Senior Design - gRNA Generation Pseudo Code

Components:

Need to determine whether or not a base pair matches a passed in base pair

Done in hardware:

1. compareBasePair(input[bp1, bp2], output[match, exactMatch]) {
2. exactMatch = bp == bp2
3. match = "or" each bit from a bitwise "and" of bp1 and bp2
4. }

Need to find potential guide sequences that are 20 base pairs long and have additional three base pairs of PAM sequence at end (-NGG) [N means it can be any nucleotide]

Find the PAM sequence using hardware, store the results using software

Pseudo Code:

Hardware:

1. findGuideSequence(input[sequence, startPos], output[matchPos]) {
2. If startPos is less than 21 base pairs into the big sequence,
3. return (need to make sure there are 20 base pairs before the - NGG to make the guide sequence, will take anything for N)
4. get first base pair from sequence starting at startPos;
5. move through the sequence, taking each nucleotide as“ currBP” until compareBasePair(currBP, G) gives“ exactMatch = 1”(can be anything
6. for N, so doesn’ t really matter what that is: PAM search ignores the N in NGG) Now,
7. continue to move through both the sequence and the PAM sequence until: compareBasePair(currBP, G) gives exactMatch = 0 or we reach the end of the PAM sequence, and all the base pairs have been matched;
8. Once the base PAM sequence is matched or it has been determined that there is no possible guide sequence after startPos, set an output signal of“ done” to 1 and“ found” to whatever is necessary, and“ position” to the leftmost bit of the first base pair of the 23 - bp sequence
9. }

Software:

1. findGuideSequences(input sequence, ouptuts array of positions) {
2. Send signal to the findGuideSequence module to start the process When done = ‘1’, stop
3. if match = ‘0’ and
4. return the array
5. If match = ‘1’ then push the resultant position into the array and restart the search process, with startPos being the least significant bit of the second final G in the PAM sequence Repeat until no new matches found , when that happens return the array of potential guide locations
6. }

Find off-target positions in the full genome and grade them based on the following formula:

A picture containing object

Description generated with very high confidence

Where ‘M’ is a position-based constant for the position in the 23-basepair sequence at which the mismatch takes place.

A close up of a logo

Description generated with very high confidence

‘d’ is the average of the distance between the mismatches. W is a function that returns 0 is there is no mismatch at the position e, and the value of M if there is, and ‘n’ is a penalty value for high-mismatch off-target hits.

In this implementation, if there are more than n = 4 mismatches, the current location is discarded.

(These are the positions at which the CRISPR-Cas9 combination may accidentally cut while looking for the actual target sequence it is given, such that it would be a cut in either a completely wrong area, or a cut in )

Hardware:

1. findSequence(seq, startPos, outputs the position of the msb of the first base pair of the sequence) {
2. Go by chunks of four base pairs from potential guide sequence at a time If more than one base pair is off in the chunk, restart search in next part of genome– don’ t want that many off - targets that close together.
3. If there are more than 4 mismatches in one sequence search, it will restart the search in the next chunk of the genome.
4. }

After all the individual off-target sites are scored, then each sequence is scored using the score of each off-target site:

A picture containing object

Description generated with very high confidence

Where Shit is essentially an array of values (the grades of each off target hit), indexed by hi.