instituto superior de engenharia de lisboa

Serial LCD Controller

4º Relatório do Projeto *Roulette Game*

Trabalho realizado por:

Gustavo Costa | Nº 52808

Ian Frunze | Nº 52867

Rafael Pereira | Nº 52880

Turma: **LEIC24D**

Docentes: **David Velez e Rui Duarte**

Licenciatura em Engenharia Informática e de Computadores

Laboratório de Informática e de Computadores (LIC)

2024 / 2025 - Semestre de Verão

Conteúdo

Lista de Figuras

1 Introdução

O módulo *Serial LCD Controller* (SLCDC) consolida-se como um componente essencial na arquitetura híbrida do sistema *Roulette Game*, atuando como interface crítica entre o subsistema de controlo (implementado em *software*) e o LCD. A sua concepção baseia-se em princípios de eficiência e robustez, permitindo uma comunicação série unidirecional que minimiza a complexidade das interconexões físicas, conforme ilustrado no diagrama de blocos da Figura **??**. A adoção de um protocolo de comunicação dedicado (Figura 10), composto por tramas de 5 bits de dados e 1 bit de paridade ímpar, assegura a integridade das informações transmitidas, mitigando riscos de corrupção de dados durante a transmissão.

O Serial Receiver, subcomponente do SLCDC, desempenha um papel fundamental na desserialização e validação das tramas. Ao reconstruir os dados em formato paralelo e aplicar um mecanismo de verificação de paridade ímpar (implementado no bloco Parity Check), garante-se que apenas tramas válidas sejam processadas, conforme validado empiricamente durante as simulações de temporização e fluxo de dados. Adicionalmente, o LCD Dispatcher opera em sincronia com os requisitos técnicos do fabricante do LCD, interpretando o bit RS para distinguir entre comandos de controlo e dados, e garantindo a entrega sequencial das informações ao display.

Os testes empíricos, realizados em ambiente de simulação, confirmaram a eficácia do módulo em cenários de carga variável. Observou-se a correta ativação dos sinais de controlo (D_{val}, `done`) e a reconstituição precisa dos dados, mesmo em condições de alta frequência de transmissão. A análise temporal, alinhada com o diagrama da Figura 10, comprovou a aderência aos tempos de estabilização e transição definidos no protocolo, reforçando a confiabilidade do SLCDC.

Do ponto de vista sistêmico, o SLCDC contribui diretamente para a experiência do utilizador, assegurando que informações críticas — como saldo de créditos, resultados de apostas e mensagens de operação — sejam exibidas com precisão e latência mínima. Sua integração harmoniosa com o módulo Control (implementado em Kotlin) demonstra a viabilidade de soluções híbridas (hardware/software) em aplicações de tempo real, conforme preconizado na arquitetura lógica da Figura 14.

Em síntese, a implementação e validação do SLCDC validaram não apenas seu cumprimento funcional — incluindo a gestão de tramas e interação com periféricos —, mas também sua relevância estratégica no projeto. Ao garantir comunicação robusta e eficiente, este módulo consolida-se como pilar para a operação integrada do Roulette Game, refletindo a importância de uma abordagem metodológica rigorosa no desenvolvimento de sistemas embarcados.

2 Objetivo do Módulo SLCDC

O principal objetivo do módulo *Serial LCD Controller* (SLCDC) é implementar uma interface de comunicação eficiente, fiável e simples entre o *software* e o display LCD. Este módulo permite ao *software* enviar comandos e dados, em série, que são posteriormente interpretados e encaminhados para o LCD em paralelo, obedecendo ao protocolo de funcionamento do próprio *display*.

A estrutura do SLCDC foi concebida para ser modular e clara, sendo composta por dois blocos principais:

- O *Serial Receiver*, que realiza a receção bit a bit da informação enviada, valida a integridade da trama (através do bit de paridade ímpar (`P`)) e a converte para formato paralelo;
- O *LCD Dispatcher*, que trata a informação validada, distingue entre comandos e dados (com base no bit `RS`), e aciona os sinais de escrita para que o LCD registe a informação corretamente.

O protocolo de comunicação é essencial para este módulo visto que ele terá de comportar uma trama recebida bit a bit, respeitando tempos definidos entre cada transição de sinal, que necessitam de ser cumpridos. Logo, é muito importante recorrer e tomar como base um protocolo claro e que consegue comportar todas as características a que o SLCDC diz respeito. Neste caso, o protocolo foi disponibilizado no enunciado e será analisado na próxima secção.

3 Protocolo de Comunicação com o SLCDC

Antes de explorar os blocos internos, é importante compreender o formato da trama com que o SLCDC comunica.

Segundo a figura mostrada de seguida, o protocolo de comunicação série com o SLCDC é composto por uma trama de 6 bits, distribuída da seguinte forma:

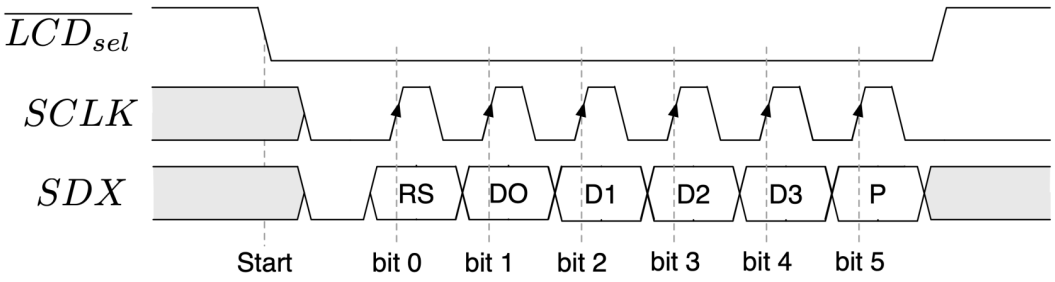

Figura 1: Protocolo de comunicação com o módulo SLCDC

- Bit 0 (RS) - Indica o tipo de mensagem: Se `RS = 0` então trata-se de um comando para o LCD já se `RS = 1` então trata-se de dados a serem apresentados no ecrã.
- Bits 1 a 4: Transportam os 4 bits de dados (*nibble*) que constituem o conteúdo real da mensagem.
- Bit 5: Bit de paridade ímpar, usado para deteção de erros de transmissão.

As mensagens são enviadas pelo módulo de controlo através de duas linhas de comunicação:

- `LCDsel` - Sinal de seleção (ativa a comunicação com o SLCDC).
- `SCLK` - Clock de série, usado para sincronizar bits.

O início de uma trama é marcado por uma transição descendente no sinal `LCDsel`. A receção dos dados é sincronizada com as transições ascendentes de `SCLK`.

É importante notar também que, entre cada envio de bit, antes e depois, é preciso respeitar determinados tempos para não ocorrer anomalias. Respeitando-os garantimos que tudo o que irá ser fruto da transmissão de bits apresentada na Figura 1, será enviado e recebido de forma correta e segura. Como base para relacionar os tempos pedidos pelo fabricante, foi-nos disponibilizada uma tabela, juntamente com um diagrama temporal informativo, que relatam totalmente os tempos a serem respeitados, relativamente ao LCD.

Serão apresentados de seguida, o diagrama temporal e a tabela, respetivamente, disponibilizados na plataforma Moodle no `pdf` com o nome de `"Slides 5ª Aula"`:

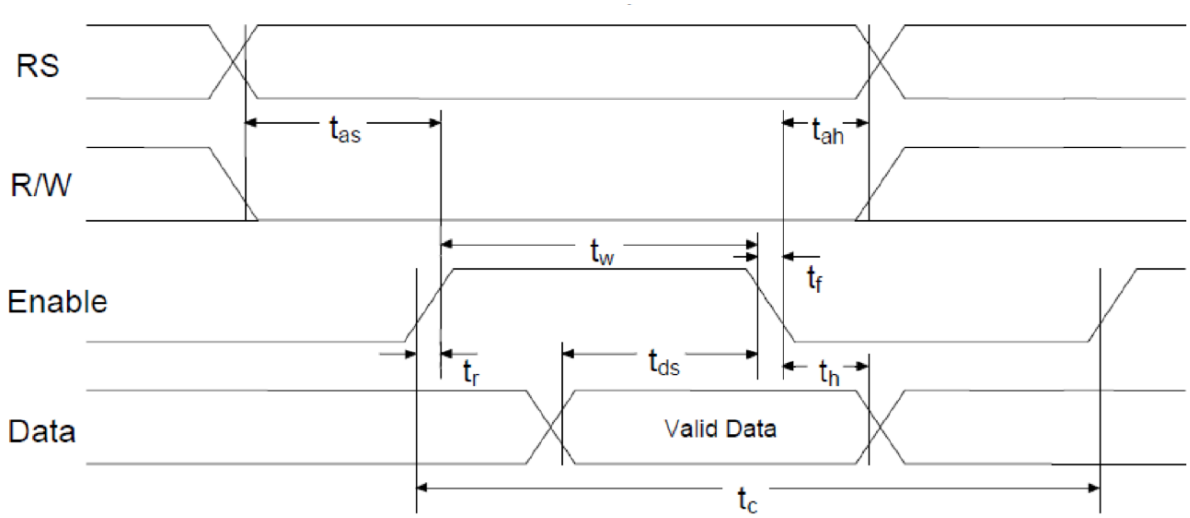

Figura 2: Diagrama temporal com indicações de tempos `Min/Max` nas funções do LCD

Write cycle

Parameter	Symbol	Min (1)	Typ (1)	Max (1)	Unit
Enable Cycle Time	t_c	500	-	-	ns
Enable Pulse Width (High)	t_w	230	-	-	ns
Enable Rise/Fall Time	t_r,t_f	-	-	20	ns
Address Setup Time	t_{as}	40	-	-	ns
Address Hold Time	t_{ah}	10	-	-	ns
Data Setup Time	t_{ds}	80	-		ns
Data Hold Time	t_h	10	-	-	ns

Figura 3: Tabela com os tempos definidos pelo fabricante

4 Módulo *Serial LCD Controller*

Conforme está descrito na arquitetura do sistema, este módulo é constituído por duas componentes, o *Serial Receiver* e o *LCD Dispatcher*, tal como é possível observar na Figura 4. Iremos explicar e aprofundar as análises sobre estes dois blocos que constituem o módulo SLCDC.

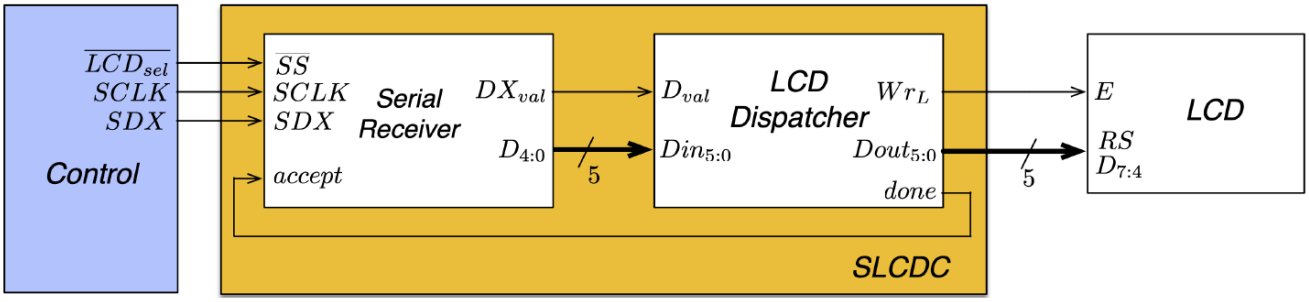

Figura 4: Módulo *Serial LCD Controller*

Este módulo desempenha um papel fundamental na interação entre o sistema de controlo do jogo (implementado em *software*) e o LCD, garantindo a apresentação clara e precisa de informações críticas ao jogador, como o saldo de créditos, resultados de apostas, mensagens de estado e, futuramente, dados de manutenção. Também é responsável por receber dados em série enviados pelo módulo *Control* (executado num PC) e convertê-los em comandos ou caracteres legíveis no ecrã LCD da *Expansion Board* da placa DE10-Lite, de 2 linhas e 16 caracteres.

A sua operação baseia-se num protocolo de comunicação específico, onde cada trama é composta por 5 bits de dados (incluindo um bit `RS` para distinguir entre comandos e dados) e 1 bit de paridade ímpar, utilizado para detetar erros de transmissão. A comunicação inicia-se com um sinal de `Start` (transição descendente no sinal LCD_{sel}, pois é *active-low*), seguido da sincronização dos bits de dados através de pulsos de clock (`SCLK`). Esta abordagem é eficaz e simplifica a arquitetura do sistema.

O SLCDC é vital para:

- A exibição de informação em tempo real, como atualizações de saldo após a "inserção"de moedas, resultados de apostas e números sorteados pela roleta.
- Futuramente, para suporte ao modo de manutenção, permitindo a visualização de contadores de moedas, jogos realizados e listas de números sorteados.
- A garantia de confiabilidade, através da verificação de paridade, assegurando que dados corrompidos não afetem a integridade da informação apresentada no instante no LCD.

Integrado com o módulo Control via *software* (em *Kotlin*), o SLCDC atua como uma ponte eficiente entre a lógica do jogo e a interface do jogador, sendo indispensável para a experiência interativa e funcionalidades essenciais do projeto global (*Roulette Game*).

5 *Serial Receiver*

5.1 Objetivo

O bloco *Serial Receiver* é responsável por receber os bits da trama, organizá-los em paralelo e verificar a sua validade com base na paridade. É a primeira etapa de receção dos dados vindos do módulo de controlo (*Control*).

5.2 Estrutura Interna

O *Serial Receiver* é constituído por quatro sub-blocos principais: *Serial Control*, *Shift Register*, *Counter* e *Parity Check*.

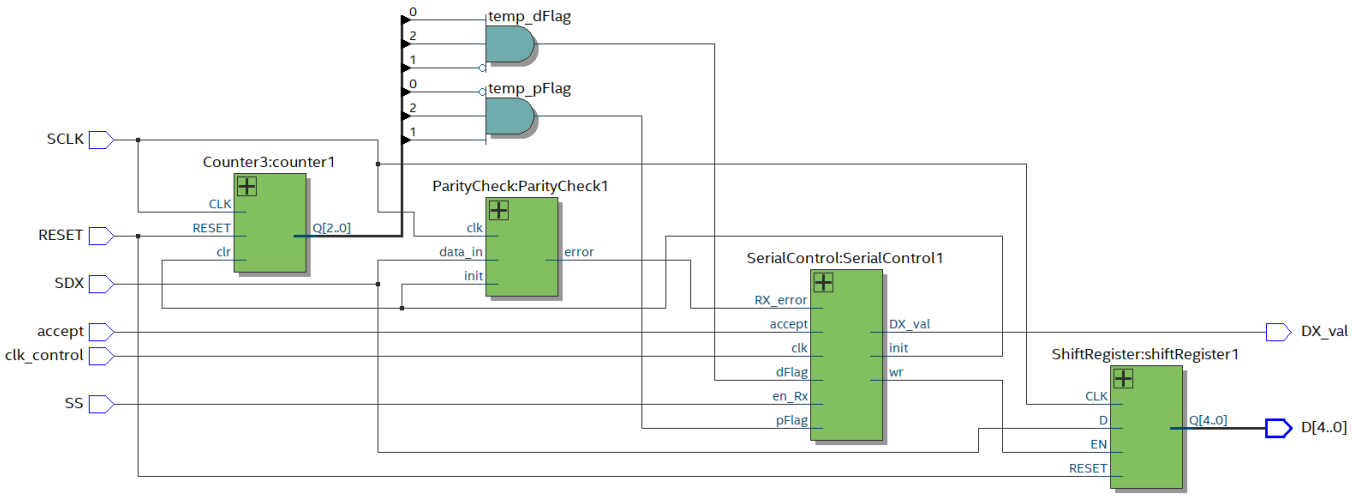

Figura 5: Ferramenta *RTL Viewer* no programa Quartus com o aspeto interior do *Serial Receiver*

5.2.1 Serial Control

É o componente responsável por gerir todo o processo de receção da trama. Ele coordena o funcionamento dos restantes sub-blocos, baseando-se nos sinais externos.

Ele tem como funções principais:

- Detetar o início da comunicação através da transição descendente do sinal LCD_{sel}, como mostrado na Figura 1. Esta ação marca o início da recação de uma nova trama.
- Sincronizar os ciclos de leitura de bits com as transições ascendentes do sinal `SCLK`, que indicam a disponibilidade de um novo bit na linha de dados.
- Controlar o carregamento do *Shift Register*: sempre que deteta um clock válido, envia um sinal de *enable* para deslocar os bits no registo de deslocamento.
- Sinalizar o fim da trama: após a receção dos 6 bits (1 bit RS, 4 bits de dados, 1 bit de paridade), ativa um sinal interno que indica que a trama está completa e pronta para ser validada.

Segue-se uma máquina de estados (Figura 6) que reflete o comportamento efetuado por este bloco.
Analisando-a detalhadamente, concluímos:

- Estado `000` (`init`) - Inicialização: Aguarda o início de uma transmissão (detecção da condição `Start`, como uma transição descendente em LCD_{sel} ou R_{sel}). Transições: Quando a condição de início é detectada, avança para o estado `001`.

- Estado `001` - Ativação da Recepção: `en_rx` (enable de recepção) é ativado para iniciar a captura dos bits. O `wr` (sinal de escrita) é mantido em '0', indicando que o dado ainda não está pronto. A `d_flag` (*flag* de dado) e `p_flag` (*flag* de paridade) são desativados (0). Transições: Após a captura dos bits, avança para o estado `010`.

- Estado `010` - Validação de Dados: `d_flag` é ativado (1), indicando que os dados recebidos estão disponíveis. O `p_flag` é ativado (1) para verificar a paridade ímpar (conforme protocolos SLCDC/SRC). O `rx_error` é mantido em '0', indicando que nenhum erro foi detectado até o momento. Transições: Se a paridade for válida, avança para o estado `011`. Se a paridade for inválida, retorna ao estado `000` ou gera um erro (não explícito na ASM).

- Estado `011` - Verificação Final: `DX_val` (validação do dado) é verificado: se `DX_val` = 0 (dado inválido), ativa `rx_error` (indicando um erro na transmissão), caso contrário, se `DX_val` = 1 (dado válido), avança para o estado `100`. Transições: Em caso de erro, retorna ao estado 000 para reiniciar o processo, já em caso de sucesso, prossegue para a confirmação.

- Estado `100` - Confirmação de Receção: `accept` é ativado (1) para informar ao sistema consumidor que a trama foi recebida e validada. Após a confirmação, `accept` é desativado (0), libertando o canal para uma nova transmissão. Transição: Retorna sempre ao estado `000` para aguardar uma nova transmissão.

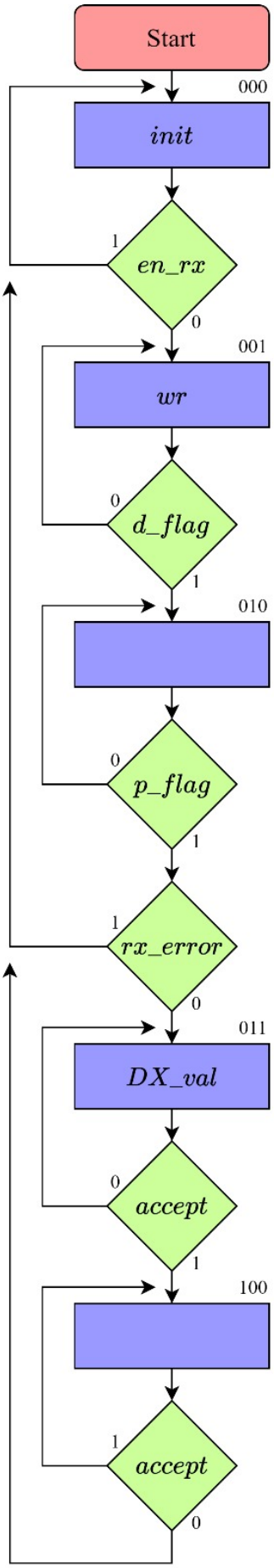

Figura 6: Máquina de estados referente ao comportamento do bloco *Serial Control*

O *Serial Control*, portanto, garante que o sistema apenas tenta processar uma trama quando todos os bits foram corretamente capturados e armazenados.

5.2.2 Shift Register

O *Shift Register* é o componente que armazena sequencialmente os bits recebidos, convertendo o sinal série (bit a bit) num formato paralelo compreensível pelos blocos seguintes.

A cada transição ascendente do sinal `SCLK`, o bit presente na linha de dados é empurrado para dentro do registo, deslocando os bits já recebidos para a esquerda.

Após 6 ciclos de clock válidos, o registo contém:

- Bit 5 – Bit de paridade.
- Bits 4:1 – *Nibble* de dados (4 bits).
- Bit 0 – Bit `RS`, que indica se a mensagem é de comando (0) ou dados (1).

Este registo é fundamental para que a comunicação em série seja interpretada corretamente. Sem este mecanismo, seria impossível capturar e estruturar os bits enviados individualmente.

Além disso, a utilização de um *shift register* permite um *design* simples em termos de lógica de *hardware*.

5.2.3 Counter

O *Counter* é o sub-bloco que acompanha o número de bits recebidos durante a transmissão de uma trama.

Este módulo inicia a contagem assim que o *Serial Control* deteta o início de uma nova trama (`LCDsel` desce). Incrementa também a contagem a cada ciclo de `SCLK` válido. Quando o counter atinde o valor 6 (número total de bits da trama), sinaliza que a trama está completa e este sdinal de "trama completa"é fundamental para ativar o bloco *Parity Check* e para notificar o sistema que os dados já podem ser processados.

Graças ao *Counter*, o sistema garante que apenas são processadas tramas completas, o que evita erros causados por interrupções, perda de sincronismo ou falhas de comunicação.

5.2.4 Parity Check

O *Parity Check* é o bloco responsável por garantir a integridade dos dados recebidos.

O seu funcionamento é o seguinte:

- Calcula a paridade doas 5 primeiros bits recebidos (`RS` + 4 bits de dados).
- Compara o resultado com o bit de paridade (6.º bit da trama), que foi enviado pelo módulo de controlo.
- O protocolo especificado no enunciado utiliza paridade ímpar. Isso significa que o núemro total de bits '1' em toda a trama (incluindo o bit de paridade) deve ser ímpar.
- Se a paridade for válida, o sinal `Dval` (*data valid*) é ativado. Este sinal informa o bloco *LCD Dispatcher* que os dados são seguros e podem ser processados.
- Se a paridade for inválida, a trama é descartada silenciosamente, protegendo o sistema de apresentar informação, desnecessária e errada, no ecrã.

Este mecanismo de validação é essencial para assegurar que o *display* LCD só apresenta dados corretos, independentemente das condições de interferência e ruído possíveis.

6 *LCD Dispatcher*

O *LCD Dispatcher* é um componente essencial no módulo *Serial LCD Controller* (SLCDC)que é encarregue de interpretar e encaminhar tramas válidas recebidas pelo *Serial Receiver* para o *display* LCD. Atua como a ponte final entre os dados recebidos em série e a sua apresentação no dispositivo de saída, assegurando que a comunicação com o LCD ocorre de forma rápida, fiável e compatível com o protocolo do fabricante.

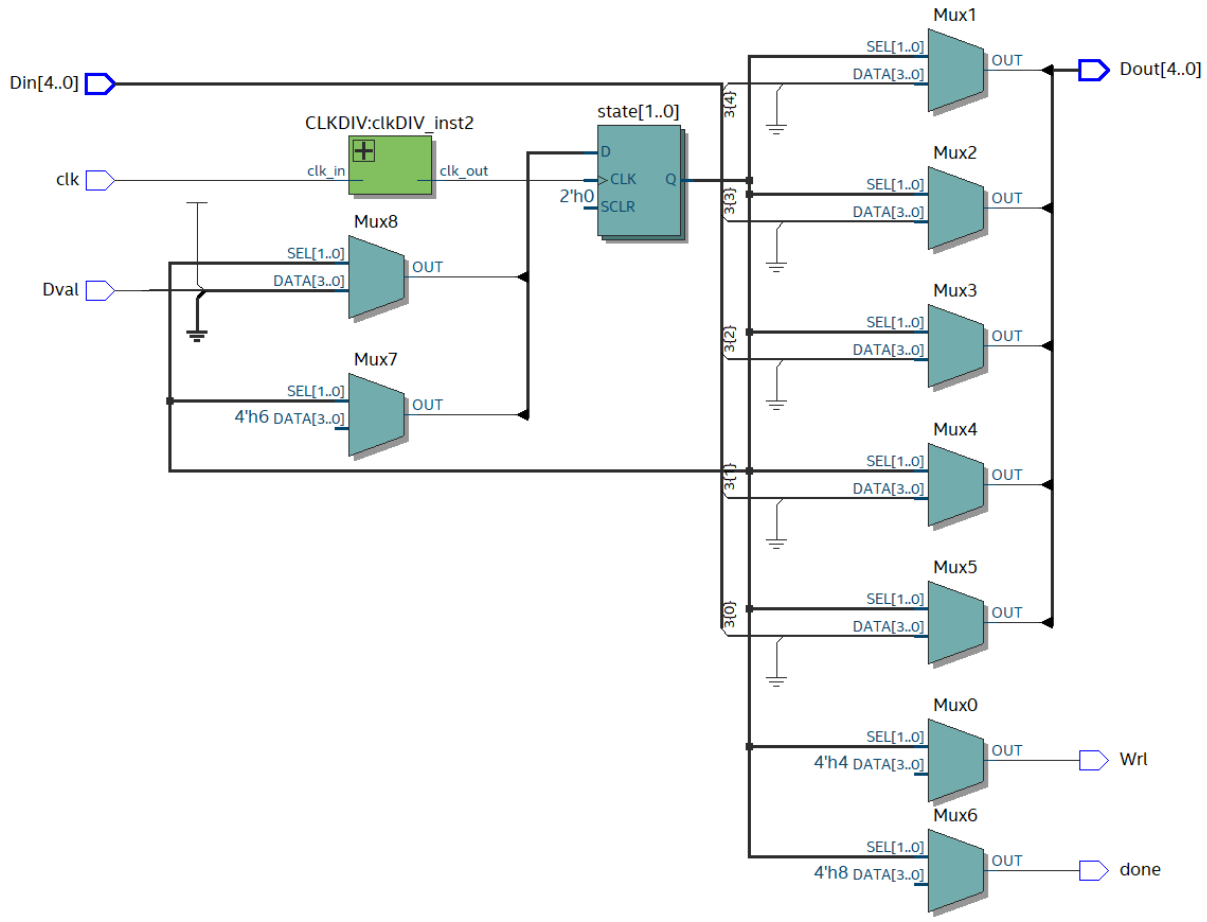

Figura 7: Ferramenta *RTL Viewer* no programa Quartus com o aspeto interior do *LCD Dispatcher*

A sua função principal é:

- Interpretar o conteúdo da trama recebida, distinguindo entre comandos de controlo e dados a exibir.
- Acionar o sinal de escrita no LCD (`WrL`), enviando o conteúdo desejado.
- Notificar o *Serial Receiver* que a trama foi processada com sucesso, através do sinal `done`.
- Assegurar que cada nova trama é tratada individualmente e sem colisões, mantendo o fluxo contínuo de receção e envio.

Este bloco é indispensável porque garante que nenhuma trama inválida é escrita no ecrã, já que apenas atua quando o sinal D_{val} está ativo — ou seja, quando a trama passou com sucesso o teste de paridade.

6.1 Funcionamento Interno

Este bloco baseia-se no comportamento de uma máquina de estados que é apresentada de seguida.
Analisando-a:

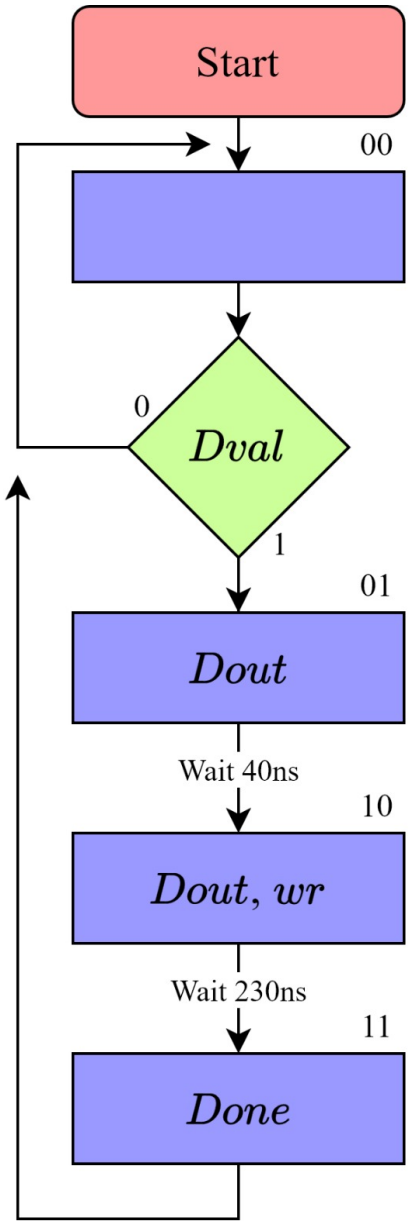

Figura 8: Máquina de estados referente ao comportamento do Módulo *LCD Dispatcher*

- `Start`: Inicia a recepção ao detectar a condição de início (transição descendente em LCD_{sel} ou R_{sel}).
- Estado `00`: Aguarda a disponibilidade de dados.
- Estado `01`: Ativa D_{val} (dado válido) e disponibiliza D_{out} (dado recebido).
- Estado `10`: Realiza a escrita (`wr`) e aguarda $230ns$ (tempo de processamento).
- Estado `11`: Sinaliza `Done` (conclusão do processamento da trama).

Os tempos de $40ns$ e $230ns$, são cumpridos de acordo com as configurações e especificações do fabricante. Significam o tempo de espera para sincronização ou estabilização dos bits e o tempo de processamento do comando (ex: envio ao LCD ou *Roulette Display*), respetivamente.

7 Testes e Simulação

7.1 Objetivo dos Testes

O objetivo da simulação foi validar o funcionamento correto dos componentes do módulo *Serial Receiver* do *Serial LCD Controller* (SLCDC), com especial atenção ao registo de deslocamento (*Shift Register*) e ao processo de validação da trama recebida. Estes testes asseguram que a interface de comunicação em série entre o módulo de controlo (em *software*) e o LCD (em *hardware*) funciona corretamente, respeitando o protocolo definido.

7.2 Ambiente de Simulação

Os testes foram realizados com recurso à ferramenta *ModelSim*, integrada no *Quartus*, utilizando a simulação RTL. Foi desenvolvida uma *testbench* em VHDL para gerar os sinais de entrada (`SCLK, SS, SDX`) de acordo com o protocolo série implementado. O módulo foi simulado isoladamente para facilitar a verificação das suas funcionalidades internas.

7.3 Testes ao *Serial Receiver*

Mostramos de seguida a captura do *RTL Simulation* relativamente ao módulo *Serial Receiver*:

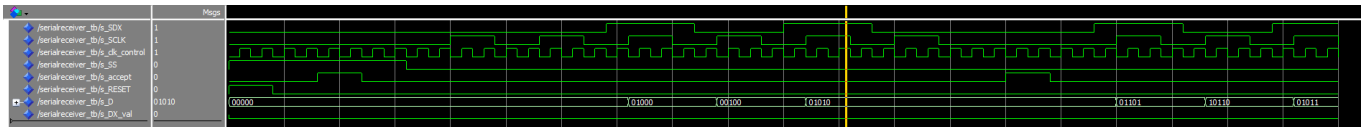

Figura 9: Ferramenta *RTL Simulation* com *testbench* (*SerialReceiver_tb.vhd*) foi executada

O *Serial Receiver* foi testado quanto à sua capacidade de:

- Iniciar a receção após a transição descendente do sinal `SS` (*Start*).
- Contar corretamente o número de bits recebidos (através do *Counter*).
- Validar a paridade ímpar da trama (através do *Parity Check*).
- Sinalizar quando uma trama válida está pronta (D_{val}).
- Aguardar a confirmação do consumidor (`accept`) antes de voltar ao estado inicial.

Durante a simulação anterior, observou-se que, após a transição de `SS` de 1 para 0, o módulo entra no estado de receção e ativa internamente o controlo do deslocamento. Também é percetível que a cada transição positiva do sinal `SCLK`, um novo bit é deslocado para o *Shift Register*. Após a receção de todos os bits da trama (6 bits), o *Counter* atinge o valor esperado e aciona o bloco de verificação de paridade. Se a paridade estiver correta, o sinal D_{val} é ativado, permitindo ao *LCD Dispatcher* reconhecer que a trama está disponível. E finalmente, ao receber o sinal `accept`, o *Serial Receiver* reinicia o processo e fica pronto para receber a próxima trama.

7.4 Testes ao *LCD Dispatcher*

O *LCD Dispatcher* é responsável por receber tramas válidas do *Serial Receiver* e entregá-las ao LCD. Após a receção de uma trama válida (sinal D_{val} ativo), este bloco interpreta a natureza da informação (controlo ou dados) e aciona o sinal `WrL` para a escrita no LCD. Em seguida, notifica o *Serial Receiver* de que a trama foi processada, ativando o sinal `done`.

Na simulação, foi possível verificar que:

- Quando o sinal D_{val} do *Serial Receiver* é ativado, o *LCD Dispatcher* interpreta corretamente o bit `RS` da trama (bit mais significativo), distinguindo entre comando (`RS=0`) e dados (`RS=1`).

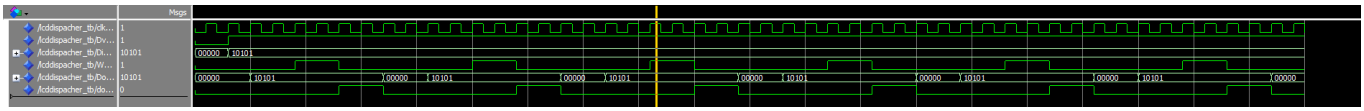

Figura 10: Ferramenta *RTL Simulation* com *testbench* (*LCDDispacther_tb.vhd*) foi executada

- O sinal `WrL` é ativado brevemente para realizar a escrita no LCD.
- Logo após a escrita, o sinal `done` é ativado, informando o *Serial Receiver* de que a trama foi consumida. Isto faz com que o sinal D_{val} seja desativado e o recetor esteja pronto para uma nova receção.
- Não foi observado nenhum atraso excessivo na propagação dos sinais, o que indica uma boa sincronização entre os blocos.

8 Conclusão

O módulo *Serial LCD Controller* (SLCDC) consolida-se como um componente essencial na arquitetura híbrida do sistema *Roulette Game*, atuando como interface crítica entre o subsistema de controlo (implementado em *software*) e o LCD. A sua concepção baseia-se em princípios de eficiência e robustez, permitindo uma comunicação série unidirecional que minimiza a complexidade das interconexões físicas, conforme ilustrado no diagrama de blocos da Figura 4. A adoção de um protocolo de comunicação dedicado (Figura 1), composto por tramas de 5 bits de dados e 1 bit de paridade ímpar, assegura a integridade das informações transmitidas, mitigando riscos de corrupção de dados durante a transmissão.

O *Serial Receiver*, subcomponente do SLCDC, desempenha um papel fundamental na desserialização e validação das tramas. Ao reconstruir os dados em formato paralelo e aplicar um mecanismo de verificação de paridade ímpar (implementado no bloco *Parity Check*), garante-se que apenas tramas válidas sejam processadas, conforme validado empiricamente durante as simulações de temporização e fluxo de dados. Adicionalmente, o *LCD Dispatcher* opera em sincronia com os requisitos técnicos do fabricante do LCD, interpretando o bit `RS` para distinguir entre comandos de controlo e dados, e garantindo a entrega sequencial das informações ao *display*.

Os testes empíricos, realizados em ambiente de simulação, confirmaram a eficácia do módulo em cenários particulares. Observou-se a correta ativação dos sinais de controlo (D_{val}, `done`) e a reconstituição precisa dos dados. A análise temporal comprovou a aderência aos tempos de estabilização e transição definidos no protocolo, reforçando a confiabilidade do SLCDC.

Do ponto de vista sistémico, o SLCDC contribui diretamente para a experiência do jogador, assegurando que informações críticas — como saldo de créditos, resultados de apostas e mensagens de operação — sejam exibidas com precisão e latência mínima. A sua ua integração com o módulo *Control* (implementado em *Kotlin*) demonstra a viabilidade de soluções híbridas (*hardware*/*software*) em aplicações de tempo real, conforme preconizado na arquitetura geral do *Roulette Game*.

Em síntese, a implementação e validação do SLCDC validaram não apenas seu cumprimento funcional — incluindo a gestão de tramas e interação com periféricos —, mas também sua relevância no projeto. Ao garantir uma comunicação robusta e eficiente, este módulo consolida-se como suporte para a operação integrada do *Roulette Game*.