PPM PROJECT

1. Overview

The ppm project has three different versions. The most complex one is named machine vs. machine : **ppmvsm**. The other two versions of the ppm game at the course web site are : (i) **a** human player playing against a machine player, human vs machine : **ppmhvsm**, and (ii) two human players play against each other, human vs human : **ppmhvsh**. This handout describes the ppmhvsm project. That is, it describes the second version, the human versus machine version in detail. Short descriptions of the other two versions are given at the end.

The ppm players, human and machine, play digits on four position displays to earn points. They play the game until one wins the game. Then, the game is restarted by **resetting**. The players take turns to play : Player 1 plays, then Player 2 plays, then, Player 1 plays, etc. Always, Player 1 starts the game when the FPGA is **downloaded** or reset. Player 1 receives a **pseudo-random digit** (simply random digit, RD) to start with. Player 1 may choose to play the digit or skip the play. Whether Player 1 plays or skips, when Player 2 gets its turn it receives a new random digit. It may play the random digit or skip the play. If Player 2 skips the play, Player 1 gets a new random digit and has the same two options as before : play or skip. If a player skips a play **no** points are subtracted. Also, both players are allowed to see the **next two random digits** which can help them plan their moves better. In summary, the game for each player is about playing a random digit on a position display to earn points, more than the opponent.

The random digit is a BCD digit, i.e. it is 0 through 9. Playing the random digit on a position display is either playing it directly (overwriting) or adding it to a display. If it is a direct play, the position is stored the random digit : RD. If it is an addition, the position is added the random digit and the result is placed on the same display. In either case, if the player has an adjacency, i.e. the digit played has an identical neighbor, the player earns more points. A 2-digit code which is not visible to the players and is on the rightmost two displays can help player earn even more points.

The largest value to play on a display is F, i.e. $(15)_{10}$. That is, the sum of RD and the position display value cannot be greater than $(15)_{10}$. If the sum exceeds $(15)_{10}$, this situation is called **display overflow**. If the player plays it, the digit played is the sum minus $(16)_{10}$. For example, if a position has E and RD is 9, after the addition the position is played 14 + 9 - 16 = 7. Then, the display blinks at a high rate, signalling there is a display overflow. If the player has an adjacency, the player earns more than 7 points. If the digit played is identical to the code digit on that position, the player earns additional points. Thus, winning the game is dependent on both chance and thinking.

The ppm circuit is a **digital system**. A digital system consists of digital circuits. Today's digital systems are numerous. Examples of digital systems are microprocessors, computers, DVD players and iPhones. They are also complex. Consequently, special emphasis is given to the coverage of digital systems in computer science and computer engineering curricula. For example, digital systems is a major topic of **CS 2214**, as well as **upper level EE courses**.

2. Black-Box View and the Input/Output Relationship (Operation) of the Ppm

The ppm project is a digital system. Starting this page, the the human versus machine version is described. The black-box view of this ppm is shown in Figure 1. It has 13 inputs and 19 outputs. One of the inputs is a clock signal generated on the board. All the remaining signals are connected to I/O devices such as **switches**, **push buttons**, **7-Segment displays**, and **LED lights** which are used to enter decisions and visualize game situations.

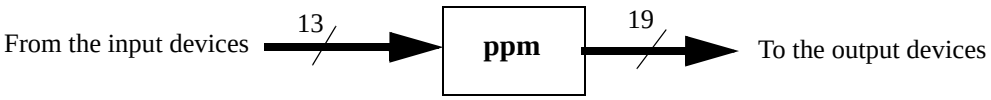

Figure 1. The ppm black box view.

2.1. The Ppm Input/Output Devices

This handout describes the use of the rightmost 4 7-segment displays, rightmost eight LED lights and the rightmost

eight switches of theboard. The other input/output devices can be used by the students to do hardware debugging. The ppm I/O devices used, other than the clock are shown in Figure 2. The detailed view of the ppm inputs and outputs is shown in Figure 3. The description of the inputs and outputs of the ppm are shown on Table 1 and Table 2, respectively. Eight inputs are from the switches and four inputs are from push buttons. 11 outputs are for the 7-segment displays and eight are for the LED lights. One signal that is not connected to an I/O device is the **Clock** signal. It is at 100MHz and used to time and synchronize operations in the digital system. Some of the input and output devices have multiple purposes as explained below.

The switches are used to input decisions and a new random digit. If the switch handle is closer to the human player, it generates logic 0, otherwise it generates logic 1. Switches, SW4-SW7, are the Player 1 display select switches to play the random digit on a display. For example, switch, SW7 selects the leftmost position display, **PD3** ; SW6 selects the next display **PD2**, etc. **SW0** is used to indicate an addition by the human player. If it is 1, it means the human player wants to add the random digit. Finally the rightmost four switches, **SW15** through **SW12** are used to input a random digit for the machine player to speed up the design of the machine player as described below. To indicate that a random digit is being input, one of SW15 - SW12 must be turned on.

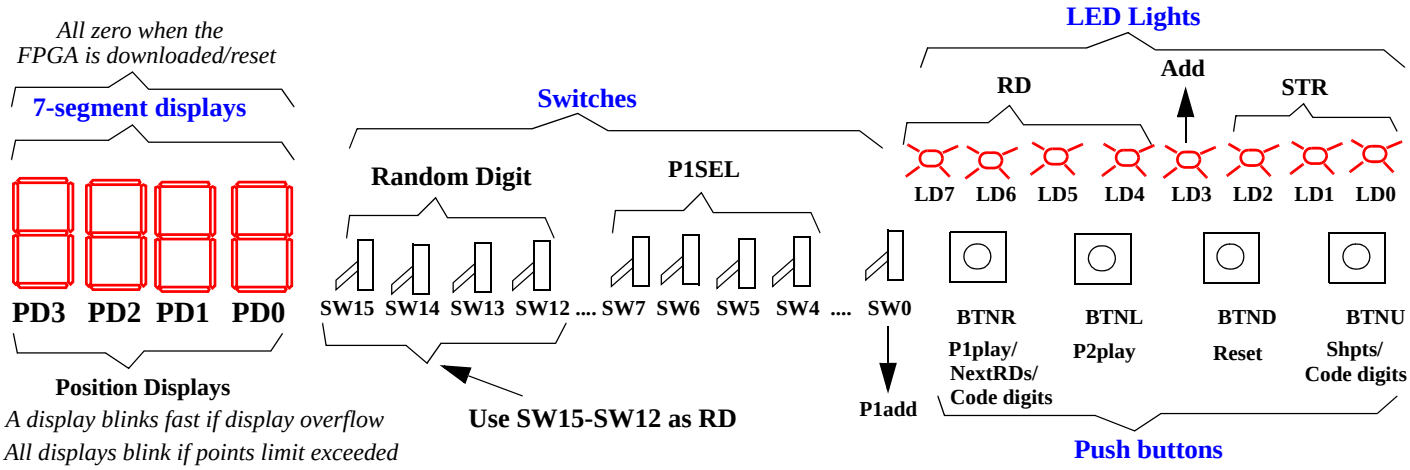

Figure 2. FPGA Board Input/Output device utilization of the ppm.

The four push buttons on the right, BTNR, BTNL, BTNU and BTND, are used by the human player to enter decisions or to get game status. Normally, when they are not pressed, they generate logic 0. As long as they are pressed, they generate logic 1. Push button 3, **BTNR**, has *four* uses. One is to give the turn to Player 1. When it is pressed, it means Player 1 wants to play. Another use is that it is pressed right after reset four (4) times in a row to get three (3) RDs and then to give the first turn to Player 1. The third use of BTNR is that before or after Player 1 plays, it can be pressed to see the next two RDs. The final use is that when pressed together with BTNU, the code digits discovered by the machine player are shown. Push button 2, **BTNL**, is used to give the turn to Player 2. Push button 1, **BTND**, resets the system, clearing all displays and points. When **BTNU** is pressed, players' points are shown on the four displays. When it is pressed together with BTNR, the code digits are shown.

11 outputs are connected to the four 7-segment displays named from left to right, PD3, PD2, PD1 and PD0. The displays show the four position display digits **by default**. They show players' points when BTNU (Shpts) is pressed : (PD1, PD0) show Player 1 points and (PD3, PD2) show Player 2 points. If BTNR pressed when Player 1 has to play, the rightmost two displays show the next two RDs. PD0 shows the next RD and PD1 shows the following RD. If BTNR and BTNU are pressed together, PD1 and PD0 show the code digits discovered by the machine player. All four displays blink if a player wins the game, i.e. when there is a points overflow. The 11 outputs are named **CA** through **CG** and **A4** through **A1**. Outputs, CA - CG, are **active-low**.

Parallel to the push buttons are eight LED lights, LD0 through LD7. All eight LED lights are used to show the game status. The current random digit, RD, is shown on LED lights **LD7 - LD4**. The random digit is a BCD number. LED light 3, **LD3**, emits light if an addition is performed on a display by the current player. LED lights **LD2 - LD0** show the current **state** the ppm system is in. That is, they indicate which step of the game ppm is in. The values are between 0 through 6. The state concept is explained in detail below.

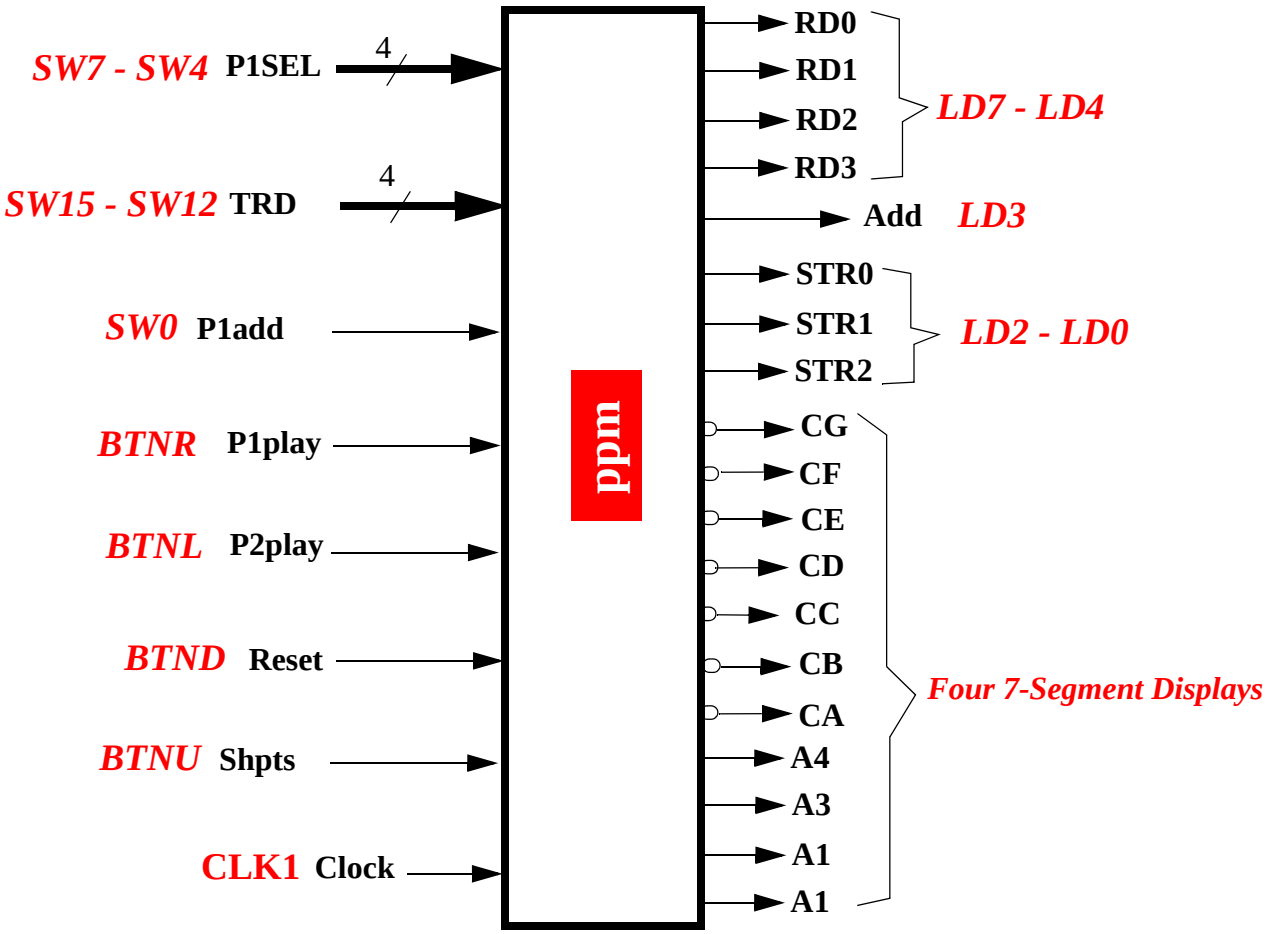

Figure 3. Inputs and outputs of the ppm.

Input device	Description
SW7 - SW4	Player 1 position display select. Each switch selects a position display to play a digit. Also, if one of the switches is 1 when Player 2 gets the turn, a random digit is input from SW3 - SW0
SW15 - SW12	Random digit input. A BCD random digit is input for the player
SW0	Player 1 add. When it is turned on, the current RD is added to the selected position display
BTNR	It is used for four different reasons : (**i**) Player 1 wants to play. When pressed, it means the human player is ready to play ; (**ii**) Pressed four times after reset to get three RDs and allow Player 1 to play ; (**iii**) Pressed before or after Player 1 plays, it shows the next two RDs : PD0 shows the next RD and PD1 shows the following RD ; (**iv**) Pressed together with BTNU, the code digits discovered by the machine player are shown on PD1 and PD0.
BTNL	Player 2 can play. When pressed, it means the human player wants the machine player to play. In three situations it is pressed : (**i**) The human player presses it after playing on a position and wants the machine player to play ; (**ii**) The human player presses it to skip the play, so the machine player plays ; (**iii**) Pressed after the machine player plays with an adjacency.

Table 1: Inputs of the ppm.

Input device	Description
BTND	Reset. When pressed, the game is reset such that all the displays, lights and points are cleared. It is as if the FPGA chip has just been downloaded the bit file. It can be pressed anytime
BTNU	It is used for two reasons : (**i**) Pressed by itself, players' points are shown on the four displays ; (**ii**) Pressed together with BTNR, it shows the code digits discovered by the machine player on PD1 and PD0.
Clock	The clock signal. It is a 100 MHz clock signal generated by an FPGA board circuit

Table 1: Inputs of the ppm.

Output device	Description
LD7 - LD4	Random Digit LED light outputs. The random digit is in BCD. They show the random digit of the current player
LD3	The current player adds. If it is on, it means the current player is adding RD to a position display
LD2 - LD0	The **state** of the game. They indicate where the game stands. The value shown is between 0 and 6. State 0 is the reset state. States 1, 2 and 3 are for Player 1 and states 4, 5 and 6 are for Player 2.
Four 7-Segment displays	By default, the displays show position displays, PD3-PD0, on which random digits are played. If BTNU is pressed, they show Player 2 points in Positional Hex on the left two displays and Player 1 points in Positional Hex on the right two displays. All four displays blink when there is a points overflow. A single display blinks fast if there is a display overflow. If a display is played with an addition and there is no display overflow, it blinks slowly. If BTNR is pressed before and after Player 1 plays, it shows the next two random digits on PD1 and PD0. If BTNR and BTNU are pressed at the same time, PD1 and PD0 show the code digits discovered by the machine player.

Table 2: Outputs of the ppm.

2.2. The Ppm Input/Output Description

The input/output relationship is the operation description which includes the game rules. A good understanding of the input/output relationship is needed to design the ppm digital system correctly. Formally, the input/output relationship should be given by an **operation diagram** since the ppm has sequential circuits. However, in order to simplify the discussion of the ppm system, we will first describe the playing rules in detail and then give the operation diagram.

2.2.1. The Ppm Playing Description

The two players, Player 1 (human) and Player 2 (machine), play until one wins the game. Then, the game is started by pressing the reset button (BTNR). Players take turns : Player 1 plays, then Player 2 plays, then, Player 1 plays, and so on. Always, Player 1 starts the game when the FPGA is downloaded or reset. The game for players is as follows :

- Play a random digit, RD, ($0 \leqslant$ RD $\leqslant (9)_{10}$), directly on display k, PD_k, $0 \leqslant k \leqslant 3$, or add the random digit to a display, PD_k, and store the result on the same display to earn points.
  - Players try to earn points more than the opponent, by looking for **adjacent** identical digits and code digits.
    - After the play, if the player has at least one adjacent identical digit, the player plays again.

- If a player chooses, the play can be skipped to give the turn to the opponent without losing points. A new random digit is given to the opponent. The opponent can play the random digit or skip the play.

2.2.1.1. An overview

After the FPGA chip is downloaded, all four 7-segment displays show zero, since by default, the four 7-segment displays show position displays, i.e. the digits played. Both players start with zero points. All LED lights are blank. If BTNU is pressed the displays show zero as players' points after downloading. BTNR is pressed four times so that human player plays first. The rightmost two displays show the next two RDs when BTNR is pressed again. The game involves guessing and thinking. Players play RD on a display and earn points depending on the value played on the display and the adjacency and code digit situations. The game ends if a player's points exceeds $(255)_{10}$. This points overflow condition is shown by blinking all four displays. The game is restarted by resetting. A player can skip a play, giving the turn to the opponent. The human player can reset the game any time.

When the FPGA is downloaded or reset, a 2-digit **code** is generated on the rightmost two displays, PD1 and PD0 : **Code**. The code is not changed until the game is over. A new code is generated once the FPGA chip is downloaded or the game is reset. The code is hidden from the players. They cannot see its digits. To determine the digits, the players have to analyze the game, especially the points earned after each play. However, playing on the rightmost two displays generate code digit rewards. Playing on the other two positions would not result in code digit rewards.

Once RD is played the reward points is calculated. A reward consists of a regular reward and a code reward. The regular reward depends on the adjacency. If there is no adjacency, the player earns what the display is stored. The player earns more if there are adjacent identical digits : If there is one adjacent digit, the earned points is **two** times the result digit played. Two adjacent identical digits means the player earns **four** times the result digit played. In the best case where there are three adjacent identical digits, the player earns **eight** times the result played. In addition, the player plays one more time. Displays on the two sides are **not** considered adjacent.

If the player plays a code digit on one of the rightmost two displays, the players earns the code reward. If the other display does not have the code digit, the player earns **eight** times the code digit as the code reward. In the best case, if the other display already has the code digit, then player earns the 2-digit code as the code reward.

2.2.1.2. The Ppm Play Sequence and Rules of Playing a Random Digit on Position Displays

This section lists the rules of the game. Although, the critical point in the game is after a random digit is received by a player, below we list all the rules in the order starting from **reset** :

1) When the FPGA chip is downloaded, all position displays, players' points and LED lights are zero. Note that the FPGA chip can be downloaded anytime, for example when the game is going on.

2) When BTNR is pressed four times in a row after downloading or a reset, three RDs are generated and then the human player gets the turn to play and receives a random digit automatically. The random digit is shown on the left-most four LED lights.

3) After receiving the random digit following downloading or a reset, the human player has **two** options : play the digit on a display or skip. If the human player skips, he/she does **not** lose points. Whether the human player plays or skips, the machine player receives a new random digit automatically when it is its turn. The human player can press BTNR to see the next two RDs where PD0 shows next RD (R1D) and PD1 shows the following RD (R2D).

- Playing the random digit on a position display means either playing it directly on a display (overwriting it) or adding the random digit to a display and storing the result there.
    - Playing the random digit directly on a display is the **default** case. To play the random digit directly, the human player turns on a switch that corresponds to the display intended, then turns off the same switch. For example, to play on position 3 (the leftmost position) with direct playing, SW7 (the leftmost switch) is turned on then off ; to play on position 2, SW6 is turned on and off, and so on.
    - In order to add the random digit to a display, the human player first turns on SW0, then turns on the switch that corresponds to the display intended then turns off the position selection switch and then turns off SW0. For example, to play on position 0 (the rightmost position) with an addition, SW0 is turned on, then SW4 is turned on and off and then SW0 is turned off.
- If the human player wants to skip, BTNL is pressed. Since the human player is expected to play, pressing BTNL signals the human player wants the machine player to play, giving the turn to the machine player. The machine player receives a **new** random digit.

4) If the human player plays the random digit, RD, on position display PD_k, reward points are added to the human player points. The earned reward points depend on the digit played, the amount of adjacency and the code digits :

- If the human player plays the random digit on display PD_k directly, the result is $\mathbf{PD_k = RD}$ and the human player earns points which is the sum of regular and code reward points :
    - The regular reward :
        - **No** adjacent RD : RD points
        - **One** adjacent RD : [(2*(RD)) plus another play]
        - **Two** adjacent RD digits in a row : [(4*(RD)) plus another play]
        - **Three** adjacent RD digits in a row : [(8*(RD)) plus another play]
    - The code reward if the digit played is on display 1 or 0 :
        - The other display does **not** have its code digit : [(8*(RD))]

- The other display does **have** its code digit : Code

- If the human player adds the random digit to display PD_k the played display gets the result : **(PD_k + RD)** :
  - If (PD_k + RD) < $(16)_{10}$, the human player earns points which is the sum of regular and code reward points :
    - The regular reward :
      - **No** adjacent (PD_k + RD) : (PD_k + RD)
      - **One** adjacent (PD_k + RD) : [(2*(PD_k + RD)) plus another play]
      - **Two** adjacent (PD_k + RD) digits in a row : [(4*(PD_k + RD)) plus another play]
      - **Three** adjacent (PD_k + RD) digits in a row : [(8*(PD_k + RD)) plus another play]
    - The code reward if the digit played is on display 1 or 0 :
      - The other display does **not** have its code digit : [(8*(PD_k + RD))]
      - The other display does **have** its code digit : Code

  - If (PD_k + RD) $\geq$ $(16)_{10}$, there is a display overflow. The display blinks at a high rate. The human player plays digit (PD_k + RD - $(16)_{10}$), and earns points which is the sum of regular and code reward points :
    - The regular reward :
      - **No** adjacent (PD_k + RD - $(16)_{10}$) : (PD_k + RD - $(16)_{10}$) points
      - **One** adjacent (PD_k + RD - $(16)_{10}$) : [(2*(PD_k + RD - $(16)_{10}$))) plus another play]
      - **Two** adjacent (PD_k + RD - $(16)_{10}$) digits in a row : [(4*(PD_k + RD - $(16)_{10}$))) plus another play]
      - **Three** adjacent (PD_k + RD - $(16)_{10}$) digits in a row : [(8*(PD_k + RD - $(16)_{10}$))) plus another play]
    - The code reward if the digit played is on display 1 or 0 :
      - The other display does **not** have its code digit : [(8*(PD_k + RD - $(16)_{10}$))]
      - The other display does **have** its code digit : Code

After the play, if there is a display overflow, the played position blinks at a high rate. Note that if a 0 is played on a display and there are adjacent 0s, the human player earns 0 points but plays again. Also, if 0 is a code digit and the other display has the other code digit, the player earns Code points.

5) After the human player plays RD, he/she is given a chance to review the situation. For example, the human player can press BTNR to see the next two random digits or BTN4 to see players' points. Then, the human player presses BTNL to give the turn to the machine player. The machine player receives its random digit automatically.

6) The machine player has **two** options : play the random digit on a display or skip. If the machine player skips, it does **not** lose points and the human player receives a **new** random digit. If the machine player plays the random digit, the human player also receives a new random digit. The machine can also see the next two RDs as the human player does. In summary :

- If the machine player plays its random digit, it earns points as described in (4).
- If the machine player skips, it does not lose points. The human player gets a new random digit.

7) After the machine player plays or skips, the human player is given a chance to review the situation and check the players' points by pressing BTNU. Then, the human player presses BTNR to play next. But, if the machine player has an adjacency, the human player presses BTNL to allow the machine player to play. The machine player plays as described in (6) above.

8) The human player has **two** options : play the digit on a display or skip. The human player can press BTNR to see the next two RDs on PD0 and PD1 or BTNR and BTNU to see the discovered code digits. If the human player skips, points is **not** lost. A **new** random digit given to the machine player. Again, we see that :

- If the human player plays its random digit, he/she earns points as described in (4).
- If the human player skips, no points is lost. The machine player gets a new random digit.

9) The game continues in this fashion until one of the players exceeds the points limit, $(255)_{10}$. Then, all four displays blink slowly, the game stops and the player is the winner. The game is restarted from the beginning by pressing BTND (Reset). The game can be reset anytime to restart. When reset, all position displays, players' points and LED lights are zero. The game can be reset even when the game is going on. The machine player **cannot** reset the system.

10) Player points can be seen by pressing BTNU at any time. On the **left** two displays Player 2 points is shown in Positional Hex and on the **right** two displays Player 1 points is shown in Positional Hex. The maximum player points is FF or $(255)_{10}$. The minimum is 0.

Figure 4 gives a number of random digit playing cases, one following another with respect to **time** right after the r**eset**. In this sequence, the machine player is an imaginary one, not the one that is at the course web site.

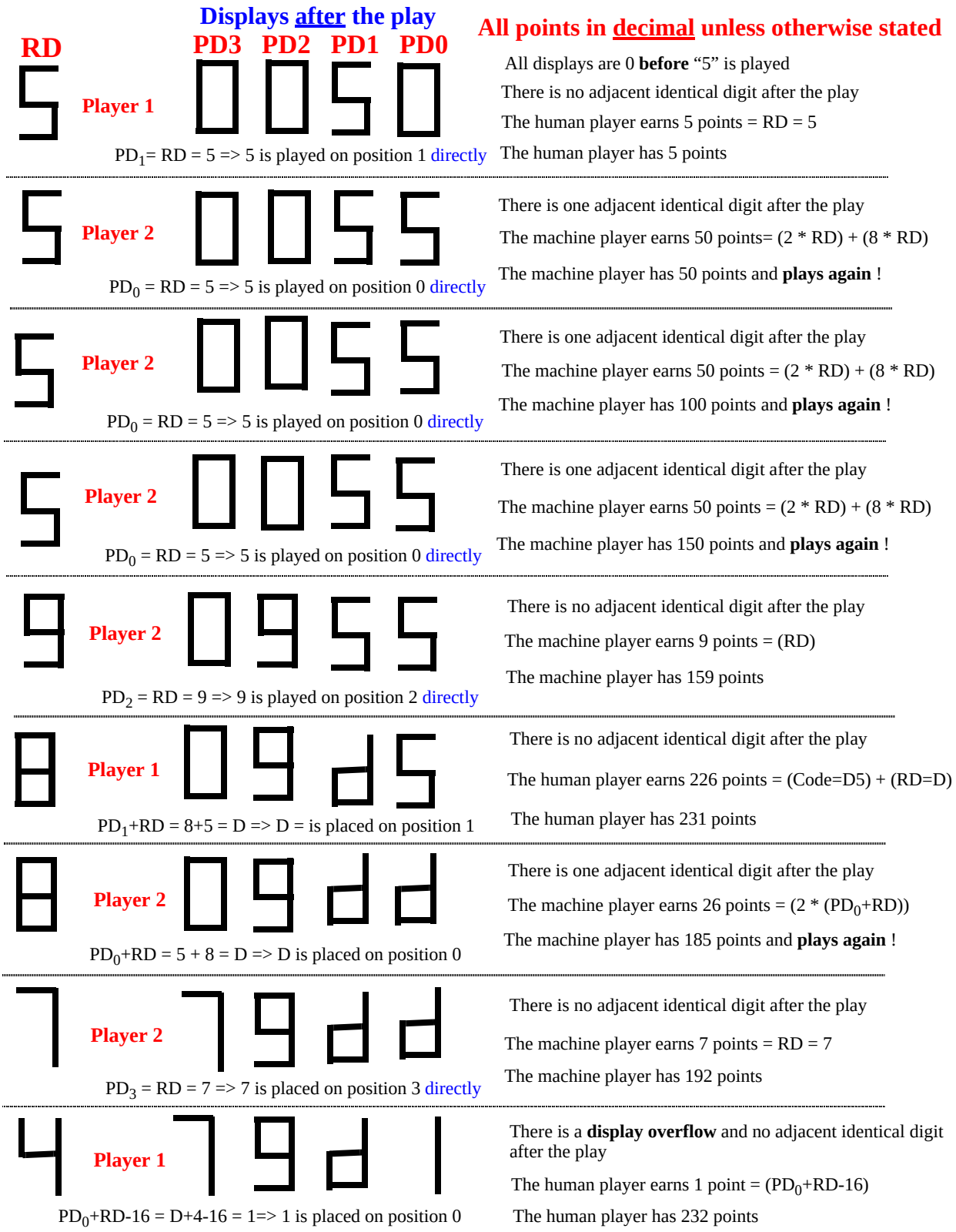

Figure 4. Examples of consecutive digit plays after a reset. Assume that the code is D5

2.2.2. Ppm Playing Modes

The operation diagram of the ppm is shown in Figure 5. The operation diagram shows the operations graphically with respect to time. In order to clearly describe the operations, i.e. which player does what and when, the operation diagram is given in terms of **modes** and **states**. The ppm system is always in a specific mode at any time. There are three modes : the **Reset** mode, the **Player 1 play** mode and the **Player 2 play** mode.

A mode consists of submodes (steps or **states**). A circle is a state in the operation diagram. In each state, the system performs specific operations. Each state takes at least one clock period. To refer to the states easily, numbers are assigned to the states. The assignment is from top to bottom, though, in real life, this is done in a way to reduce the hardware. Each line connects one state to another and is taken if there is a label next to the line and the label is true. For example, if a state has a circled line directed at itself and another line with a label directed at another state, then the system stays in the state until the condition in the label is satisfied. That is, it waits for certain input/event (a push button press/the machine player completes its thinking) indicated by the label to occur to move to the next state. States 0, 1, 3, 4 and 6 are as such. In some other states, there is no label and so a move to another state happens without waiting. Also, certain state transitions change the mode to another mode. In the next section, the operation diagram is used to obtain the major operations and blocks of the ppm. The concept of state is discussed in detail when sequential circuits are covered in class.

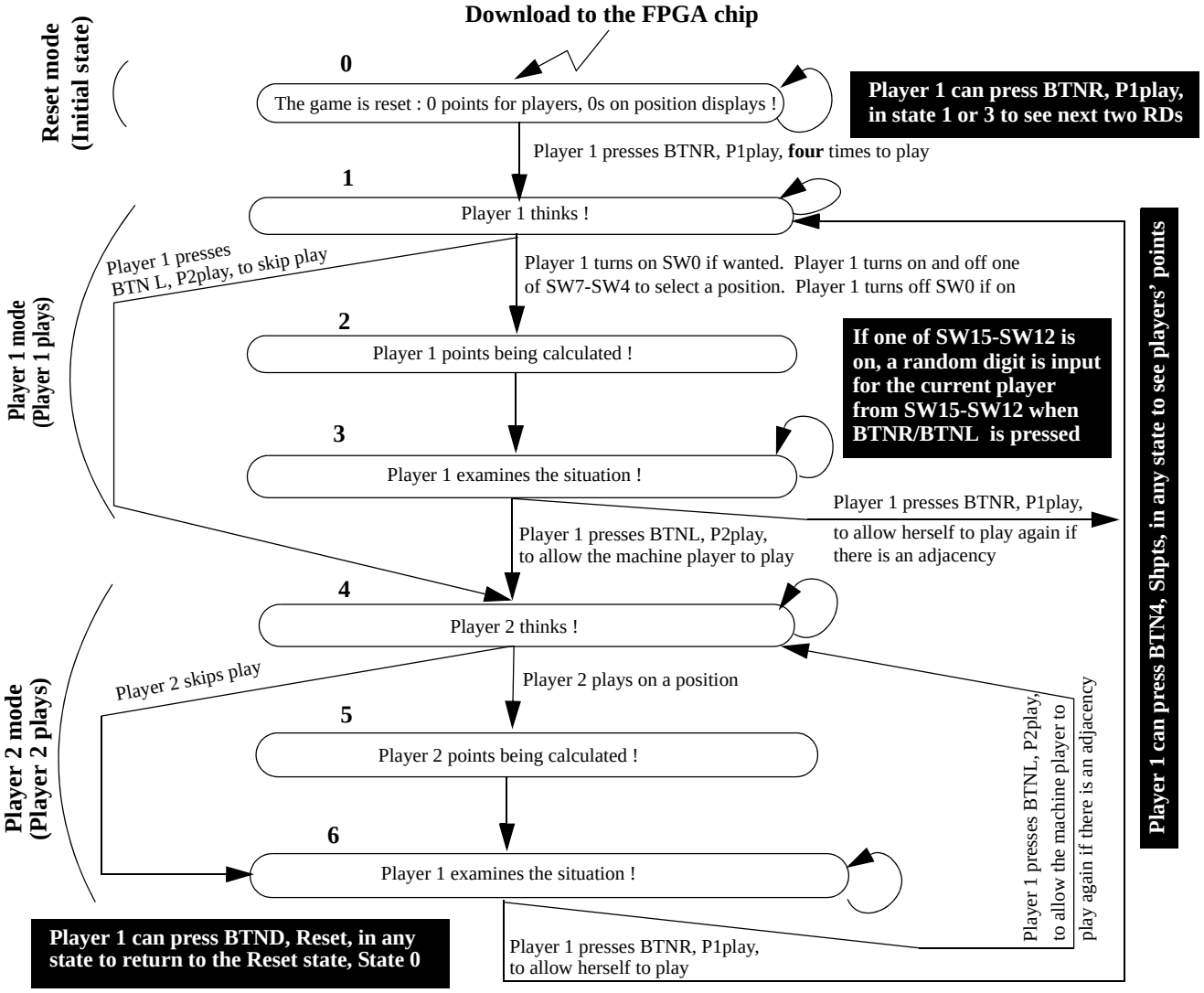

Figure 5. The operation diagram of ppm.

At any moment, the current state number is shown on the rightmost three LED lights : LD2 - LD0 (see Figure 3). Thus, if it is the Reset state, the lights show 0. If the human player is thinking, they show 1, if human player points is

being calculated they show 2 and finally when the human player is examining the situation they show 3. If the machine player is thinking, they show 4, if machine player points is being calculated, they show 5 and when the human player is examining the play again they show 6. These lights are also useful in determining which player has won the game. We know that when a player exceeds $(255)_{10}$, the four displays blink slowly. If BTNU (Shpts) is pressed the points also blink slowly. By looking at the points one cannot determine which player has won the game. If LED lights are checked, they would indicate the winning player : If the lights show 3, the human player has won the game and if they show 6, the machine player has won the game.

3. The Machine Player

The machine player plays against the human player. It is active in state 4 of the ppm operation diagram given in Figure 5. In that state the machine player thinks and makes a decision which is either playing on a position or skipping the play.

3.1. Machine Player Playing Strategies

While exceeding $(255)_{10}$ points before the human player does is the main goal of the machine player, accomplishing it can be done in different ways, i.e. by using different playing strategies. A strategy is a high-level game playing plan that is needed especially when there are alternative plays. The strategy determines how much into the future the machine player can project or how much it can distribute the burden of a single (difficult) play to several plays. Students who are familiar with chess playing can relate to this discussion.

One can think of a number of playing strategies for the ppm game. A strategy that is easy to implement is the **random playing strategy** where the player (i) randomly selects a position and (ii) randomly decides whether to play directly or with an addition. It does not skip. The random playing strategy can result in missed adjacencies and code rewards and so the machine player would lose games against the human player most of the time. A slightly more complex strategy, the **fixed playing strategy** is that the player always plays in a fixed way, regardless of the situation. For example, the player plays in a round robin way on position displays with direct playing then additions, then direct playing again and so on, making its plays very predictable. This strategy would also miss adjacencies and code rewards and lose to the human player many times. One can think of other similar less intelligent machine player strategies. For example, the player checks from left to right which position has a **single** adjacency, and then plays there. If there is no adjacency, the player plays on the rightmost position. This strategy would miss two adjacency cases and code rewards and lose games often.

There are a number of more intelligent playing strategies. Overall, for such strategies, the machine player needs to gather information and then decide what to do. For example, it would gather reward points information for all positions and then decide. In this **points oriented playing strategy** the player always plays on the position that earns the largest regular reward points. The reason why this strategy can be for an intelligent machine player is that the player has to look for regular reward points for all the positions with direct playing and with additions. This is not a simple task. The drawback of this strategy is that looking for largest points at the moment means not thinking about adjacencies that can eventually result in more points soon. Because, we know that not always largest adjacencies mean largest earnings. However, going for adjacencies is favorable since the player is given another chance to play and earn more points while the opponent waits for the turn. Therefore, even a more intelligent player would gather adjacency and regular reward points information for all positions and then decide. But, this intelligent can also lose games since it does not keep track of code digits which can provide additional points besides the regular points.

Another intelligent machine player strategy can be the one that checks regular reward points, adjacencies and also thinks about the future, that is at least the next random digit. This player would try not to play large numbers on displays all the time, unless necessary. By keeping display values low would lead to more adjacency situations before display overflows. This strategy implies that the machine player has different plays for the same combination of display and random digit values. For example, a machine player plays for adjacencies if it is ahead and when it is behind, it plays for largest regular reward points. Again, this machine player would also lose games since it does keep work on the code digits.

A more intelligent machine player checks regular reward points, adjacencies and code digits. We know that the code digits are not visible, but can be determined by game situations. That is, by observing how many points after a play, the player can determine the code digit on that position. For a machine player, it means it has to work not only in state 4 of the ppm operation diagram, but also in states 2 and 5 during which human player and machine player points are calculated. Only in states 2 and 5 the CODERWD bits can be non-zero hence the need to be active in states 2 and 5. As we see below, the course web site machine player uses such a strategy.

Students note that their machine player strategy needs to deal with situations when there are two or more equally good positions to play given an intelligent strategy. For example, if playing on two positions results in the same regular reward points and no code reward, which one will be selected ? One solution is that the fixed strategy is used where always the rightmost of these two positions is selected. Another solution is that one of these two positions is randomly chosen. Therefore, students will have **two** strategies, one **primary** strategy that deals with most of the situations and a **secondary** one that handles equally playable situations.

3.2. The Machine Player Playing Against the Human Player

The machine player strategy is a relatively intelligent one. It thinks ahead and plays so that its earnings currently and in the future are maximized as much as possible. It checks for a number of situations and consequently performs one of six (6) actions when it is its turn. When it encounters a specific case, it decides to skip. In any situation, if there are equally good playing possibilities, it uses the fixed playing strategy where it plays on the rightmost of the playable positions. There is an exception to that in case there is code digit playing possibility. There are other details about the strategy that are shown in Figure 6.

Overall, first the machine player tries to win the game with the current random digit. Otherwise, it tries to keep displays small unless there is a code digit play possibility on displays 1 or 0. If no, it goes for largest adjacencies, not for largest regular reward points when it is ahead. If there are zeros on the displays in the beginning of the game, it plays on zero positions for future adjacencies. When there is no code digit possibility and it is not ahead, it plays for the largest regular reward points. It also checks the next RD (R1D) to see if it is equal to a code digit. Note that the strategy in Figure 6 indicates what the machine player does in states 2, 4 and 5 of the ppm operation diagram.

Let's analyze the course web site machine player strategy in detail. The strategy consists of condition checks and actions. The machine player first determines if it is close to winning the game, a condition. That is, if the current machine player points plus the largest regular reward points it can get exceeds $(255)_{10}$, then it is close to winning.

Therefore, it plays for the most regular reward points, an action. Otherwise, it checks in the order of (i) playing a code digit, (ii) if the next RD is a code digit, (iii) if it is early in the game and (iv) if it is ahead of the human player. If it is ahead, it goes for adjacencies and plays for the future. If the machine player is not ahead, it plays for largest reward points. Below we describe first the actions and then the conditions :

- Action 0 : Play on the (rightmost) largest regular reward points position (directly if equal)
  - The machine player collects regular reward points information for each position for both cases of the position is played directly and with an addition. It selects the largest reward position. If there are several positions with the same largest reward points, it plays on the rightmost of these positions. If playing directly and adding result in the same largest points, it chooses playing directly .
- Action 1 : Play on the (rightmost) largest adjacency position (directly if equal)
  - The machine player collects adjacency information for each position for both cases of the position is played directly and with an addition. It selects the largest adjacency position. If there are several positions with the same largest adjacency, it plays on the rightmost of these largest adjacency positions. If playing directly and adding result in the same largest adjacency, playing directly is chosen.
- Action 2 : Play on the (rightmost) zero position directly
  - The machine player checks each position if it is zero. It selects the position with a zero and plays the random digit directly. If there are several positions with zero, it plays on the rightmost of these zero positions.
- Action 3 : Play on the (rightmost) largest display position directly
  - The machine player checks each position to see which has the largest value. It selects the position with the largest value and plays the random digit directly. If there are several positions with the same largest display value, it plays on the rightmost of these positions.
- Action 4 : Play on the (leftmost) code digit position (with an addition if equal)
  - The machine player checks positions 0 and 1 with direct playing and additions to see if any results in a code digit that it discovered. If yes, it plays on that position. If both positions result in a code digit, it selects the leftmost of these two position. If direct playing and adding gives code digits, then it selects adding.
    - In order to discover the code digits, the Action 4 circuitry checks the value of CODERWD in states 2 and 5 of the ppm operation diagram. We know that the CODERWD is valid only in these two states when the new player points is calculated. If the CODERWD value is non-zero, then the Action 4 circuitry stores the digit played and its position number on two registers.
- Action 5 : Skip the play and give the turn to the opponent.

Conditions :

- Largest regular reward completes game ?
  - Does the sum of the largest regular reward and the current machine player points exceeds $(255)_{10}$?

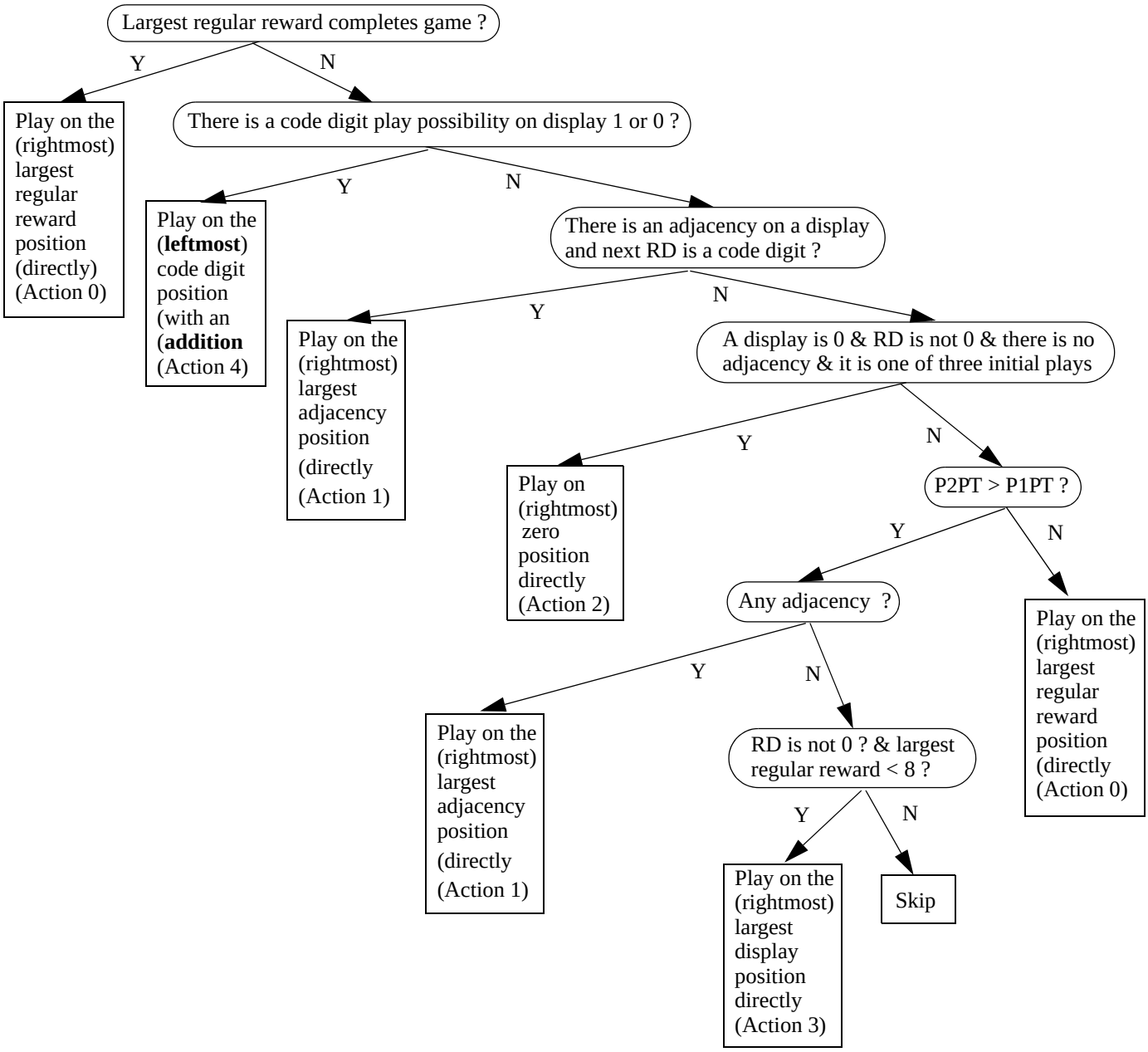

Figure 6. The playing strategy of the machine player, playing against the human player.

- There is a code digit play possibility on display 1 or 0 ?
  - Does playing on position 0 or 1 with direct playing or adding generate a code digit ?
- There is an adjacency on a display and next RD is a code digit ?
  - Does playing the random digit on any position results in an adjacency and the next random digit if played directly is equal to one of the discovered code digits ?
- A display is 0 & RD is not 0 & there is no adjacency & it is one of three initial plays
  - Has the machine player played not more than three times yet (the game has just started) and there is no adjacency on displays and RD is not zero and a display is zero ?
- P2PT > P1PT ?
  - Is the machine player ahead in terms of game points ?

- Any adjacency ?
  - Is there any adjacency on the displays if the machine player plays the random digit either directly or with

an addition ?

- RD is not 0 ? & largest regular reward < 8 ?
  - Is the random digit non-zero and the largest regular reward is less than 8 ?
    - The machine player considers this condition when it is ahead and there is no code digit possibility nor adjacency. Its goal is to reduce the largest display value so that the human player does not earn a lot of points.

A note about keeping the displays small is that if players constantly add to the displays they would earn large points. But, eventually display overflows occur and they earn smaller points. A general playing strategy can be looking for adjacencies and code digit (not for large reward points) to gradually increase the display values to have more adjacency opportunities before display overflows. For example, if the displays are (A644), adding RD = 4 on position 2 would result in one adjacency and $(20)_{10}$ points : (AA44). Playing RD on position 2 directly results in two adjacencies with lower points $(16)_{10}$: (A444). But this second play keeps the displays low so that there would be more adjacency opportunities until there are display overflows. However, looking for adjacencies and code digits requires more thinking effort. For the machine player that means considerable amount of additional hardware.

There is even more thinking effort needed if a player considers next two RDs to earn more points faster. For example, if the current displays are (A644), RD = 4 and next RD = 6, then playing as (AA44) may be better since the player plays again to get another adjacency with RD = 6 : (AAA4). Another example is that the displays can be (FCFE), RD = 2 and next RD = 1, then playing the 2 as (FEFE) would give a good opportunity to the opponent since the 1 can be to get (FFFE). The opponent would have an adjacency situation ! The machine player does not check for these two situations though. The reason is that the FPGA space is exceeded if any other feature is added.

3.2. The Machine Player, Playing Against the Other Machine Player

This ppm version is the one where two machine players play against each other. It is used for the term project. The switches used by the human player SW7-SW4 and SW0 are **not** needed. The human player presses BTNR to allow the first machine player to play and BTNL to allow the second machine player to play. BTND and BTNU are used as before. The rules of the game are identical to the human vs. machine game rules. Both players can be input a random digit from switches SW15-SW12 to debug the two player circuits.

The **Player 2** machine player uses the playing strategy described in Figure 6, since it is playing against the human player. The first machine player implemented has a different playing strategy. It uses the playing strategy shown in Figure 7. Player 1 always seeks for the largest regular reward points. If there are positions with identical largest regular reward points, it plays on the rightmost of these. If playing directly and with an addition results in the same regular largest reward points, playing directly is chosen. It never skips. Overall, Player 1 is much simpler than Player 2, the course web site machine player.

Figure 7. The playing strategy of Player 1 machine player, playing against the other machine player.

Note that machine vs. machine ppm project uses all eight displays and all 16 LED lights. That is, four aditional displays and eight additional LED lights are used compared with the human vs. human and human vs. machine projects :

- Displays 4 and 5 show Player 1 points
- Displays 6 and 7 show Player 2 points
- LED lights 8 to 11 show the next random digit
- LED lights 12 to 15 show the following random digit

3.3. The Human vs. Human Playing Description

This ppm version is the one where two human players play against each other. Both human players use SW7-SW4 to select a position and SW0 to add the random digit. The rules of the game are identical to the human vs. machine game rules. A random digit can be input to either player manually.