Projeto UART Full-Duplex com CRC

1. Definição dos Módulos do Circuito

O projeto será dividido em módulos menores, cada um com uma função específica. Esta abordagem modular é fundamental em HDL e permite que cada parte seja testada individualmente.

- **uart_tx (Transmissor Serial):** Recebe um byte de 8 bits e o converte em um fluxo de bits serial. Ele adiciona os bits de start e stop para definir o início e o fim da transmissão.
  - **Entradas:** clock, reset, tx_data (8 bits), start_tx.
  - **Saídas:** tx_serial_out (1 bit), tx_busy.
- **uart_rx (Receptor Serial):** Fica "escutando" o fio serial. Quando detecta um bit de start, ele lê os próximos bits e os monta em um byte completo, sinalizando que a mensagem está pronta.
  - **Entradas:** clock, reset, rx_serial_in (1 bit).
  - **Saídas:** rx_data (8 bits), data_ready, rx_busy.
- **crc_gerador (Gerador de CRC):** Implementa a lógica do CRC (registrador de deslocamento com XORs). Ele recebe a mensagem e calcula o valor de verificação, que será enviado junto com a mensagem.
  - **Entradas:** clock, reset, crc_data_in (8 bits), data_valid.
  - **Saídas:** crc_value (o valor do CRC, por exemplo, 8 bits).
- **crc_verificador (Verificador de CRC):** Recebe a mensagem e o CRC enviado. Ele recalcula o CRC da mensagem e o compara com o valor recebido.
  - **Entradas:** clock, reset, check_data_in (8 bits), check_crc_in (8 bits), data_valid.
  - **Saídas:** crc_error_flag (1 bit).
- **top_level (Módulo Principal):** Este é o "cérebro" do projeto. Ele instanciará todos os módulos acima e os conectará. No caso de um sistema full-duplex, ele terá dois pares de uart_tx e uart_rx para a comunicação em ambas as direções. Ele também gerenciará a lógica para passar dados para os módulos de CRC e para decidir o que fazer com base no sinal de erro.

---

2. Fluxo de Trabalho do Projeto em Verilog

1. **Escreva o Código:** Comece criando o código para os módulos menores (uart_tx, uart_rx, crc_gerador, etc.). Teste cada um individualmente com um testbench simples para garantir que funcionam.
2. **Crie o top_level:** Junte todos os módulos no arquivo principal (top_level.v), instanciando-os e conectando-os corretamente.

3. **Crie o Testbench do Sistema:** Escreva o testbench final para o top_level. Este testbench irá:
    - Gerar o sinal de clock.
    - Fornecer os dados que serão enviados.
    - Simular a comunicação full-duplex, enviando e recebendo mensagens.
    - **Inserir erros intencionalmente** na simulação para testar se o verificador de CRC funciona.
4. **Compile e Simule:** Use o Icarus Verilog para compilar e simular todo o projeto.
   ```
   Bash
   iverilog -o top_level.out top_level.v uart_tx.v ...
   vvp top_level.out
   ```
5. **Visualize os Resultados:** Use o GTKWave para abrir o arquivo de simulação (.vcd). Observe as formas de onda de todos os sinais para verificar se a lógica está correta. O "display" do seu projeto será o sinal crc_error_flag no GTKWave, mostrando o status da comunicação.

Por que não usar um Simulador de Circuitos?

Embora simuladores de circuitos gráficos como o Falstad sejam ótimos para entender a lógica básica de portas e flip-flops, eles não são a ferramenta adequada para um projeto desta complexidade.

1. **Complexidade e Escala:** Um projeto como o UART com CRC envolveria centenas, talvez milhares de portas lógicas e flip-flops. Tentar desenhar e conectar cada componente visualmente resultaria em um diagrama impossível de gerenciar e depurar.
2. **Nível de Abstração:** O Verilog opera em um nível de abstração muito maior. Em vez de se preocupar com cada fio e porta, você descreve o **comportamento** do circuito com código (por exemplo, "se o sinal dados_prontos for 1, salve os dados"). Isso é muito mais eficiente para projetos grandes.
3. **Depuração e Teste:** Em Verilog, o testbench permite testar o circuito de forma automatizada e repetitiva. Injetar erros e verificar a saída em diferentes cenários é trivial. Em um simulador gráfico, você teria que mudar manualmente as entradas e observar os resultados a cada passo, o que é inviável para testes completos.
4. **Padrão da Indústria:** O uso de HDL é o padrão na engenharia de sistemas digitais. Ao desenvolver o projeto em Verilog, você está usando as mesmas ferramentas e métodos que são aplicados em chips e FPGAs reais, o que torna o aprendizado muito mais relevante e profissional.