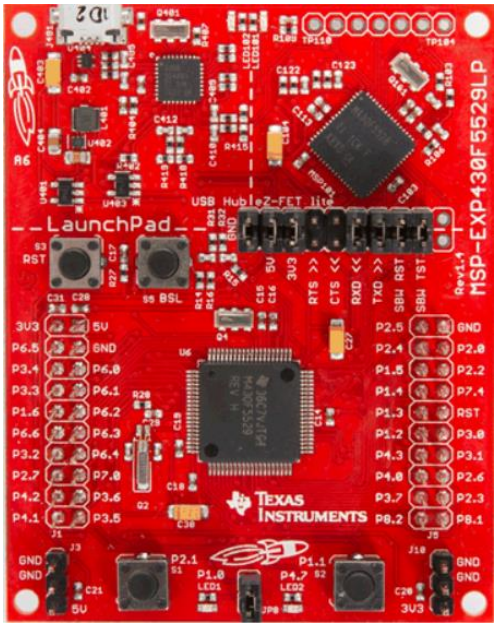
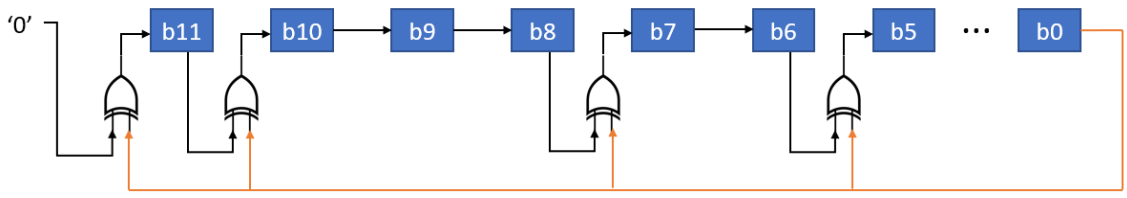
Desafio de programação 1

Vamos desenvolver um jogo de teste de reflexo. Apresentaremos ao usuário um aviso luminoso através dos LEDs e o jogador deve responder apertando o botão correspondente o mais rápido possível. Se o LED vermelho piscar, o usuário deve apertar o botão S1 (da esquerda). Se o LED verde piscar, o usuário deve apertar o botão S2 (da direita). Se os dois LEDs acenderem, o usuário deve apertar os dois botões.



Para deixar o jogo mais desafiante, vamos gerar intervalos de tempo aleatórios para confundir o jogador. Precisaremos então de uma subrotina que gera números aleatórios e outra subrotina para gastar tempo.

1) Vamos programar uma rotina que gera números pseudo-aleatórios usando uma técnica conhecida como Galois Linear Feedback Shift Registers. (1,5 pontos)



A implementação da técnica consiste em deslocar um registro de 12-bits para a direita e aplicar uma porta ou-exclusiva em bits selecionados. A escolha desses bits é importante para gerar aleatoriedade e periodicidade à sequência de números aleatórios.

As posições dos bits que recebem o valor da porta XOR são marcadas por polinômios, onde a potência da variável x indica a posição do bit. Esse nome vem da teoria de operações em campos de Galois. No exemplo da figura temos o polinômio gerador $x^{11} + x^{10} + x^7 + x^5$.

2) A rotina que conta tempo recebe como argumento o valor de tempo em milissegundos. Essa rotina deve ser capaz de gastar tempo no intervalo de 128ms até 4 segundos para tornar o jogo mais interessante. Considere que o clock do processador está em $2^{20}\ Hz$. (2,5 pontos)

DICA: Minha recomendação é criar uma rotina que conte intervalos múltiplos de 128ms e o valor de tempo requisitado pelo usuário será dividido por 128 (note que se trata de um número que é potência de 2, para facilitar a divisão).

3) Vamos criar uma rotina que reaproveita o número aleatório gerado e extrai os bits 1 e 9 para fazer os LEDs piscarem de maneira aleatória. Além do tempo aleatório, teremos aleatoriedade na escolha dos LEDs. Faça uma rotina que acenda o LED verde se o bit 1 do número aleatório for '1', e que acenda o LED vermelho se o bit 9 do número aleatório for '1'. Em outras palavras, os LEDs devem imitar o valor dos bits 1 e 9 do número aleatório. Atenção, se os bits 1 e 9 forem '0', você deve gerar outro número aleatório, pois não há como jogar sem piscar os LEDs. (1,5 pontos)

4) Quando o jogador reagir, devemos contabilizar o tempo entre o piscar do LED e o pressionar do botão. Faça então uma rotina que monitora os dois botões sem travar a execução. Cada ciclo de monitoramento deve ser registrado num acumulador de 32-bits que servirá como tempo de reação do jogador. Se o jogador conseguir reagir dentro de um tempo estipulado, ele ganha. Caso contrário, perde. Você decidirá quanto tempo é necessário para apertar os botões. Isso, obviamente, regula a dificuldade do jogo. Lembre-se de remover os rebotes dos botões. Para seguir adiante, você pode esperar a condição dos dois botões estarem soltos para liberar a resposta para o usuário. (2,5 pontos)

5) Resposta ao usuário: Devemos providenciar uma forma visual para avisar o usuário se ele acertou ou errou o botão. Sugiro piscar o LED verde 3 vezes, em caso de acerto; piscar o LED vermelho 3 vezes em caso de erro e piscar os dois LEDs 3 vezes caso o usuário tenha demorado muito para apertar o botão. (2 pontos)