



Simulating concept of arbitrage betting using hardware circuits.

0 stars

0 forks








1 watching

Activity

Branches

Tags

Public repository

 I-m-a-o Add files via upload	d8b928b · 10 months ago	
 logisim	Add files via upload	last year
 snapshots	Add files via upload	last year
 verilog	Add files via upload	last year
 videos	Add files via upload	10 months ago
 README.md	Update README.md	10 months ago

Guaranteed Profits with Arbitrage Betting

Team Details

- ▼ Detail

Semester: 3rd Sem B. Tech. CSE
- Section: S2
- Member-1: Hayden Soares, Roll No: 221CS224, email: haydensoares.221cs224@nitk.edu.in
- member-2: Granth Tiwari, Roll No: 221CS220, email: granth.221cs220@nitk.edu.in
- Member-3: Vishal Kamath, Roll No: 221CS261, email: vishalkamath.221cs261@nitk.edu.in

Abstract

- ▼ Detail

Background: Betting is one of the most popular ways of making fast cash. Usually, betting odds offered by bookmakers are always tilted in the favor of the bookmaker (called the house edge): given enough trials, the bookmaker will profit. Thus, the bettor ends up losing.

Arbitrage betting is a strategy that involves betting on all outcomes with distinct bookmakers, to ensure a profit for the bettor, regardless of outcome. This is normally used when different bookmakers significantly disagree on the odds of an event or when they make an error in calculating the odds of the event. This isn't always possible (the probabilities across the outcomes must add to less than 1), yet when so, you're mathematically guaranteed to profit.



Motivation: As students of mathematics, we were motivated to find a unique topic to apply simple principles of mathematics. We also wanted it to be usable by the public, so we chose betting, since it has become very popular. Finally, we wanted to implement a betting strategy that had guaranteed returns, thus being more beginner friendly.

Unique Contribution: We realized that most people who arbitrage bet, do it manually: they surf through different betting sites and painstakingly manually calculate if an arbitrage is possible on hundreds of games. We aim to design a circuit to automate this process.

Approach: The circuit will take inputs representing the odds of an event (win and loss) offered by 2 different betting companies (say Team A win odds from company X and Team A loss odds from company Y) for the two companies. The circuit will then determine whether an arbitrage bet is possible or not. If so, it will be displayed by an LED. Finally, if an arbitrage bet is possible, a board will display the approximate minimum GUARANTEED PROFIT that will be achieved (as a percentage).

Working

▼ Detail

In our design, we accept company odds of the form X.Y i.e a 2 digit decimal number, with one digit before and after the decimal point respectively. The main circuit takes in the two digits separately as two BCD 4 bit numbers (ignoring the decimal point).

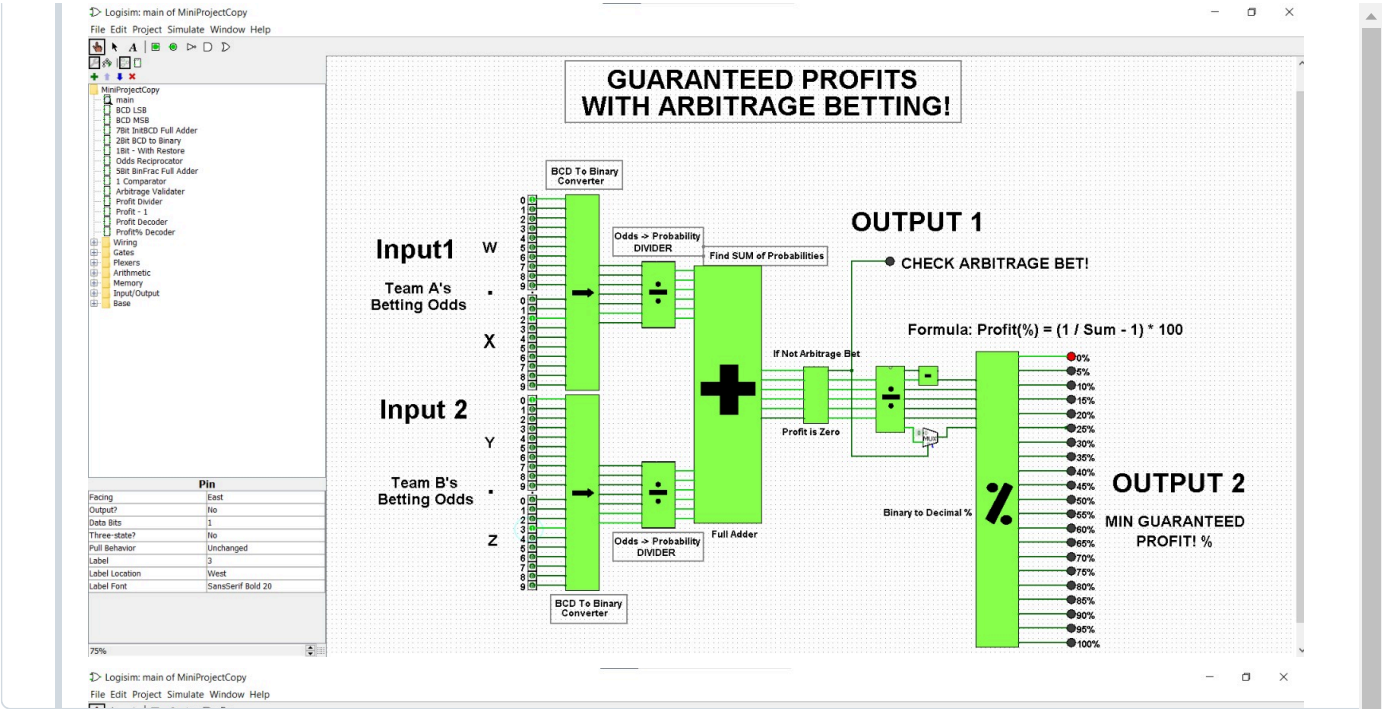
After receiving the two input odds, say W.X and Y.Z (or equivalently, WX and YZ), the first stage of the circuit converts the decimal/BCD input into two 7 bit binary numbers. This is done by mapping the LSB to a 4 bit binary number (like 3 to 0011 or 7 to 0111) and the MSB to 7 bit binary numbers, which represent 2 digit decimal numbers (like 3 to 0011110 or 7 to 1000110). The optimization is done via the Quine - McCluskey Method. Then, a 7 bit binary adder is used on the two binary numbers for each of the corresponding BCD digits to obtain the final binary representations: $M = M_0 \dots M_6$ and $N = N_0 \dots N_6$, for WX and YZ respectively.

In order to check if the bet is an arbitrage bet or not, we use the following formula: $P_1 + P_2 < 1$ (base 2), where P_i represents the probabilities, which is the reciprocal of the odds i.e. $P_1 = 10/M$ and $P_2 = 10/N$. Therefore, in the second stage of the circuit, we perform division of each of the odds from binary number 0001010.00000 (representing 10, with 5 decimal precision) using a custom built 7 by 7 bit divider circuit, using restore operation (credit: ALL ABOUT ELECTRONICS). From this, we obtain two 6 bit binary numbers $M' = M_0' \dots M_5'$ and $N' = N_0' \dots N_5'$, where the MSB represents the 1s place (base 2), and the remaining 5 bits are for bits after the binary point. Then, we add M' and N' using a 5 bit full binary adder circuit. Let the number obtained be $P = P_0 \dots P_5$. Finally, we use a one bit comparator circuit to check if the MSB is 0 or not. If it is 0, then we have an arbitrage bet, and thus the LED will flash ON. Else, the LED will be OFF. Thus, the user will know if it's possible to make a GUARANTEED PROFIT with the given possible odds!

If the bet is an arbitrage bet, then we will be guaranteed to make a profit no matter how we bet. Thus, we can calculate a minimum Guaranteed Profit using the following formula: $(1 / \text{SUM} - 1) * 100$. Thus, in the third stage of the circuit, we use another custom divider circuit to divide 1(base-2) by P, which was obtained above. We then use a simple 2 bit subtractor to subtract 1, since the subtraction will not affect bits after the binary point. Instead of implementing a complex multiplier circuit, in the final stage, we map each of the binary fractions (of the form: X.PQRST) to an approximate percentage in the range 0 - 100% profit (approximation, as we have only limited bits). This is done with a decoder-like circuit, whose expressions can be once again obtained via the Quine - McCluskey Method.

Logisim Circuit Diagram

▼ Detail



Releases

No releases published



[Create a new release](#)

Packages

No packages published

[Publish your first package](#)

Contributors 2

-  **Hayden-Soares**
-  **I-m-a-o** Granth Tiwari

Languages

-  **Verilog** 100.0%