressed, the PointScore counter should be reset to “0” for both players.

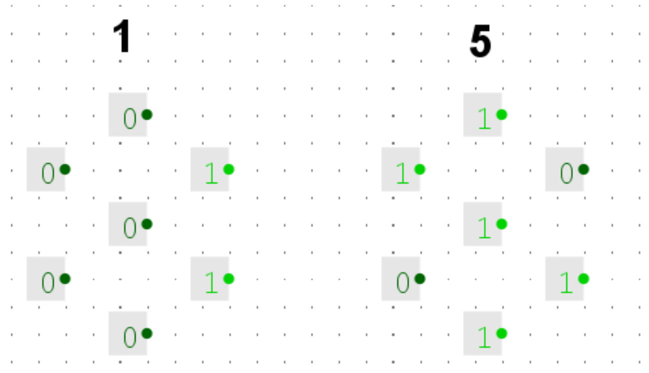
Stage 4 – Points Score Display:

It is now time to implement the main points display for each player. Two 7-segment displays are provided for each player, with each pair to be used to display the actual 2-digit points score (i.e., “00”, “15”, “30”, “40”, “Ad”) for that player.

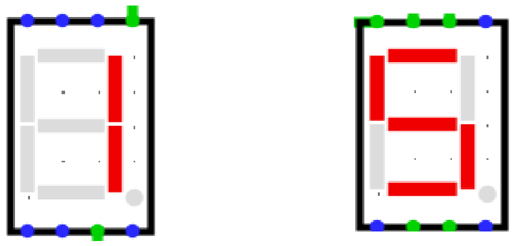
Each segment of the 7-segment display component has a corresponding input pin which, when turned on, activates that segment as shown in the example below.



You will need to feed each 7-segment display the inputs required to light up the required segments for each score. To support this, the subcircuit, PointScoreDisplay has been provided. Inside this subcircuit you will see each displayable score has been encoded as a set of 7 constants (each either on or off), with the layout of constants corresponding to the location of that segment on the 7-segment display. For example, for the score “15”, the constant values are:



which if connected correctly to the 7-segment LED component, would correspond to the following displayed values (note that the input pins in green are the "on" segments):



Other scores are similarly encoded.

There are three input pins to the PointScoreDisplay subcircuit, "In0", "In1" and "In2", which are intended to accept the 3-bit binary number (from Stage 3) indexing each of the displayable numbers such that:

Decimal Index	In0	In1	In2	Output Display
0	0	0	0	"00"
1	1	0	0	"15"
2	0	1	0	"30"
3	1	1	0	"40"
4	0	0	1	"Ad"

Your task is to implement the PointScoreDisplay subcircuit and integrate it with the main interface. That is, implement the PointScoreDisplay subcircuit to accept a 3-bit binary index value, and return the values required to display the associated 2-digit point score on the pair of 7-segment displays for each player as per the above table.

The PointScoreDisplay subcircuit provides output pins labelled: OutA0-OutA, and OutB0-OutB6 for each digit respectively which you can use to send back data to the main interface. You will need to connect these outputs to the appropriate 7-segment display pins in the main interface. You can decide which set maps to which digit!

Reset condition: When the reset button is pressed, the 7-segment displays should both show "00" for both players.

Stage 5 - Decrement Buttons

In Stages 2 and 3 you implemented the "+" button functionality for the Game Score and the Point Score. Of course, user errors can happen, or scores sometimes need to be adjusted, so it is a good idea to have the ability to deduct scores as well.

Your task is to implement the "-" buttons for both the Game Score and the Point Score.

- For the Game Score, a click of the associated "-" button should result in the rightmost "on" LED turning off. If all the LEDs are off then the button click should be ignored (ie., no change to the display).
- For the Point Score, a click of the associated "-" button should result in the displayed point score decrementing to the previous score displayed. Unlike the "+" button however, the counter *should not wrap around* and so once the score reaches "00", any further "-" button clicks should be ignored.

When implementing the above functionality, your solution must also be careful to avoid race conditions, and/or illegal states. Full marks for this stage are only available for solutions that implement the functionality as described and avoid such conditions.

Assessment Criteria:

- Completeness of Solution: (**70% of total mark**)
    - Completion of Stage 1: (up to 10 of the 70 marks available)
    - Completion of Stage 2: (up to 30 of the 70 marks available)
    - Completion of Stage 3: (up to 40 of the 70 marks available)
    - Completion of Stage 4: (up to 60 of the 70 marks available)
    - Completion of Stage 5: (up to 70 of the 70 marks available)
- Quality of the solution (**15% of total mark**)
    - Clarity and modularity of design (including appropriate use of subcircuits)
    - Efficient layout and use of components
    - Readability: use of labels and easy to find/use UI components.
    - Innovation/elegance of solution
- 5-minute video demonstration and reflection (**15% of total mark**)
    - Quality and clarity of the video
    - Depth of understanding and critical reflection on the design and implementation

    Note that your video should be well planned and structured and be strictly no longer than 5 minutes. **A one-mark deduction for every 10 seconds over 5 minutes will be applied.** It should make succinct, well thought out points about your design, and without this, will attract few (if any) marks. You are encouraged to discuss its contents with your tutor.

Submission

Your completed submission must be made through Canvas - (Go to Assignment 1 under "Assignments" before the due date/time).

Everyday day late will incur a *10% deduction*.

Each *submission must be* *<u>zip file</u>* *containing*:

- the actual Logisim files (.circ source files) for testing. Each file MUST be labelled stageX.circ, where X is the stage completed.
- The video file or a file providing a link the file online. Please make sure of you are providing your video as a link, *<u>that the file is not made publicly accessible (i.e.., it should only be accessible to those with the link)</u>*

Academic Integrity

<u>This is an individual assessment task</u> and it is required that you work alone on your solution for assignment<u>. This means any work submitted must be entirely and only the product of your own design and implementation</u>.

This also means you *<u>must not share your solution with any other student</u>*, *<u>or make any part of your solution publicly available available online</u>*.

Markers will be cross-checking work, and will expect to see progress being made on assignments during the dedicated lab classes.

Any breaches of academic integrity will immediately attract a mark of 0 for the assignment, and probable further disciplinary actions in accordance with Swinburne's Academic Integrity policies and procedures.