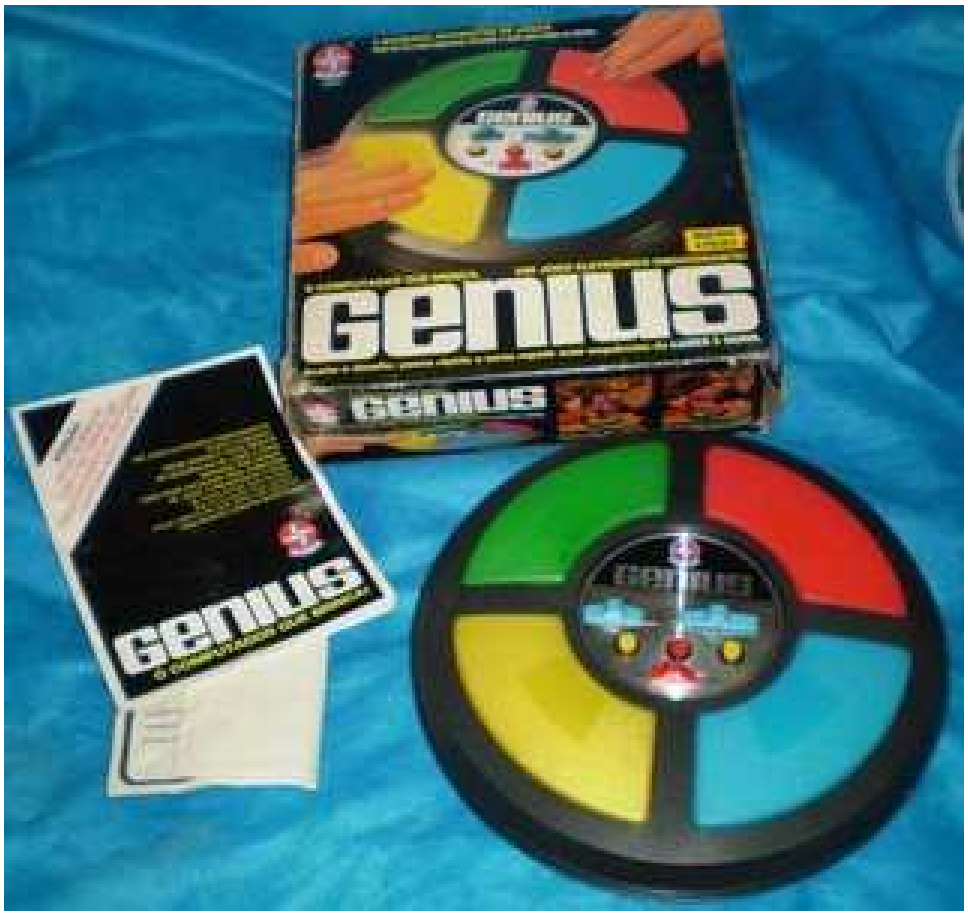
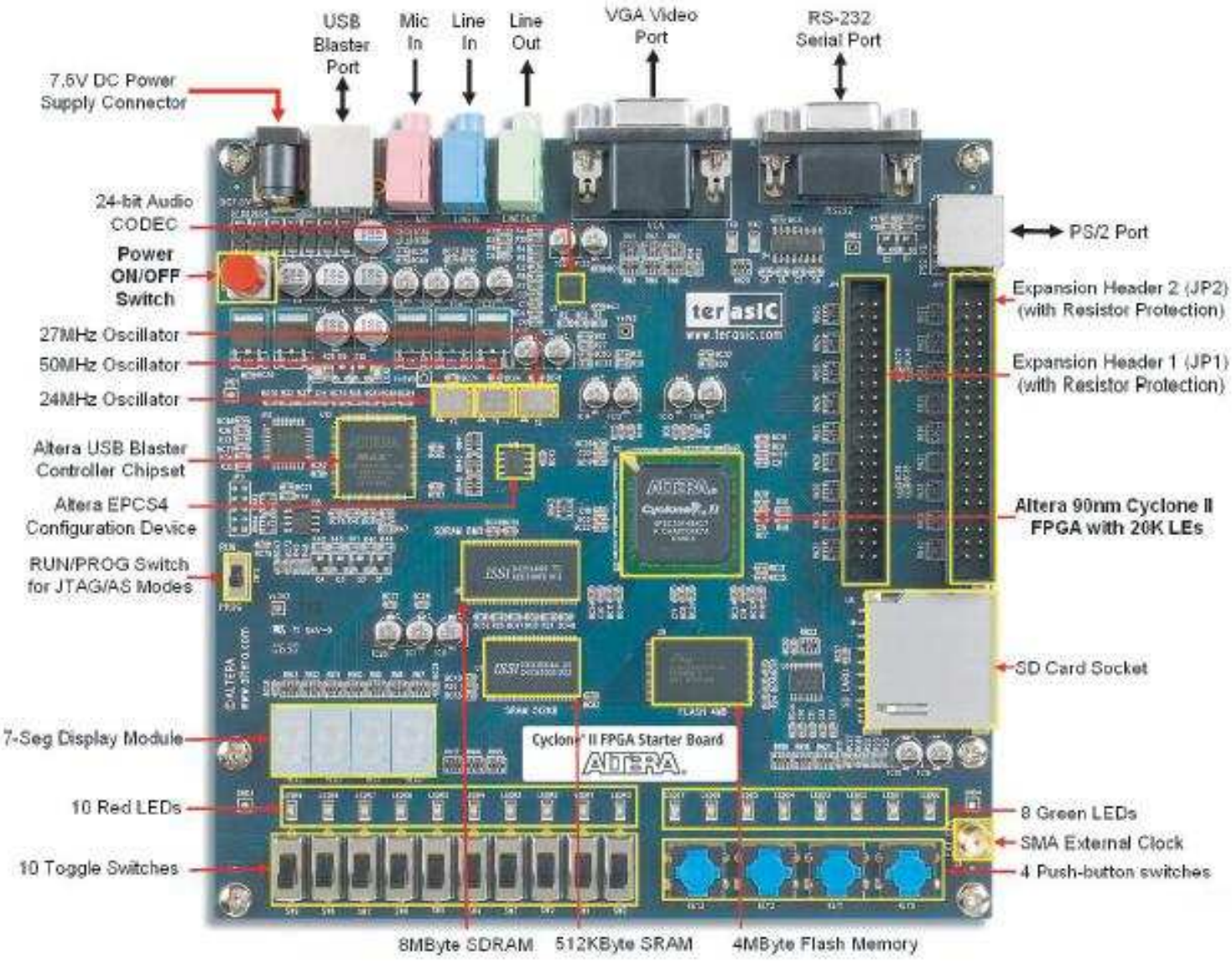
Projeto Dirigido: Genius



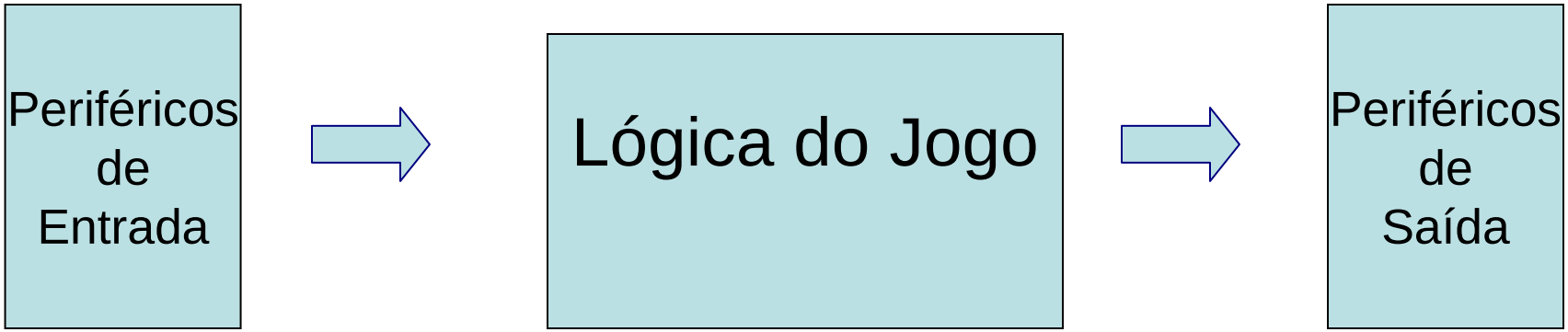
Genius era um brinquedo muito popular na década de 80 e que buscava estimular a memorização de cores e sons. Com um formato semelhante a um objeto voador não identificado, possuía botões coloridos que emitiam sons harmônicos e se iluminavam em seqüência. Cabia aos jogadores repetir o processo sem errar.

Genius em FPGA-SDB

Figure 1–1. Starter Development Board



Especificação Funcional



chaves ou botoeiras para representar os elementos de uma sequência

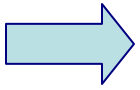
1. Gerar uma sequência de 1 elemento.
2. Exibir a sequência.
3. Esperar que o jogador repita a sequência.
4. Se o jogador acertar a sequência, adiciona mais um elemento à sequência enquanto o número de elementos for menor que XX e vá para (2).

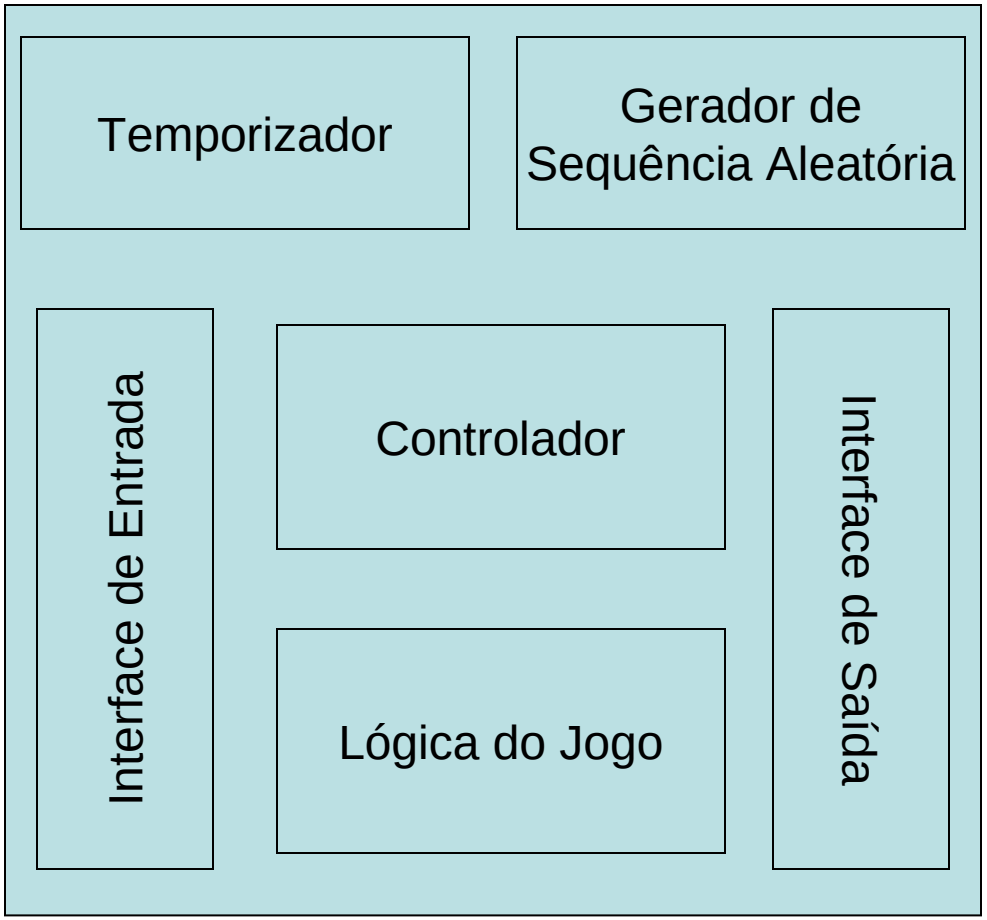
Leds ou displays de 7 seg. para mostrar os estados do jogo

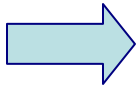
Especificação Estrutural

? chaves

? botoeiras

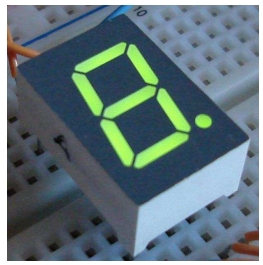






? *leds* verdes

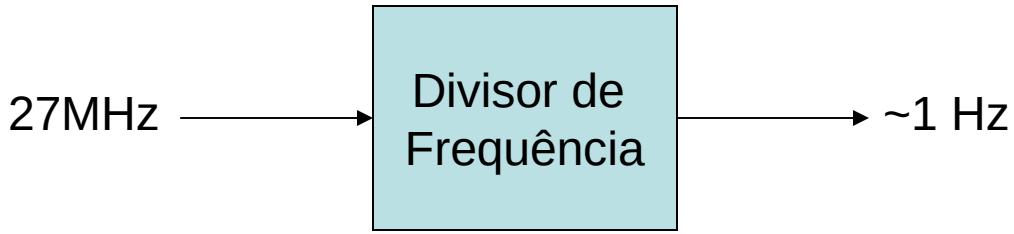
? *leds* vermelhos



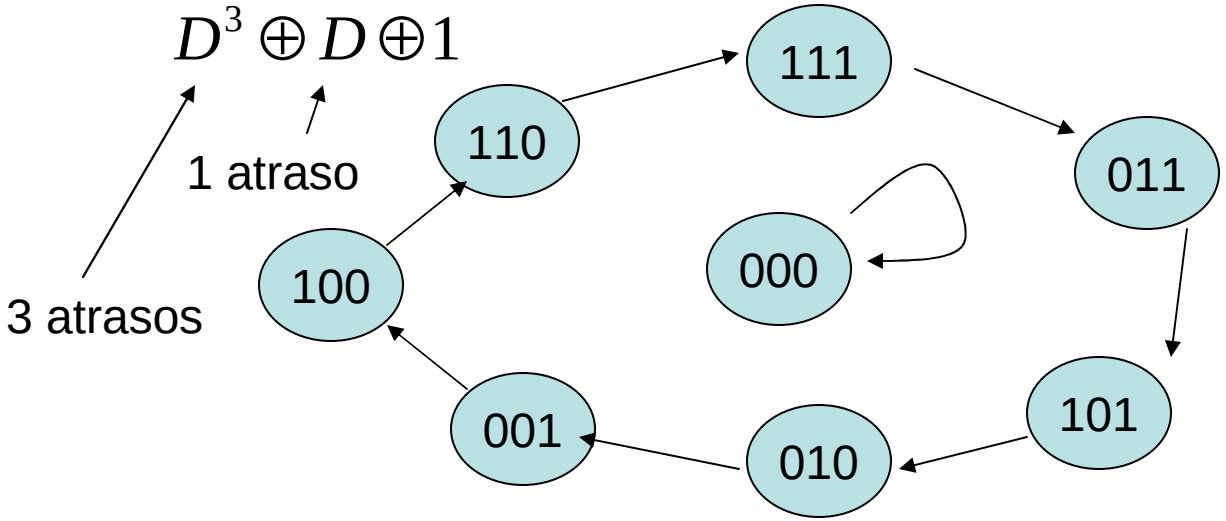
? *displays*

Temporizador e Gerador

- Temporizador: sinais de referência temporal



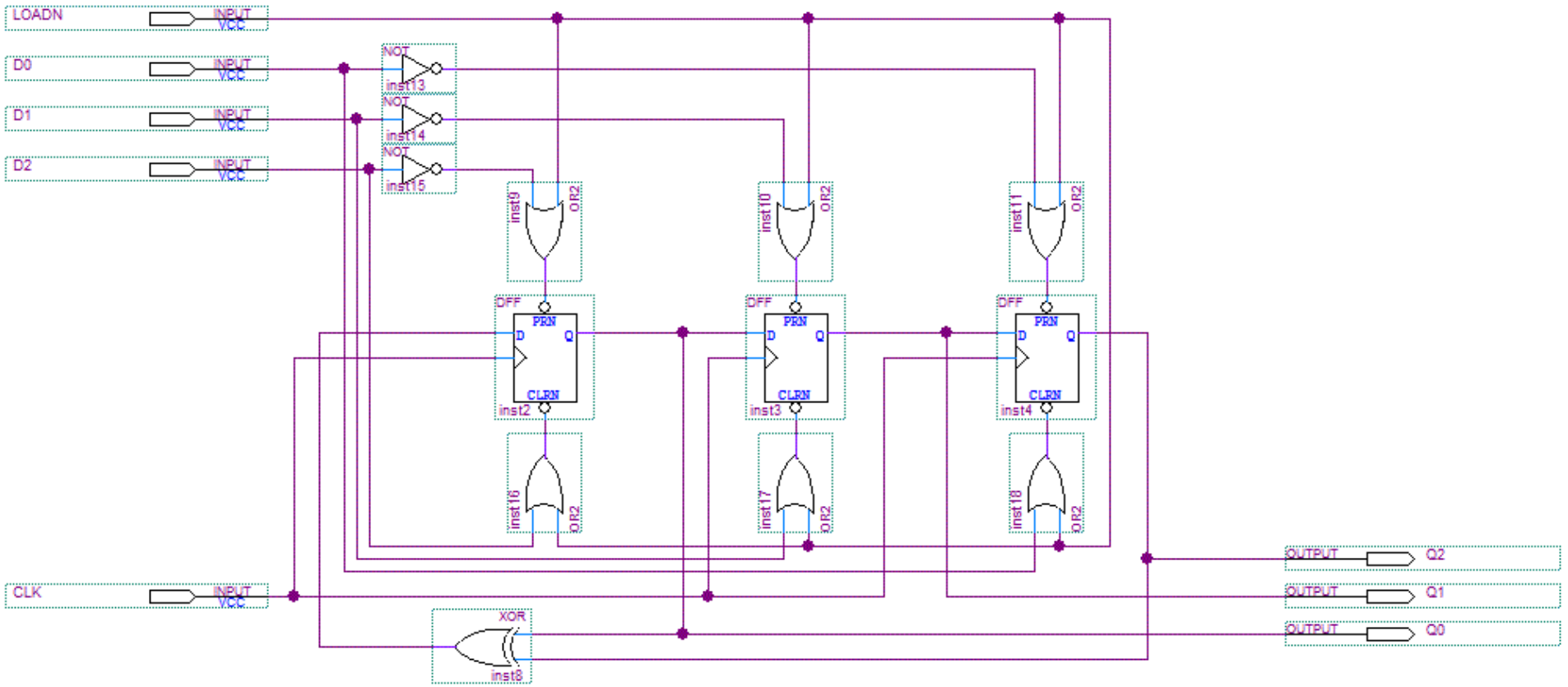
- Gerador: máquina autônoma de máxima sequência



Em cada jogo gera-se aleatoriamente uma sequência e esta sequência é preservada em um mesmo jogo.

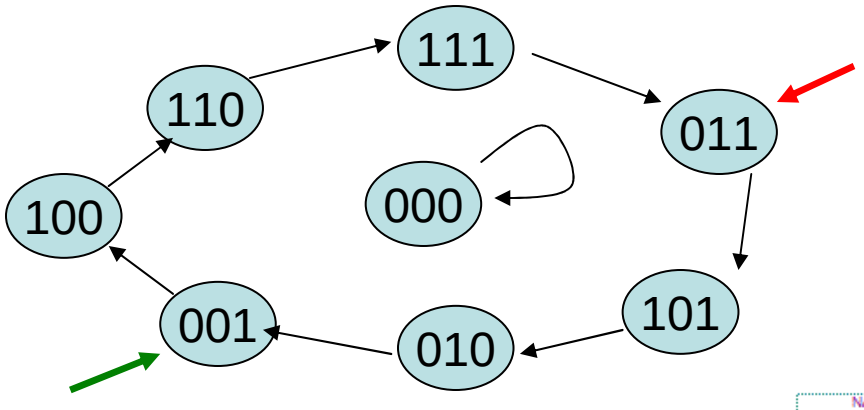
Como implementar um gerador?

- Uso de 3 *flip-flps* D e portas lógicas.

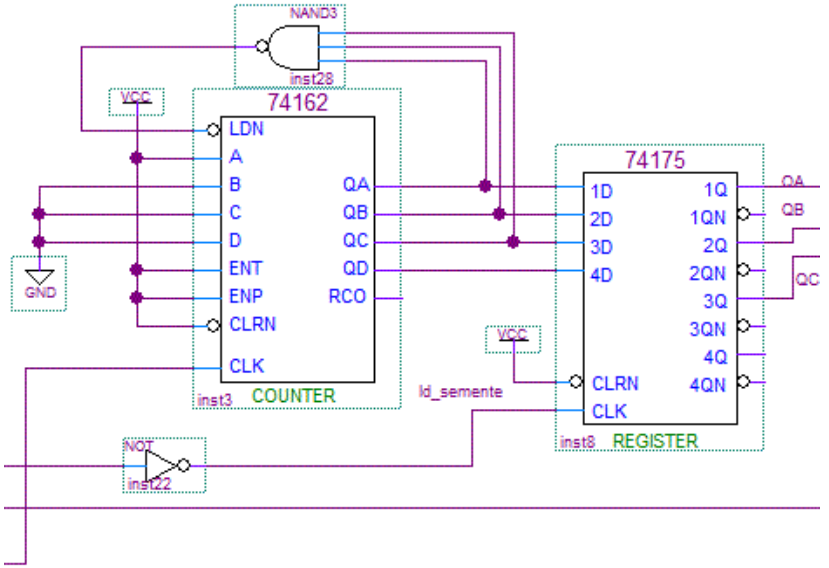


Como gerar “aleatoriamente” uma sequência?

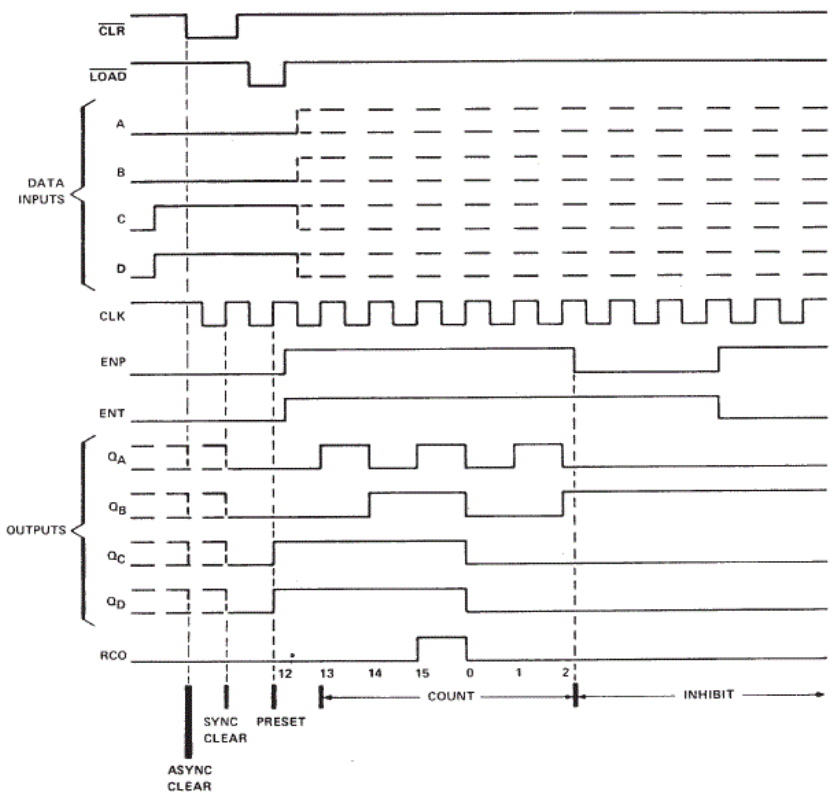
- Estratégia adotada: escolher aleatoriamente um estado da máquina autônoma no início de cada jogo e armazenar esta “semente”.



- Como escolher aleatoriamente este estado?
  - Contador correndo “livremente” a 27MHz.
  - Ler aleatoriamente o estado do contador e armazená-lo.



7474, 74162 e 74175



74162: Contador Síncrono com CLEAR síncrono

FUNCTION TABLE
(EACH FLIP-FLOP)

INPUTS			OUTPUTS	
CLEAR	CLOCK	D	Q	$\overline{Q}$†
L	X	X	L	H
H	↑	H	H	L
H	↑	L	L	H
H	L	X	Q_0	$\overline{Q}_0$

H = high level (steady state)
L = low level (steady state)
X = irrelevant
↑ = transition from low to high level
Q_0 = the level of Q before the indicated steady-state input conditions were established.
† = '175, 'LS175, and 'S175 only

74175: Registrador de 4 *flip-flops* D com CLEAR

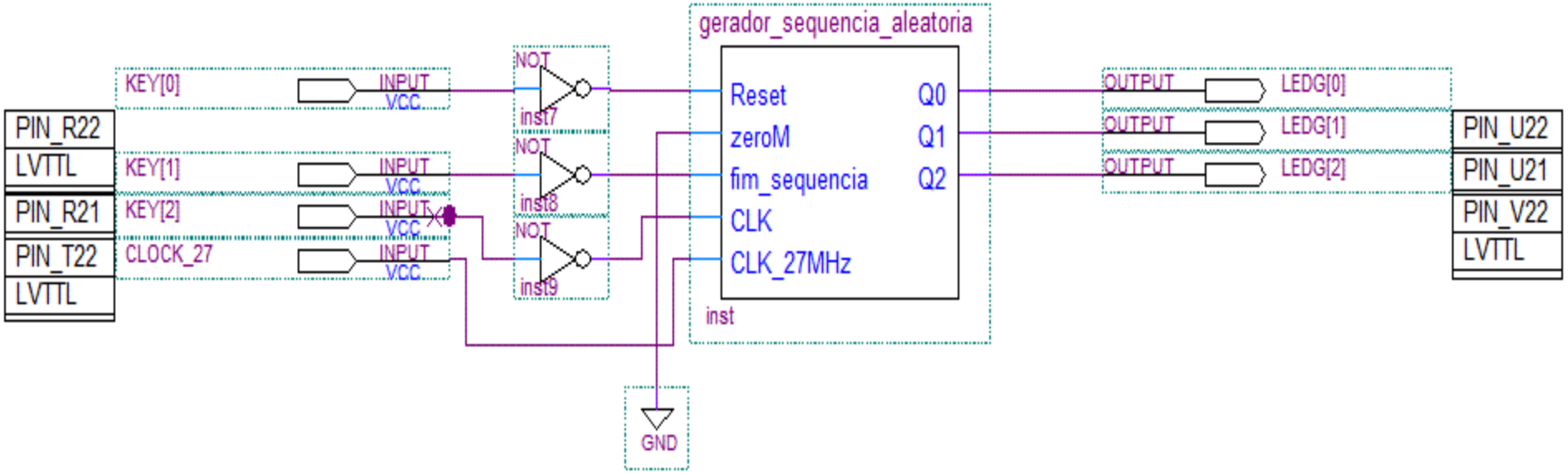
7474: *flip-flops* D com CLEAR e PRESET

INPUTS				OUTPUTS		FUNCTION
$\overline{CLR}$	$\overline{PR}$	D	CK	Q	$\overline{Q}$	
L	H	X	X	L	H	CLEAR
H	L	X	X	H	L	PRESET
L	L	X	X	H	H	----
H	H	L	↑	L	H	----
H	H	H	↑	H	L	----
H	H	X	↓	Q_n	$\overline{Q}_n$	NO CHANGE

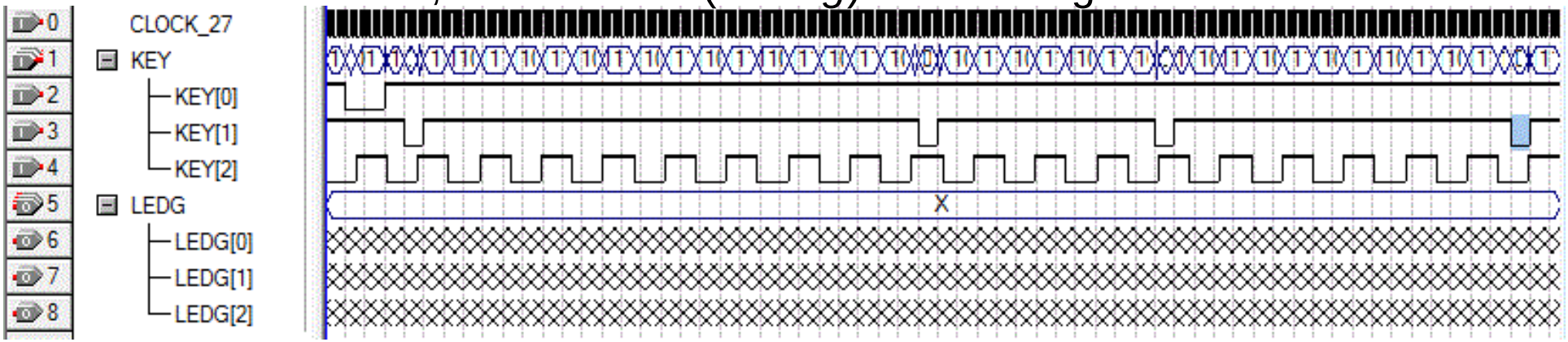
X : Don't Care

1º. Experimento

1. Analise o circuito gerador_sequencia_aleatoria

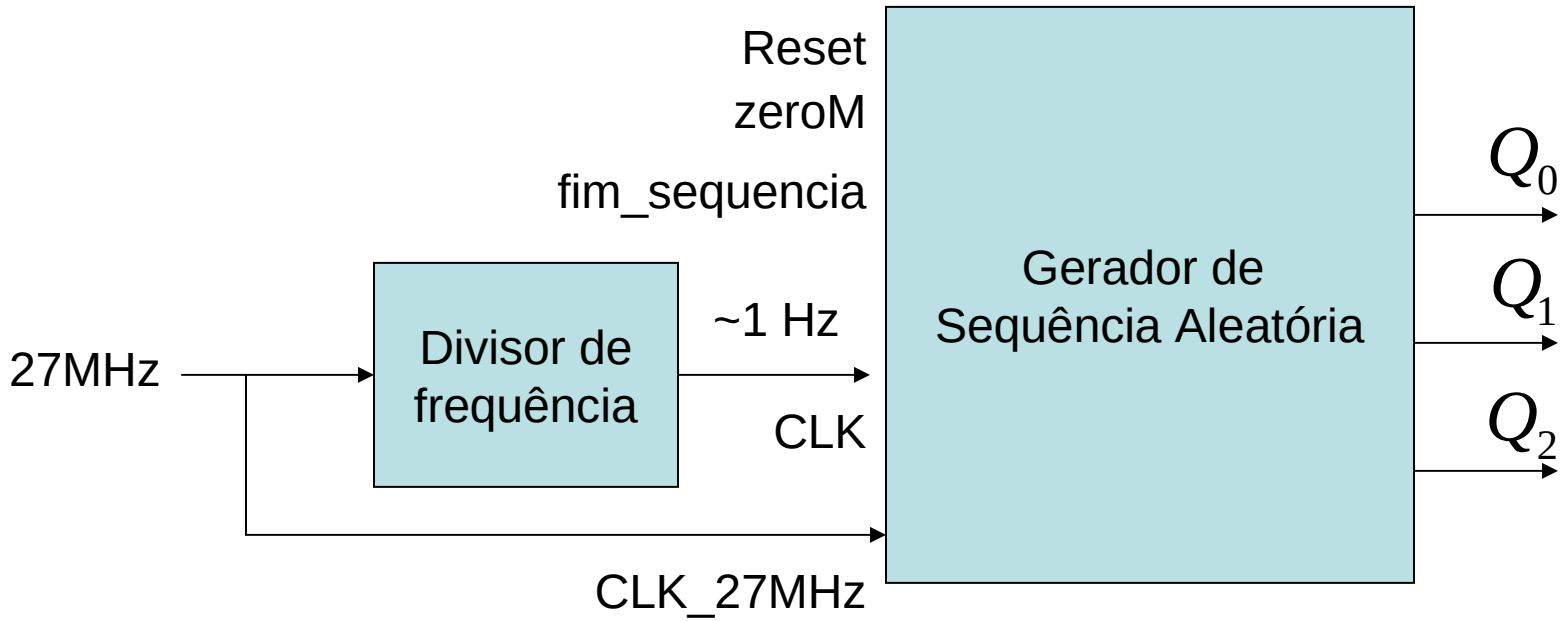


2. Anaalise o circuito, simulando-o (*Timing*) com as seguintes formas de onda



1^0. Experimento

3. Adicione o divisor de frequência da aula anterior e programe o circuito

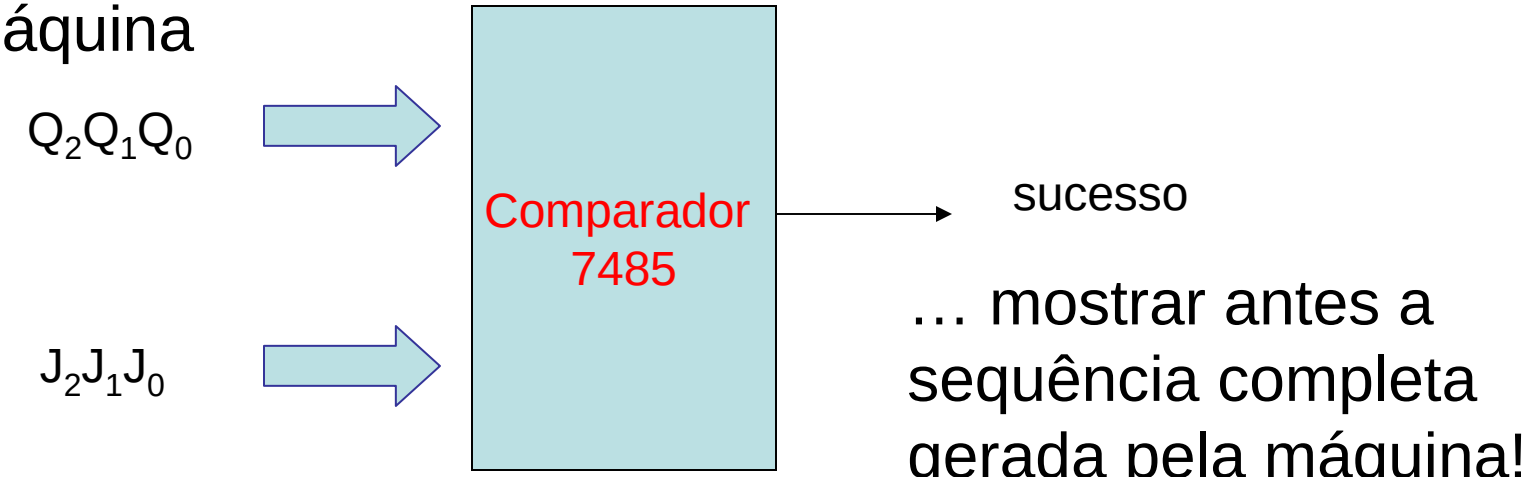


4. Apresente os resultados do projeto implementado em FPGA reproduzindo a sequência de ações simuladas através dos periféricos.

Interface de Entrada e Lógica do Jogo

- Interface de Entrada: um conjunto de 7 elementos, cada um mapeado em uma chave

- Lógica do jogo: comparar a sequência do jogador com a da máquina

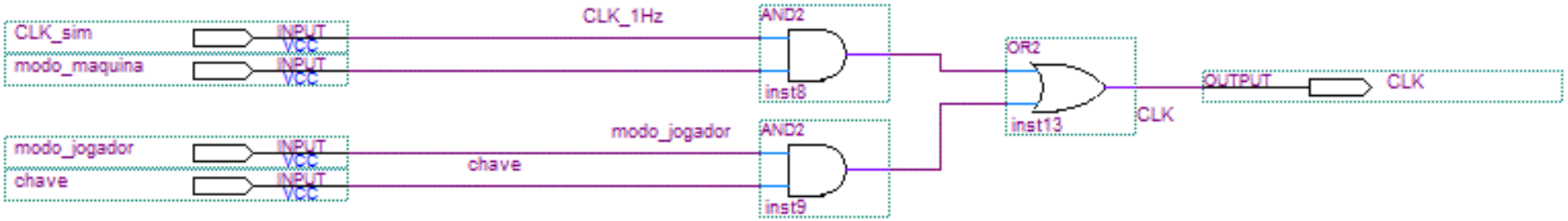


… mostrar antes a sequência completa gerada pela máquina!

Como adequar o tempo do jogo ao tempo da máquina (vez da máquina) e ao tempo do jogador (vez do jogador)?

- Vez da máquina: os elementos da sequência são “varridos” a uma frequência pré-estabelecida (CLK_sim), visualmente distinguível (em torno de 1Hz).
- Vez do jogador: os elementos da sequência são “varridos” a frequência do jogador (chave).

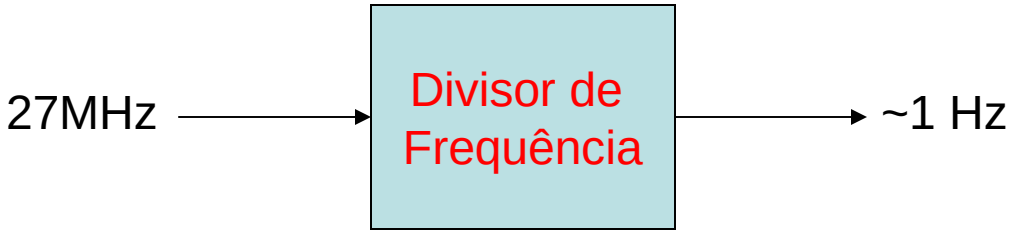
Como chavear entre estas duas frequências?
Introduzir um estado: modo da sequência!



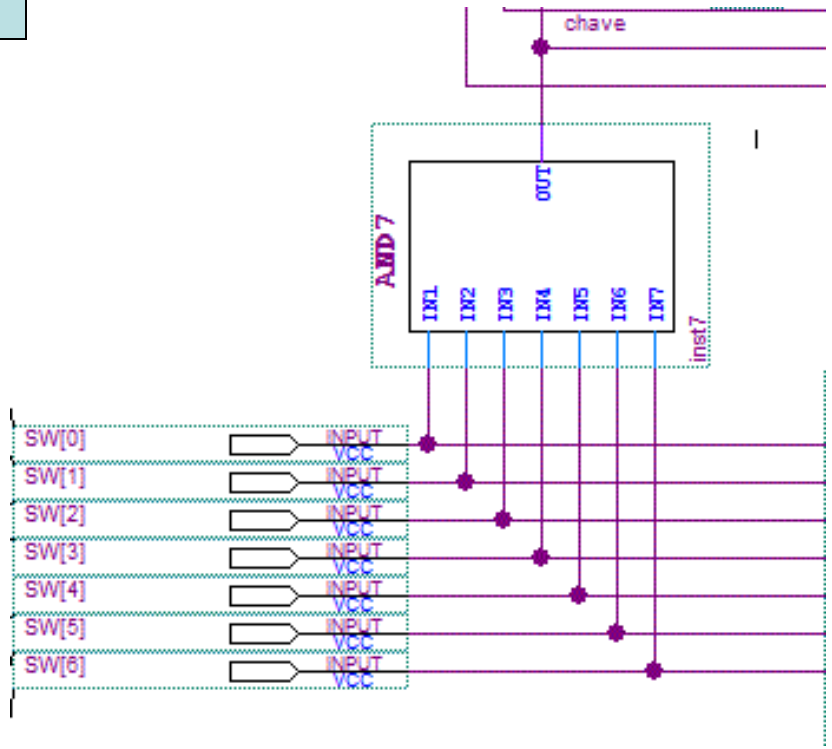
Comparação só é habilitada no modo do jogador!

Como gerar CLK_sim e chave?

•CLK_sim: pode derivar diretamente do CLOCK do sistema.

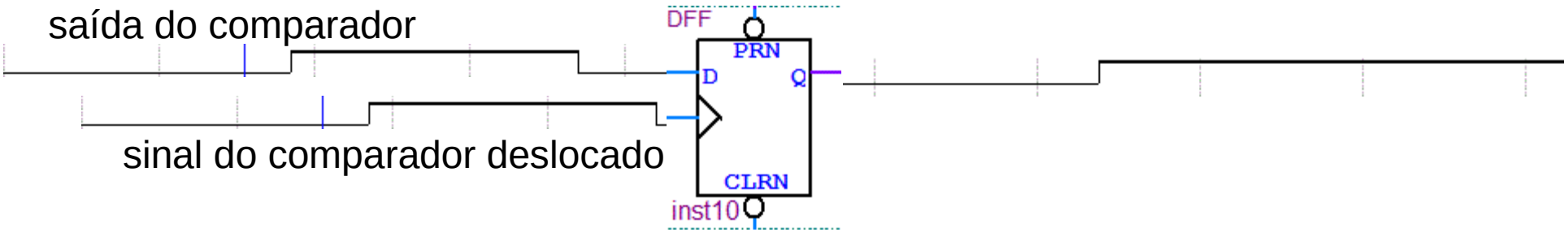


•chave: os elementos da sequência são “varridos” a frequência do jogador (chave).



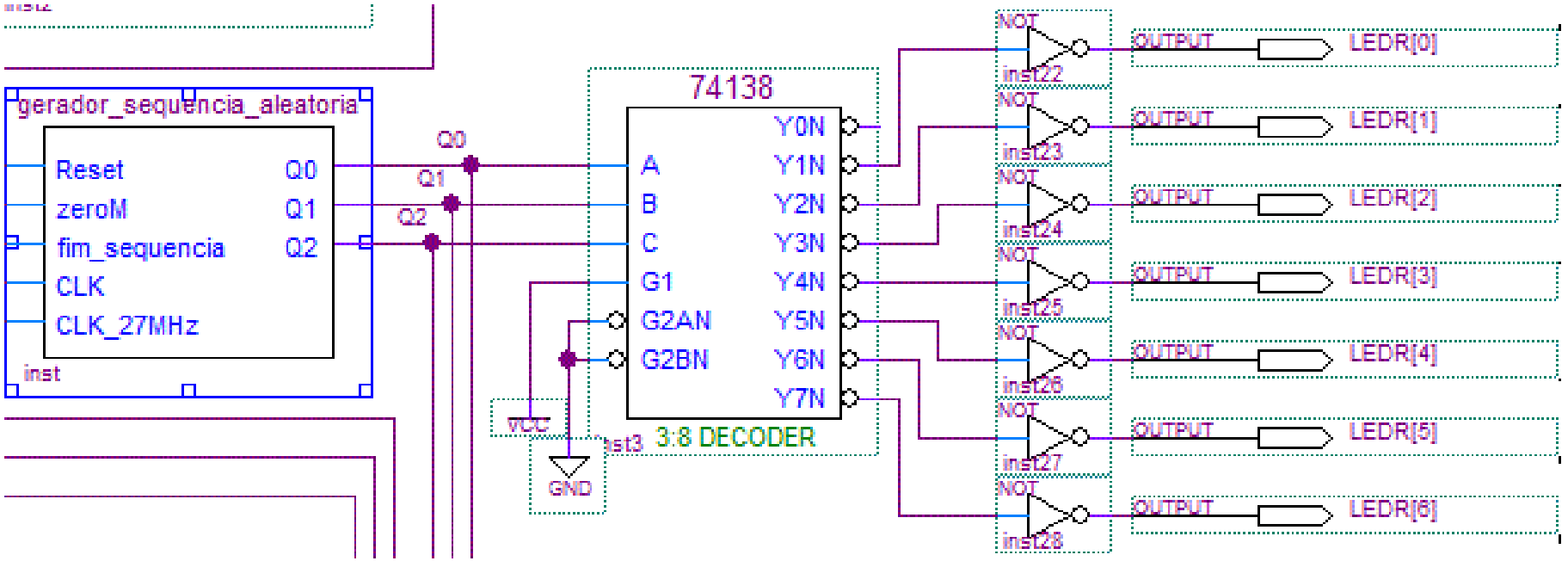
Como comparar e preservar o estado “sucesso” até a próxima entrada do jogador?

Comparador 7485 é um circuito combinacional. O resultado da comparação depende das entradas. Para assegurar corretude nos processamentos subsequentes, precisa-se armazenar o estado da comparação até a próxima entrada. Em qual momento que o *flip-flop* D (armazenador de bit) deve ser engatilhado?



Como mostrar os elementos da sequência gerada pela máquina “unitariamente”?

- Utilizar 7 *bits* ao invés de 3 → Decodificador 74138.



7485: Comparador de 4 *bits*

Function Table

Comparing Inputs				Cascading Inputs			Outputs		
A3, B3	A2, B2	A1, B1	A0, B0	A > B	A < B	A = B	A > B	A < B	A = B
A3 > B3	X	X	X	X	X	X	H	L	L
A3 < B3	X	X	X	X	X	X	L	H	L
A3 = B3	A2 > B2	X	X	X	X	X	H	L	L
A3 = B3	A2 < B2	X	X	X	X	X	L	H	L
A3 = B3	A2 = B2	A1 > B1	X	X	X	X	H	L	L
A3 = B3	A2 = B2	A1 < B1	X	X	X	X	L	H	L
A3 = B3	A2 = B2	A1 = B1	A0 > B0	X	X	X	H	L	L
A3 = B3	A2 = B2	A1 = B1	A0 < B0	X	X	X	L	H	L
A3 = B3	A2 = B2	A1 = B1	A0 = B0	H	L	L	H	L	L
A3 = B3	A2 = B2	A1 = B1	A0 = B0	L	H	L	L	H	L
A3 = B3	A2 = B2	A1 = B1	A0 = B0	L	L	H	L	L	H
A3 = B3	A2 = B2	A1 = B1	A0 = B0	X	X	H	L	L	H
A3 = B3	A2 = B2	A1 = B1	A0 = B0	H	H	L	L	L	L
A3 = B3	A2 = B2	A1 = B1	A0 = B0	L	L	L	H	H	L

H = High Level, L = Low Level, X = Don't Care

74147: Codificador

'147, 'LS147

FUNCTION TABLE

INPUTS									OUTPUTS			
1	2	3	4	5	6	7	8	9	D	C	B	A
H	H	H	H	H	H	H	H	H	H	H	H	H
X	X	X	X	X	X	X	X	L	L	H	H	L
X	X	X	X	X	X	X	L	H	L	H	H	H
X	X	X	X	X	X	L	H	H	H	L	L	L
X	X	X	X	X	L	H	H	H	H	L	L	H
X	X	X	X	L	H	H	H	H	H	L	H	L
X	X	X	L	H	H	H	H	H	H	L	H	H
X	X	L	H	H	H	H	H	H	H	H	L	L
X	L	H	H	H	H	H	H	H	H	H	L	H
L	H	H	H	H	H	H	H	H	H	H	H	L

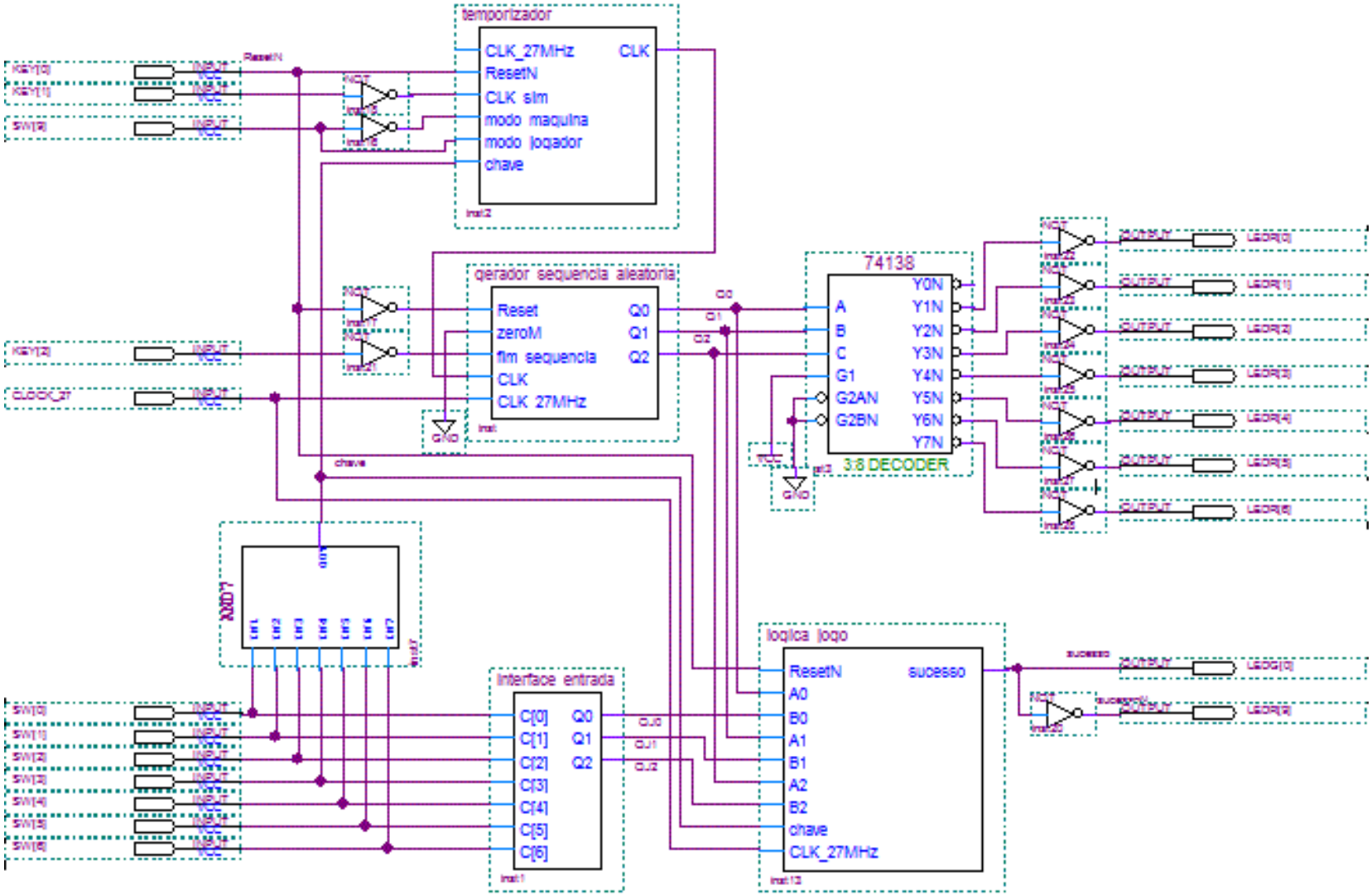
H = high logic level, L = low logic level, X = irrelevant

74138: Decodificador

Inputs					Outputs							
Enable		Select										
G1	G2 (Note 1)	C	B	A	YO	Y1	Y2	Y3	Y4	Y5	Y6	Y7
X	H	X	X	X	H	H	H	H	H	H	H	H
L	X	X	X	X	H	H	H	H	H	H	H	H
H	L	L	L	L	L	H	H	H	H	H	H	H
H	L	L	L	H	H	L	H	H	H	H	H	H
H	L	L	H	L	H	H	L	H	H	H	H	H
H	L	L	H	H	H	H	H	L	H	H	H	H
H	L	H	L	L	H	H	H	H	L	H	H	H
H	L	H	L	H	H	H	H	H	H	L	H	H
H	L	H	H	L	H	H	H	H	H	H	L	H
H	L	H	H	H	H	H	H	H	H	H	H	L

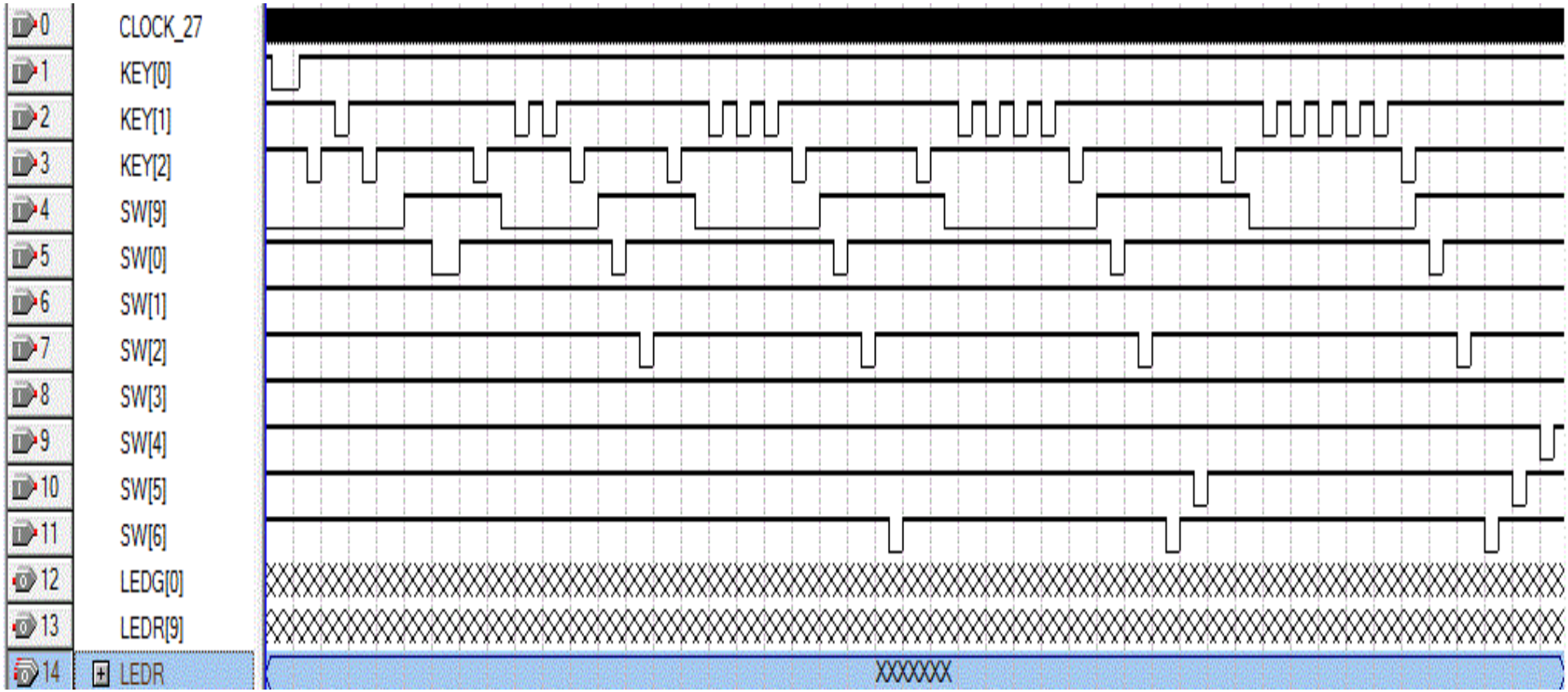
2^0. Experimento

1. Desenhe o esquemático



2^0. Experimento

2. Analise o comportamento do circuito, simulando-o com as seguintes formas de onda. Observe que pulsos devem ter largura maior que 1.5us.



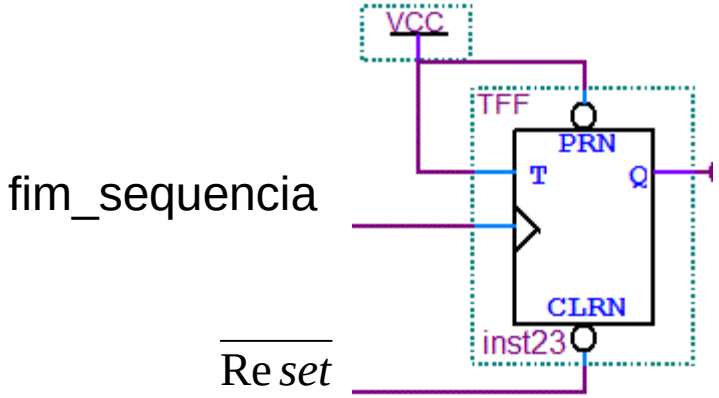
3. Apresente os resultados do projeto implementado em FPGA reproduzindo a sequência de ações simuladas através dos periféricos.

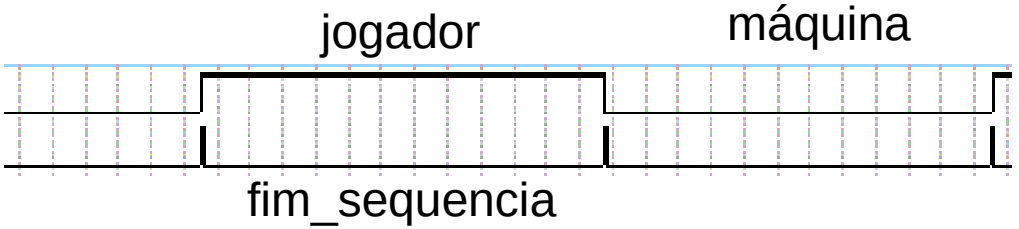
Controlador

- Automatizar alguns passos que podem ser independentes do jogador (gerar automaticamente os sinais Key[1], Key[2] e SW[9]):
  - gerar a sequência durante a jogada da máquina a uma frequência pré-estabelecida.
  - incrementar um elemento após cada jogada
  - assegurar que a sequência da jogada anterior seja repetida
  - alternar o estado da máquina para receber a sequência do jogador (a frequência do jogador)
  - assegurar que a máquina seja bloqueada quando a sequência máxima é alcançada
  - sincronizar a ação do jogador com a da máquina na transição de estados para assegurar que todos os elementos da sequência da máquina sejam visíveis

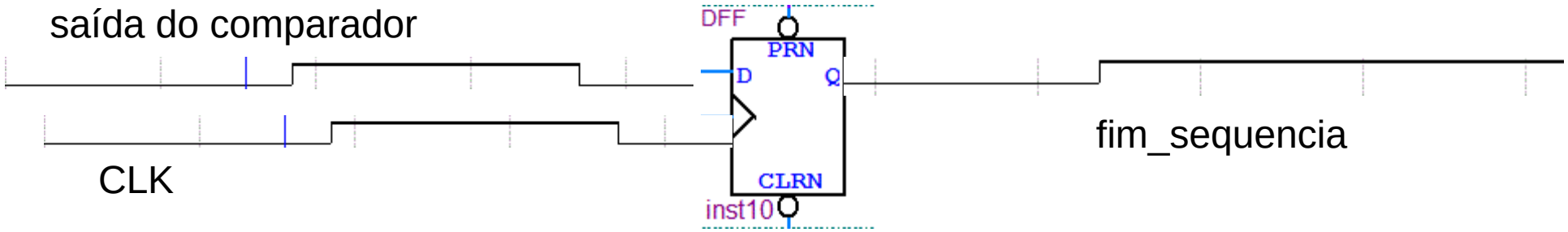
Como alternar entre modo de máquina e modo de jogador de forma automática?

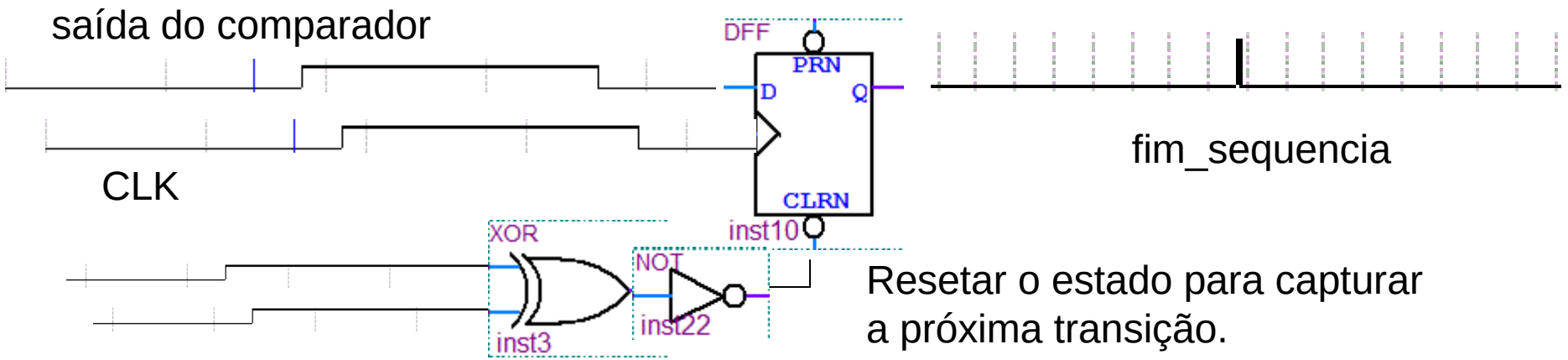
- *Bit* de estado armazenado em um *flip-flop* T





Como gerar pulsos fim_sequencia?



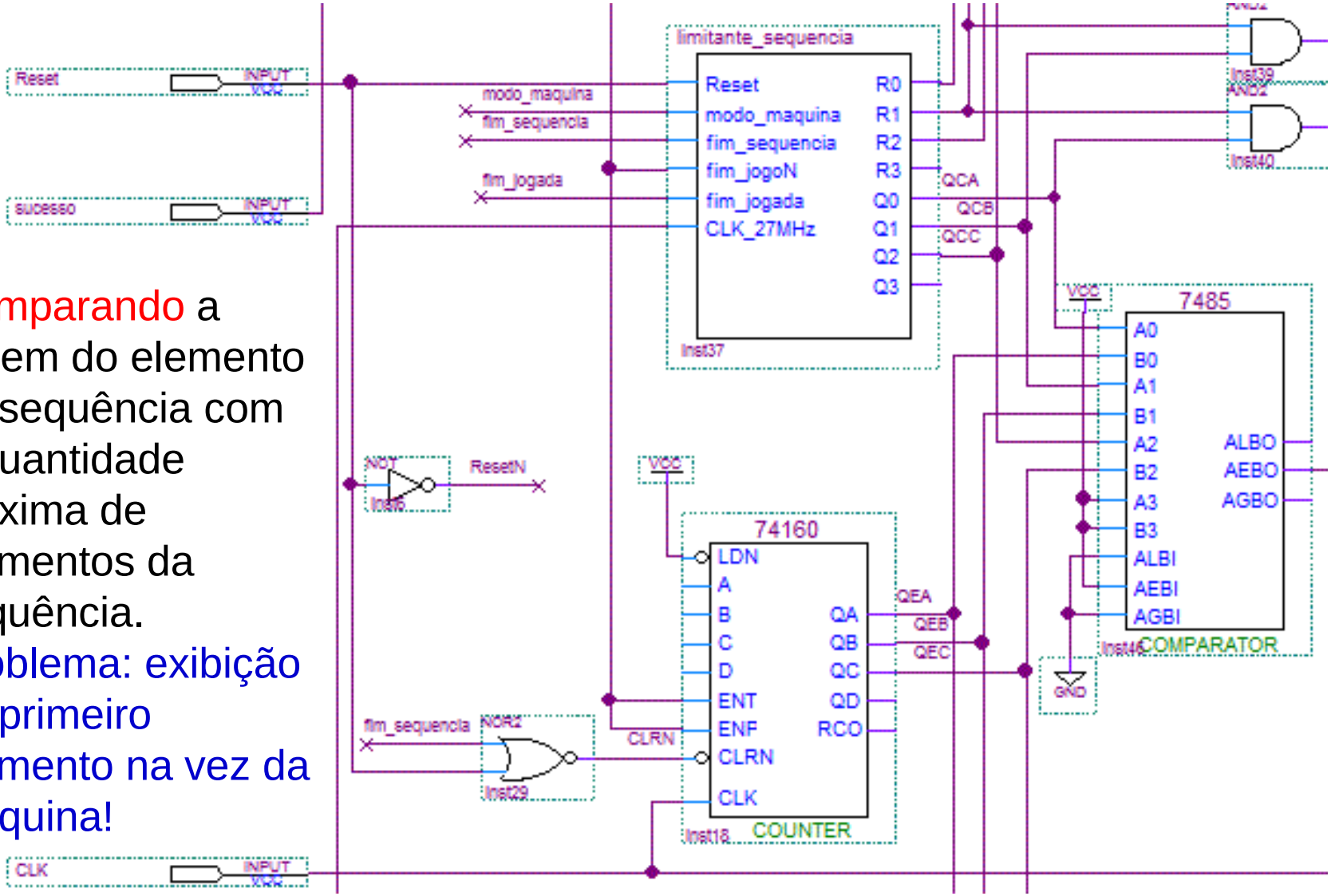


Resetar o estado para capturar a próxima transição.

Como controlar o número de elementos de cada sequência?

Comparando a ordem do elemento na sequência com a quantidade máxima de elementos da sequência.

Problema: exibição do primeiro elemento na vez da máquina!



Como identificar que fim_sequencia coincide com fim_jogada ou fim_jogo?

- fim_jogada:após duas sequências: sequência da máquina e sequência do jogador

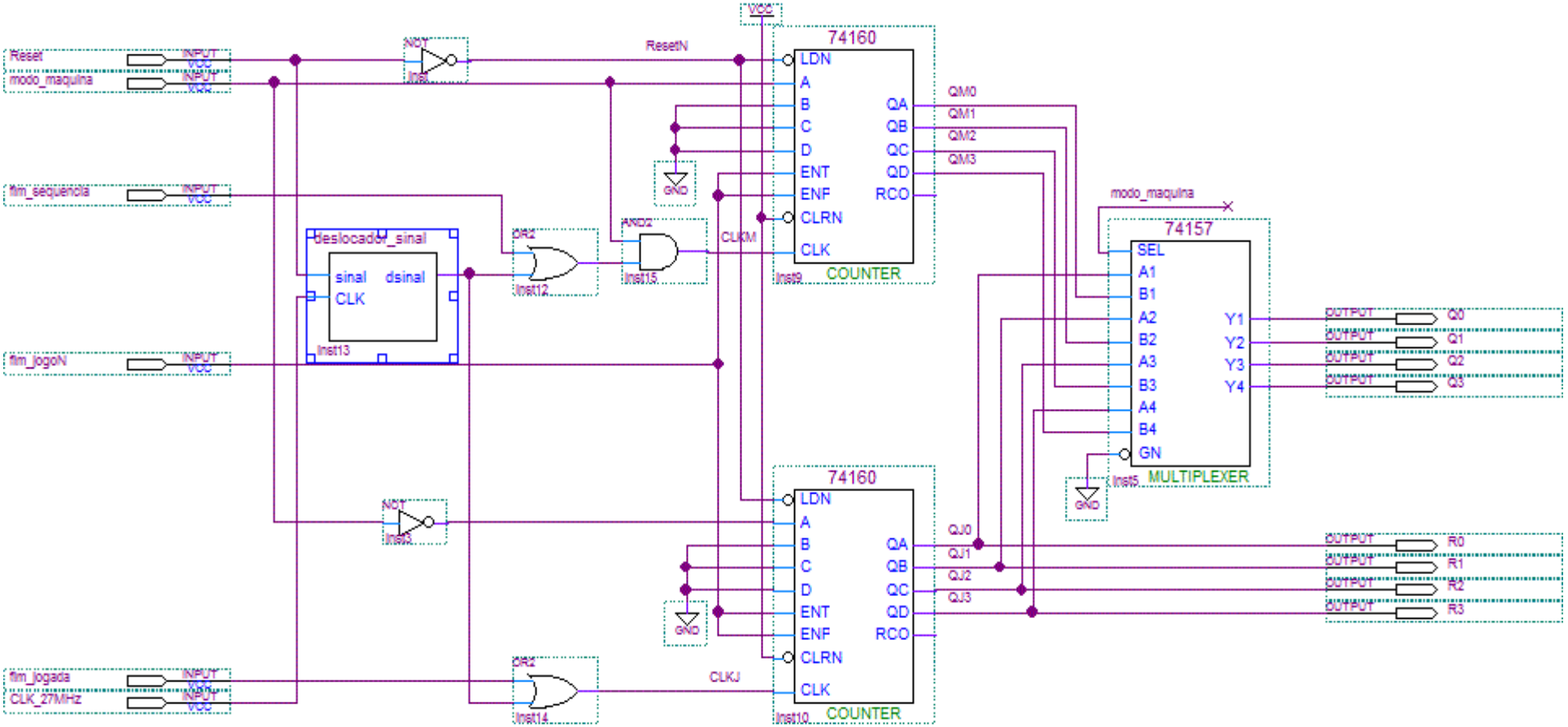
$$fim\_jogada = fim\_sequencia \wedge \mathrm{mod}\, o\_jogador$$

- fim_jogo: quando o limitante da sequência atinge o valor máximo permitido.

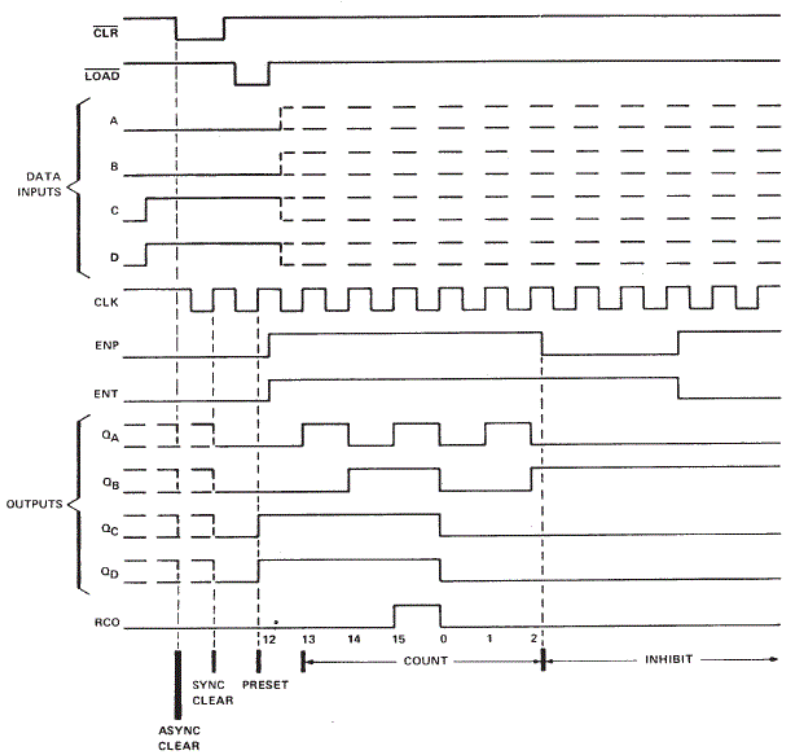
$$fim\_jogo = fim\_sequencia \wedge valor\_\max$$

Como “sincronizar” o primeiro elemento de uma sequência com outros?

Ignorar o estado 000 na vez da máquina! Na vez do jogador a referência inicia de 000 até valor máximo; e na vez da máquina, inicia com 001. As duas referências são multiplexadas.



7485, 74157 e 74160



74160: Contador Síncrono com CLEAR assíncrono

Function Table

Comparing Inputs				Cascading Inputs			Outputs		
A3, B3	A2, B2	A1, B1	A0, B0	A > B	A < B	A = B	A > B	A < B	A = B
A3 > B3	X	X	X	X	X	X	H	L	L
A3 < B3	X	X	X	X	X	X	L	H	L
A3 = B3	A2 > B2	X	X	X	X	X	H	L	L
A3 = B3	A2 < B2	X	X	X	X	X	L	H	L
A3 = B3	A2 = B2	A1 > B1	X	X	X	X	H	L	L
A3 = B3	A2 = B2	A1 < B1	X	X	X	X	L	H	L
A3 = B3	A2 = B2	A1 = B1	A0 > B0	X	X	X	H	L	L
A3 = B3	A2 = B2	A1 = B1	A0 < B0	X	X	X	L	H	L
A3 = B3	A2 = B2	A1 = B1	A0 = B0	H	L	L	H	L	L
A3 = B3	A2 = B2	A1 = B1	A0 = B0	L	H	L	L	H	L
A3 = B3	A2 = B2	A1 = B1	A0 = B0	L	L	H	L	L	H
A3 = B3	A2 = B2	A1 = B1	A0 = B0	X	X	H	L	L	H
A3 = B3	A2 = B2	A1 = B1	A0 = B0	H	H	L	L	L	L
A3 = B3	A2 = B2	A1 = B1	A0 = B0	L	L	L	H	H	L

H = High Level, L = Low Level, X = Don't Care

7485: comparador

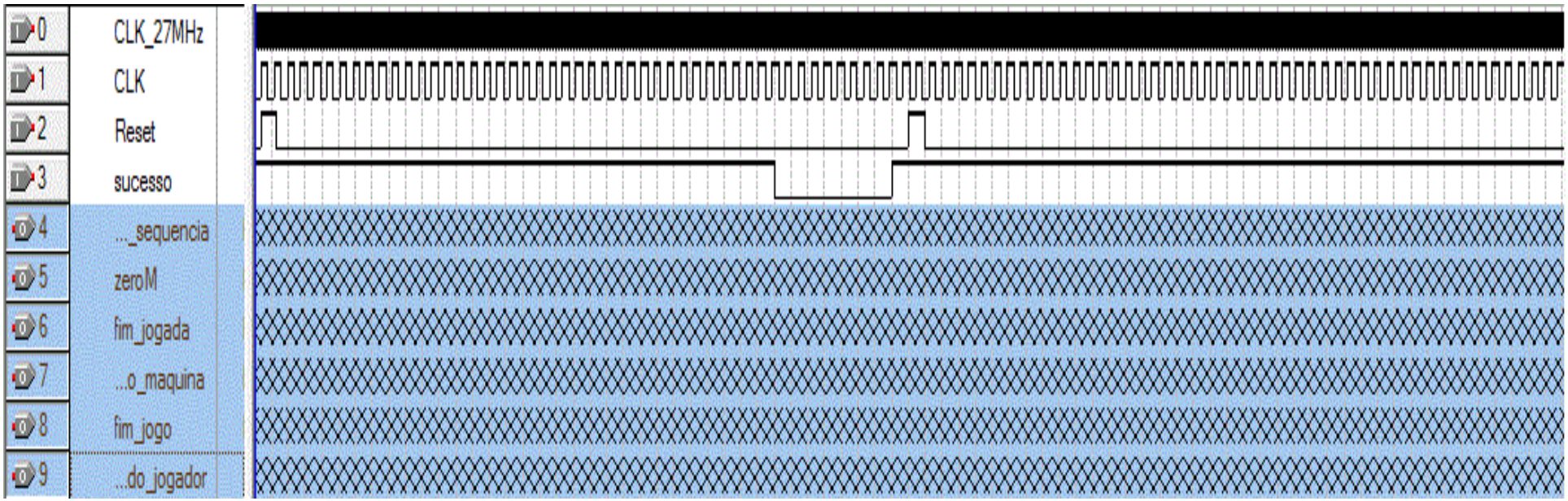
74157: multiplexador de *4 bits*

Inputs				Output Y
Strobe	Select	A	B	
H	X	X	X	L
L	L	L	X	L
L	L	H	X	H
L	H	X	L	L
L	H	X	H	H

H = High Level, L = Low Level, X = Don't Care

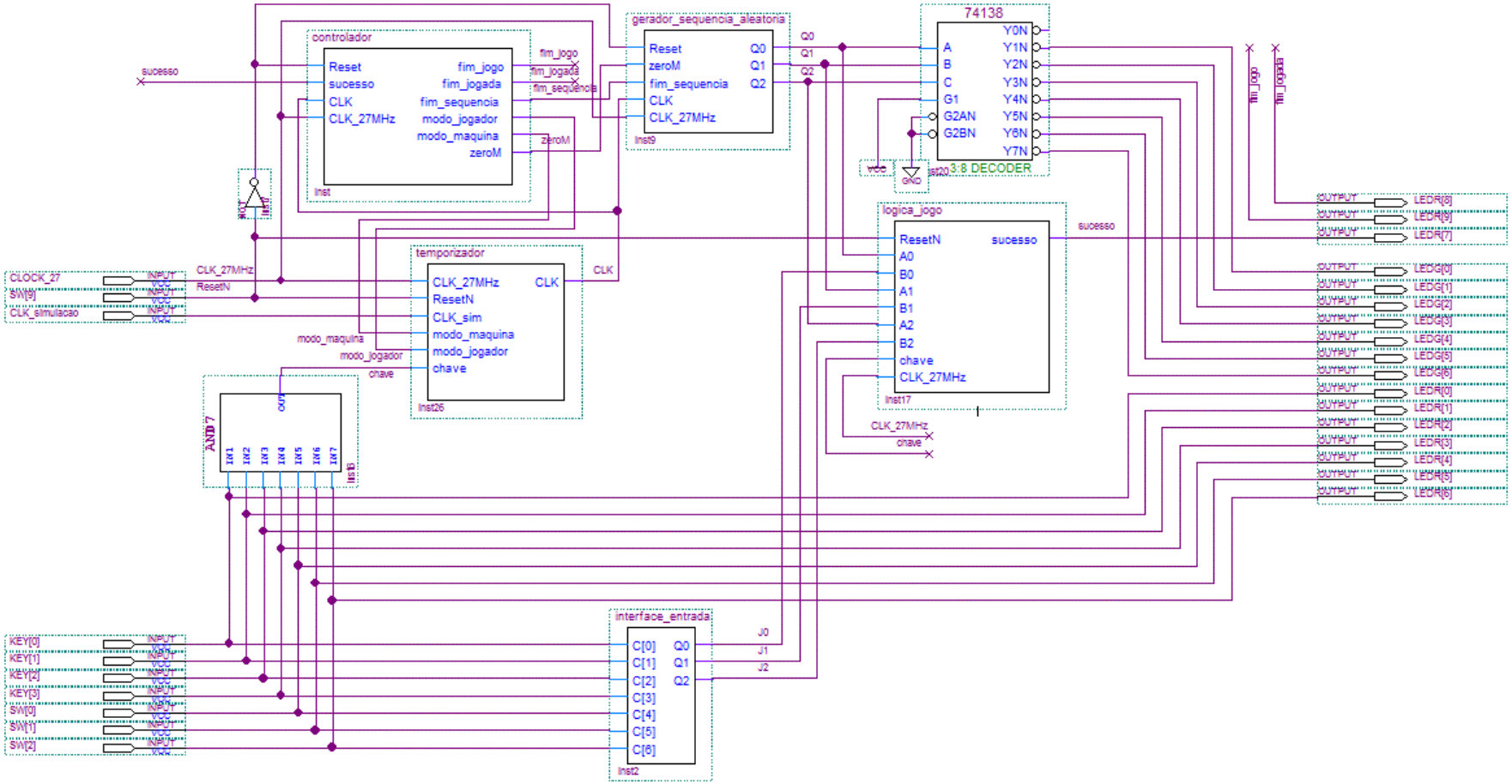
3^0. Experimento

1. Analise o circuito de controlador simulando-o (*Timing*) com as seguintes formas de onda



3^0. Experimento

2. Integre os blocos funcionais conforme o esquemático.



3. Edite as formas de onda para simular o comportamento do circuito
4. Apresente os resultados do projeto implementado em FPGA reproduzindo a sequência de ações simuladas através dos periféricos. ÇLK-simulacao deve ser substituído pelo sinal da saída do divisor de ferquência.

Interface de Saída

- Projete a interface de saída. Utilize a sua criatividade para mostrar os resultados com os periféricos disponíveis no *kit* FPGA-SDB
  - a jogada em que se encontra
  - os elementos de uma sequência gerada tanto pela máquina quanto pelo jogador
  - quando ocorre um erro
  - quando alcançaa sequência máxima sem erro.

4º. Experimento