CEC300 Project #3

<u>GPS Gold Code Generation:</u>

The purpose of this assignment is to expose you to how gold codes are generated. As discussed in class, Gold codes have high-auto-correlation, and low-cross-correlation, which makes them useful to identify each of the GPS satellites.

You are tasked to compute the first 10 output bits of GPS PRN #1. The feedback and output connections for PRN #1 are shown below. Initialize the G1 and G2 lfsr registers with all 1's at the beginning of your code. In the figure below note that ⊕ represents an exclusive OR operation. E.g., 1⊕1=0, 1⊕0=1. In Matlab, use the xor(a,b) function.

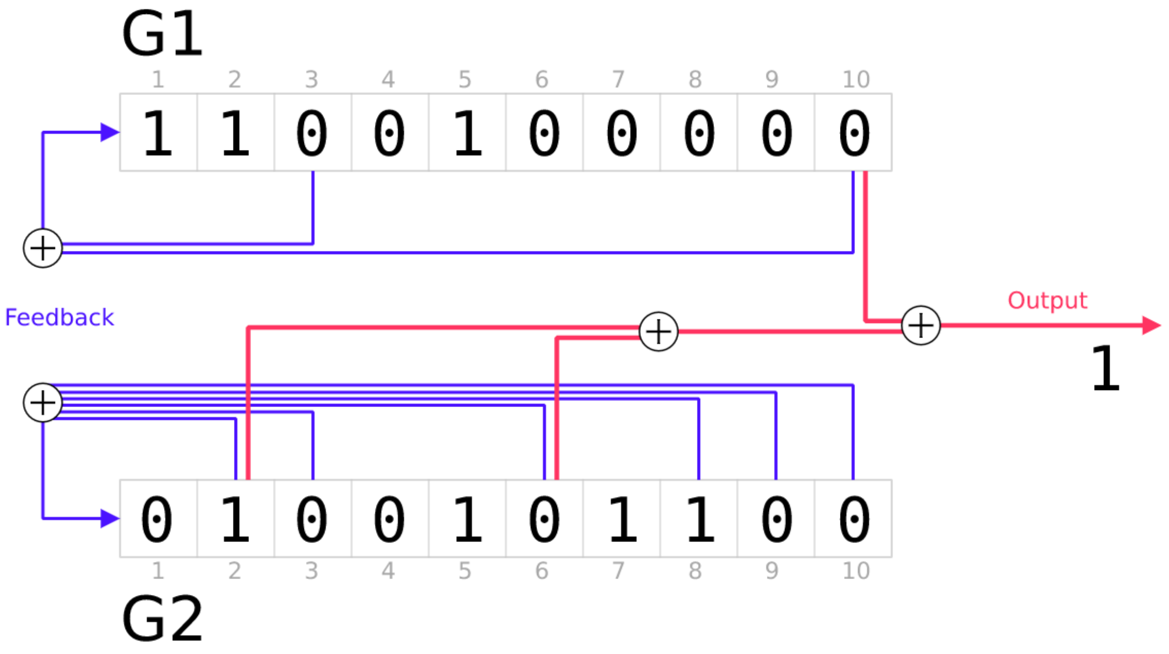

Fig. 1. Operations to generate the PRN#1 bits.

Fig. 1 shows how the Gold code is generated. First, an output is generated by performing exclusive OR (xor) between specific positions of the G1 and G2 registers. This output is the least significant bit of the PRN code. Then, another exclusive-or operation is used to compute the new most significant bit (MSB) of the register. The registers are shifted one position to the right, and the new computed MSB is placed. The resulting PRN code is shifted to the left, and the process is repeated.

<u>The deliverables of the assignment are:</u>

a. Write a pseudo-code that shows how will you solve the problem. Submit this as a PDF file in Canvas. (40 points)

b. Write a Matlab function that will calculate the first 10 bits of the PRN#1 code, using the operations described in the figure. (60 points)

Hints:

bin2dec() – Transforms a binary number to decimal. E.g., >> bin2dec('1111111111')

%*result*: ans =1023

dec2bin() – Transforms a decimal number to binary. E.g., >> dec2bin(1023)

%*result*: ans = '1111111111'

Shift bits to the right:

bitsrl(a,k) – Shifts the bits of number a, k times to the right. E.g.,

bitsrl(int16(1023),1) *result*: ans = int16 511 **Note:** int16() transforms the double number to a signed integer with 16 bits. This is required for the function to work. Also note that 511 is '0111111111'.

Shift bit to the left:

bitshift(a,k)

Get the bits to perform the OR operations:

bitget(a,k) – The output is the bit k of the number a. **Notes:** a needs to be the decimal that corresponds to the binary number. **k here is measured from left to right. In the assignment figure is the opposite. You need to figure out what the required positions are for this function**.

E.g.: >> bitget(511,10)

%result: 0. bitget(511,9) result= 1. bitget(511,8) result: 1.

Add bit to the most significant bit (MSB) position:

Bit – this contains the bit to add.

Number – This is the number to which we will add the bit. For this assignment, this number was already shifted to the right before adding the bit.

New_number – Number with the bit added.

Equation: New_number=Number+bit*512 Note: 512 corresponds to '1000000000'. If bit is 1, then a bit 1 is added to the number. If bit is 0, then the most significant bit stays as 0.

E.g., 511 corresponds to '0111111111'. If we add bit =1 to the MSB position, we have:

New_number= 511+1*512 = 1023, which is equivalent to '1111111111'.

Another example:

>> bin2dec('0011011010')

%result: 218. Add bit 1 to MSB: New_number=218+1*512 = 730.

>> dec2bin(730) result: '1011011010'