DIGITAL LOGIC SYSTEMS - SPRING 2019
PROJECT 5: GAME TRACKER (FSM)

Deadline: Sunday 16th June, 2019

The synchronous circuit *game* tracks after the results of the game rounds of two players. At each clock cycle t, two players (player-0 and player-1) submit their round-scores as the inputs $X[0](t)$ and $X[1](t)$, respectively. The game continues until one of the players gains a 2 point advantage relative to his opponent, after which a new game is initialized.

The circuit must output a notification about the winning player: If at time t, player i gains a score which causes him to lead by 2 points, then the output of the circuit must be $y[i](t) = 1, y[1-i](t) = 0$. Pay attention that the output responds immediately to the current input! In all other cases, the output is zero.

Specifications

Inputs: $CLK(t), reset(t) \in \{0,1\}$; $X[1:0](t) \in \{0,1\}^2$
Outputs: $Y[1:0](t) \in \{0,1\}^2$
Functionality: Given a clock cycle t, let $\tau(t) \in \{0,1\}$ indicate whether this is a first clock cycle of a game. Formally

$$\tau(t) = 1 \Leftrightarrow reset(t) = 1 \text{ or } y(t-1) \neq 0^2$$

The outputs $Y[1:0](t)$ must respect the following:

$$Y[0](t) = 1 \Leftrightarrow \sum_{t'=max\{t''|\tau(t'')=0\}}^{t} X[0](t') - X[1](t') = 2$$

$$Y[1](t) = 1 \Leftrightarrow \sum_{t'=max\{t''|\tau(t'')=0\}}^{t} X[1](t') - X[0](t') = 2$$

See example below for the execution of 16 clock cycles

Table 1. Simulation example of `game` synchronous circuit

t	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**
$X[1:0](t)$	01	01	00	10	11	00	10	10	01	01	01	10	00	01	01	00
$reset(t)$	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0
$\tau(t)$	1	0	1	0	0	0	0	1	0	0	0	1	1	0	0	1
$Y[1:0](t)$	00	01	00	00	00	00	10	00	00	00	01	00	00	00	01	00

Your Assignment

The final goal is to implement `game` synchronous circuit, which satisfies the above specifications. Only the Logisim file is to be submitted, however, you are advised to follow the following guidelines

(1) Design the FSM(`game`). Important notice: there is a reset input signal, however you do not need to account for it in your state diagram. Whenever reset(t)=1, it means that the state at time t is your initial state.
(2) Synthesize FSM(`game`) using as few flip-flops and logical gates as you can. Assume that the cost of a flip-flop is 100 and the cost of a basic combinational gate is 1, this means that you should prefer shorter state encodings.
(3) Your implementation must be contained within the provided `template_game.circ` file. Begin with implementing the combinational circuits `Cdelta` and `Clambda`. Then, to complete the design: instantiate `Cdelta` and `Clambda` circuits in the `game` circuit - using the canonical form of a synchronous circuit. Make sure to **modify the initial state string in the existing MUX** (for a proper initialization).
(4) You must not modify the IO ports of the `game` circuit. However, you should modify the data-bits of the S,NS buses and of the register according to the number of bits you use for the state encoding.

Submission Instructions

(1) Submit a single Logisim (".circ") file. No prints/screenshots. This file must be named `ID1_ID2_game.circ` with `ID1` and `ID2` replaced by each partner's 9 digit ID number.
(2) Use the provided `template_game.circ` file as a template, and implement your designs in the circuit named `game`. Do not move or modify the input/output ports, the "blackbox" layout, and the names of the existing circuits!
(3) You are not allowed to create "helper-circuits". All your design must be contained in the `Cdelta`, `Clambda` and `game` circuits. You should modify all the S,NS related IO ports in the provided circuits, but you must not modify the IO ports of the `game` circuit.
(4) Only one of the students in a pair needs to upload the submission. Do not upload the same work twice!
(5) You may not use gates with *fan-in* larger than 2. Exception is for Logisim's MUX2:1, with a fan-in of 3, however is allowed to be used.
(6) You are allowed to use the *data-bits* attribute of the gates to perform bitwise operations.
(7) You are allowed to use "Arithmetic" library of Logisim (Adder,Shifter circuits). as well as the "Plexers" library.
(8) Do not use extra modules from the "Memory" Library. We already placed a register in the `game` circuit. You only need to modify its data-bits attribute.