Universidade de São Paulo

Escola Politécnica, Engenharia de Computação
PCS3645 - Laboratório Digital II
Turma 3 - Professor Paulo Cugnasca
Bancada B6

Experiência 03

Interface com Sensor Ultrassônico de Distância – Planejamento

Nome Completo	**N USP**
Henrique Freire da Silva	12555551
Mariana Dutra Diniz Costa	12550841

São Paulo, 18 de Setembro de 2023

Sumário

1 INTRODUÇÃO

No presente relato, será apresentado o projeto de um circuito digital para interfacear um sensor ultrassônico (modelo HC-SR04) na interpretação de leituras de pequenas distâncias.

Com o objetivo de síntese e contínuo desenvolvimento desse módulo para demais experiências, as seguintes seções compreendem uma etapa de familiarização com a lógica de demodulação do circuito, seguida da aplicação em VHDL, simulação com o *ModelSim*, síntese com o *Quartus Prime* e, finalmente, verificação por meio de testes práticos.

1.1 Requisitos Funcionais

A cada medição, o circuito deve registrar o valor modulado pelo sensor HC-SR04 e apresentá-lo em *displays* de sete segmentos. A interface de interação do usuário deve ser realizada através da placa programada, de forma que medidas e *resets* possam ser acionados por chaves.

1.2 Requisitos Não Funcionais

A arquitetura proposta deve descrever um modelo estrutural com módulos de fluxo de dados e unidade de controle. O circuito deve ser de fácil depuração, em que a representação da medida amostrada deve estar na base decimal (codificação BCD) e em *cm*. Ademais, o projeto deve ser descrito em VHDL e sintetizado pelo *Quartus Prime* para a placa DE0-CV.

2 DESCRIÇÃO DO PROJETO

Um sensor ultrassônico é um dispositivo que emite ondas de alta frequências para estipular a posição relativa de um objeto no qual seu sinal reflete.

Sabendo a velocidade com que o sinal se propaga em um ambiente controlado (aproximadamente $340m/s$ a 15^{o}C) pode-se estabelecer a relação $2D = T \times V$, em que D $[m]$ é a distância entre o sensor e o objeto; T $[s]$ é o tempo entre o envio e a recepção do sinal; e V $[m/s]$ a velocidade transversal da onda.

Para o caso de uso a curtas distâncias, torna-se mais interessante a adoção das unidades em cm e μs. Dessa forma, tem-se:

$$2D = T \times V \times 10^{-4} \Rightarrow D = T \times 170 \times 10^{-4} \Rightarrow D = T \times 0.017 \Rightarrow D \approx \frac{T}{58.82}[cm] \qquad (1)$$

A partir disso, é possível gerar um sinal que modele a distância percebida pelo sensor e a codifique pela modulação do sinal (PWM). Sabendo dessa característica do sensor, é possível gerar um circuito que interprete o pulso gerado e que armazene o dado para posterior uso.

O projeto dessa interface entre o sensor e qualquer dispositivo que vá fazer uso da informação de distância é discorrido nas seguintes seções, partindo do estudo incial de partes fornecidas até desenvolvimento e teste de demais fluxos.

2.1 Projeto do Circuito de Interface

A entidade principal de interface desenvolvida consiste de uma arquitetura que conecta o seu fluxo de dados e unidade de controle às portas declaradas (Apêndice A.1).

Um componente adicional *contador_cm* foi criado de forma a abstrair a funcionalidade de decodificar um valor numérico de distância a partir de um pulso genérico de largura L (entidade vide Apêndice A.4). Nele, é instanciado um *contador_bcd* para apresentar os 3 dígitos em base decimal, um *contador_m* (contador modular) para gerar o *tick* de contagem para o contador anterior, e um detector de borda; esses componentes se encontram, respectivamente, nas linhas 47, 59 e 73 do Apêndice B.1.

O sinal *echo* gerado pelo sensor e recebido pelo *contador_cm* é usado para produzir o *tick* a partir da saída "meio"a cada 2941 ciclos de *clock* (período de 20 ns), tal que se o período entre *ticks* corresponda ao valor teórico em μs apresentado como divisor da Equação 1. Ademais, o uso da saída mediana de contagem serve ao propósito de arredondar o número ao inteiro mais próximo, antecipando a contagem à meia unidade.

Esse sinal de *tick* gerado é utilizado para realizar a contagem BCD, que por sua vez gera os dígitos exportados pela interface. Por fim, o sinal "fim", indicador do término da medida, é gerado pela borda de descida do pulso recebido (*echo* do sensor). Isso se dá pela inversão desse sinal e inserção do mesmo no detector de borda, já que a troca de nível lógico implica em inversão das transições.

2.1.1 Unidade de controle

A unidade de controle fornecida foi estudada e adotada integralmente no projeto da interface. A entidade desse módulo segue o Apêndice A.2.

A transição de estados de execução segue o algoritmo seguinte, descrito em linguagem natural:

```
* Enquanto não receber o pulso MEDIR, aguarda o ínicio da medida;
* Inicia a preparação para medida, resetando os componentes internos do sistema;
```

```
* Gera um pulso de TRIGGER;
* Aguarda resposta em forma do pulso ECHO;
* Enquanto o pulso ECHO estiver ativo, incrementa a medida detectada;
* Armazena o valor em registrador, amostrado na saída MEDIDA;
* Levanta o sinal PRONTO.
```

Esse mesmo comportamento pode ser verificado na arquitetura apresentada no Apêndice B.2 entre as linhas 18 e 46, em que é descrita a lógica de transições e as saídas de cada estado (máquina de Moore).

A princípio, os estados "inicial", "espera_echo"e "medida"não possuem sinais de controle associados. No caso do estado "inicial", tem-se uma distinção de seu seguinte ("preparacao") para que não seja descartada a medida enquanto não for feita outra requisição de medida ao sistema (sinal "zera"na linha 40 do Apêndice B.2). Os demais estados mencionados poderiam ser contraídos a um único, porém, em futuras adaptações do módulo, a diferenciação deles pode ser necessária.

2.1.2 Fluxo de dados

O fluxo de dados foi criado de forma a interligar o *contador_cm* mencionado no topo da seção atual; um gerador de pulso configurado para produzir de acordo com a especificação do HC-SR04; e um registrador para conter a medida detectada, contendo a entidade descrita no Apêndice A.3.

O *contador_cm* recebe os sinais comuns do sistema (*clock* e *reset*) e o pulso do sensor, como mencionado anteriormente. Sua saída é ligada à entrada do registrador, que a inscreve em memória no estado de "armazenamento". Ainda no registrador, há o sinal de *clear* que é levantado no estado de preparação ("preparacao") para medida.

Por último, a instância do componente "gerador_pulso"produz pulsos com largura de 10 μs a partir da entrada "gera"declarada e a contagem de 500 ciclos de *clock*.

2.2 Estratégia de Testes

O *testbench* do circuito de interface foi configurado para adotar os casos da Tabela 1. Para cada iteração rodada, a largura de pulso é definida de acordo com a segunda coluna da tabela.

A sequência de sinais e intervalos aplicados é como segue: espera-se pela próxima borda de descida do *clock* para sincronizar o início do teste; levanta o sinal "medir"(Apêndice A.1) por 5 ciclos; espera um tempo simulado de 400 μs (representativo do tempo real de medição e modulação do sinal); produz o *echo* com a devida largura para a interface; aguarda pela sinalização de fim da medida (sinal "pronto", Apêndice A.1); e adiciona um preenchimento em tempo (100 μs) antes de repetir esses passos em uma nova iteração com outro caso de teste.

Código do teste	Largura de pulso (μs)	Largura de pulso (cm)	Leitura esperada
1	1789	30.41	30
2	3575	60.78	61
3	5882	100	100
4	42	0.71	1

Tabela 1: Casos de teste de medição definidos no *testbench*.

Dos testes apresentados na Tabela 1, o último (4) serve apenas para análise teórica de funcionamento do dispositivo e não fará parte do teste prático da montagem experimental, tendo em vista a limitação do sensor ultrassônico adotado no projeto ($2 < D < 400[cm]$).

Com relação à metodologia de verificação prática dos testes propostos, serão realizados os seguintes passos:

1. *Reset* global do sistema pelo acionamento da chave SW0;
2. Posicionamento do objeto com auxílio de instrumento de medição (régua ou trena);
3. Acionamento do sinal de medida da chave SW1;
4. Registro na Tabela 3 do valor coletado do *display* e da largura de pulso observada na ferramenta *Scope*;
5. Repetir os passos 3 e 4 até que se tenha 10 registros de medidas.

Os passos 1-5 serão repetidos para cada caso de teste. Ao fim desse processo, o restante da Tabela 3 será preenchida com os valores calculados e médias observadas.

2.3 Simulação e Síntese do Design

Depois de definidos os casos de testes, conforme apresentados na Tabela 1, e descritos em *testbench*, o projeto foi integrado ao *software ModelSim* para fins de simulação. Desse modo, foi possível simular os cenários testados e acompanhar o comportamento de cada sinal de entrada, saída e depuração ao longo do funcionamento do circuito.

Para o primeiro cenário, apresentado na Figura 1, a ativação do sinal "medir"no início do caso faz com que o sensor dispare o Trigger de 10 μs (destacado em verde) e receba, no sentido de retorno, um pulso de *echo* com duração de 1789 μs. Conforme a relação descrita no início da Seção 2, essa largura representa, na realidade, uma distância de $\frac{1789}{58,82} = 30,41[cm]$.

O período de ativação do *echo* representa a permanência da Unidade de Controle no estado "medida", o que permite o incremento em uma unidade da contagem interna do contador BCD a cada *tick* gerado pelo *contador_m*, isto é, a cada 2941 μs do pulso, correspondente a 1 *cm* da medição. Quando a borda de descida do *echo* é detectada, o circuito passa ao estado de armazenamento, em que a resposta encontrada é guardada em um registrador para, depois, poder ser mostrada na saída hexadecimal "Medida", em roxo. No destaque, em vermelho, é possível verificar o acionamento do sinal "pronto"logo após a descida do *echo*, indicando o fim da medida proposta pelo caso 1 de teste.

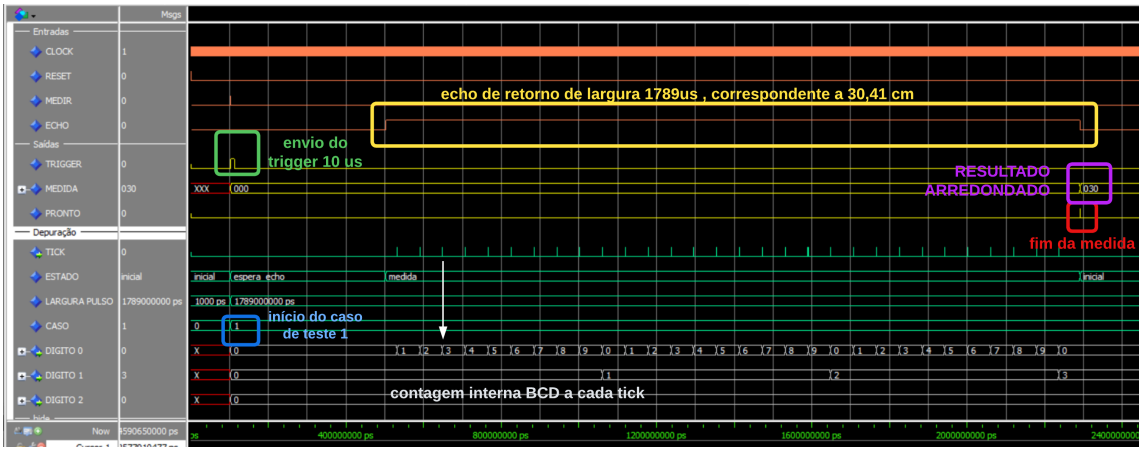


Figura 1: Formas de onda para o teste de cálculo da distância a 30,41cm.

As Figuras de 2 a 4 a seguir seguem a mesma lógica da primeira, representando sempre o disparo de *trigger* seguido do retorno *echo*, a contagem BCD - conforme os *ticks* - e o resultado da distância

em hexadecimal após o indicador de fim da medida. É relevante notar, em cada uma das imagens de simulação, o funcionamento correto do mecanismo de arredondamento da medida final para o inteiro mais próximo encontrado e a disponibilização do resultado em 3 dígitos decimais.

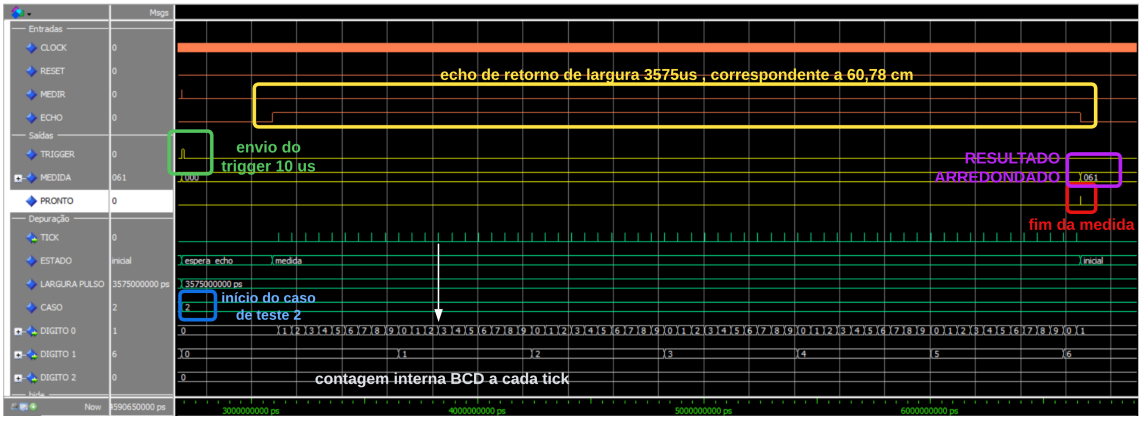


Figura 2: Formas de onda para o teste de cálculo da distância a 60,78cm.

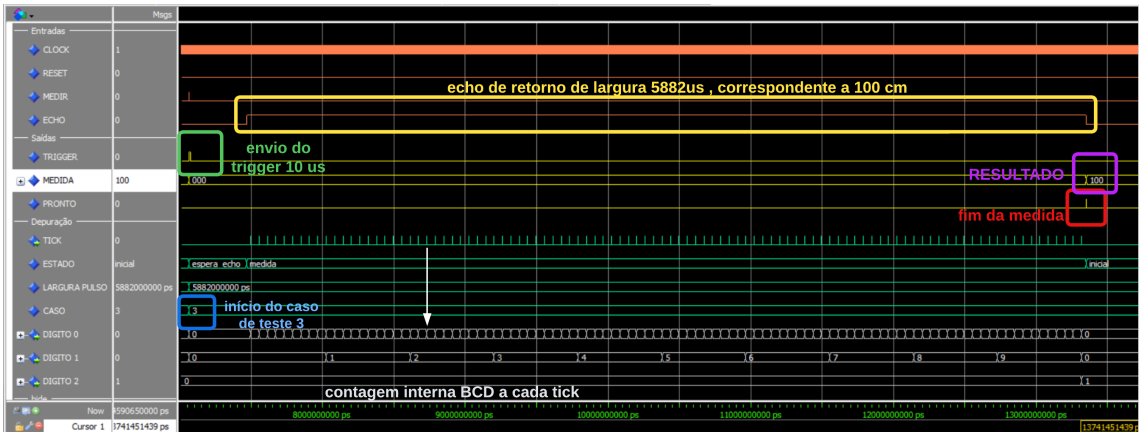


Figura 3: Formas de onda para o teste de cálculo da distância a 100cm.

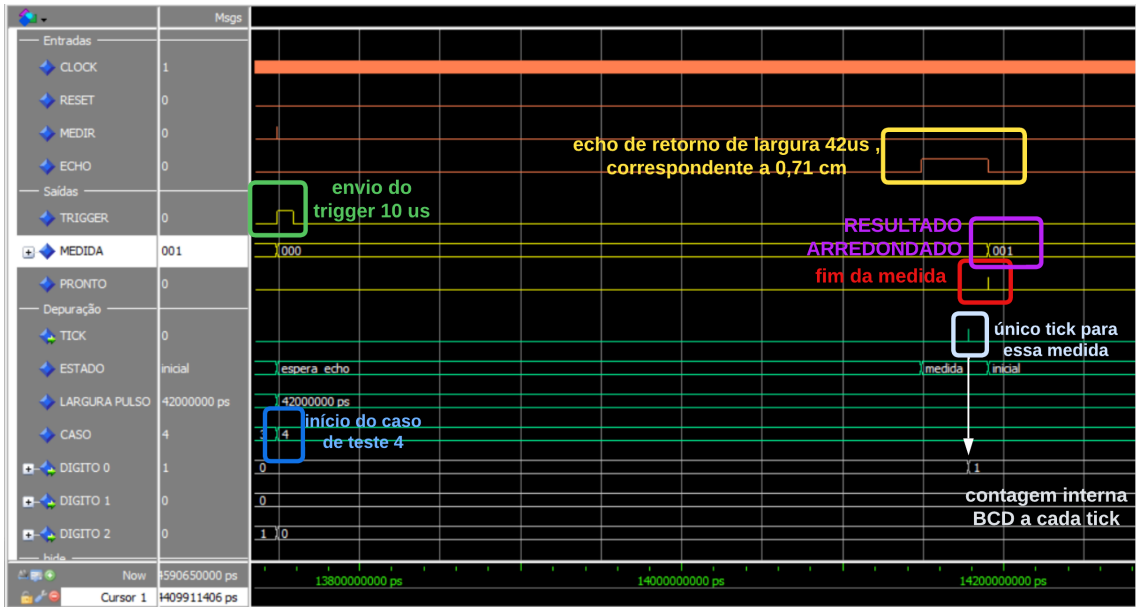


Figura 4: Formas de onda para o teste de cálculo da distância a 0,71cm.

Após realizar as simulações e confirmar, de fato, o bom funcionamento do circuito projetado para a interface com o sensor ultrassônico, os componentes do projeto foram importados para um novo arquivo na ferramenta *Intel Quartus Prime*, onde foi possível analisar cada componente e sintetizar o circuito final.

Partindo dessa sintetização, foram criadas visualizações por meio da ferramenta *RTL viewer*, através da qual foi possível verificar os diagramas de bloco do circuito e confirmar sua conformidade com as especificações definidas na Seção 2. Nesse sentido, foram ilustrados o circuito completo de interface (Figura 5, os componentes formadores do Fluxo de Dados (Figura 6) e o funcionamento interno do medidor/conversor de distância *contador_cm* (Figura 7).

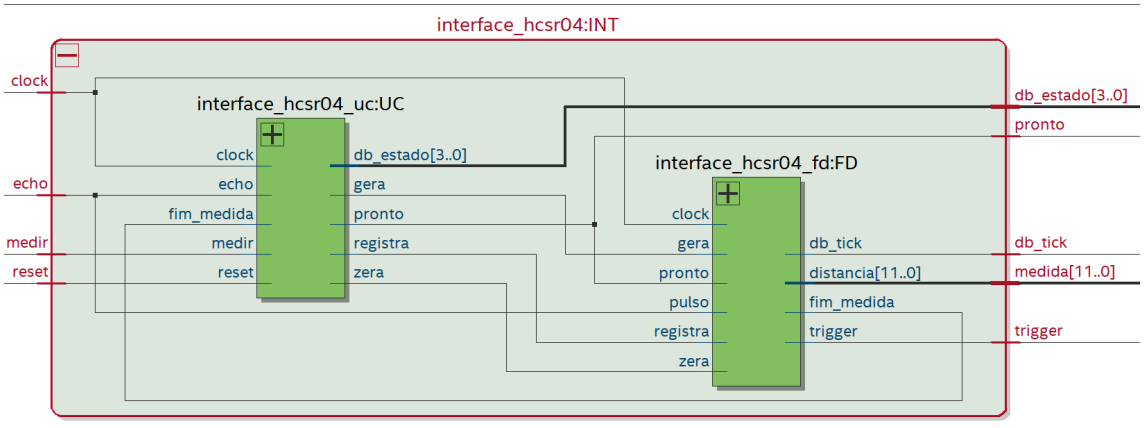


Figura 5: Diagrama RTL do circuito de interface com o sensor ultrassônico.

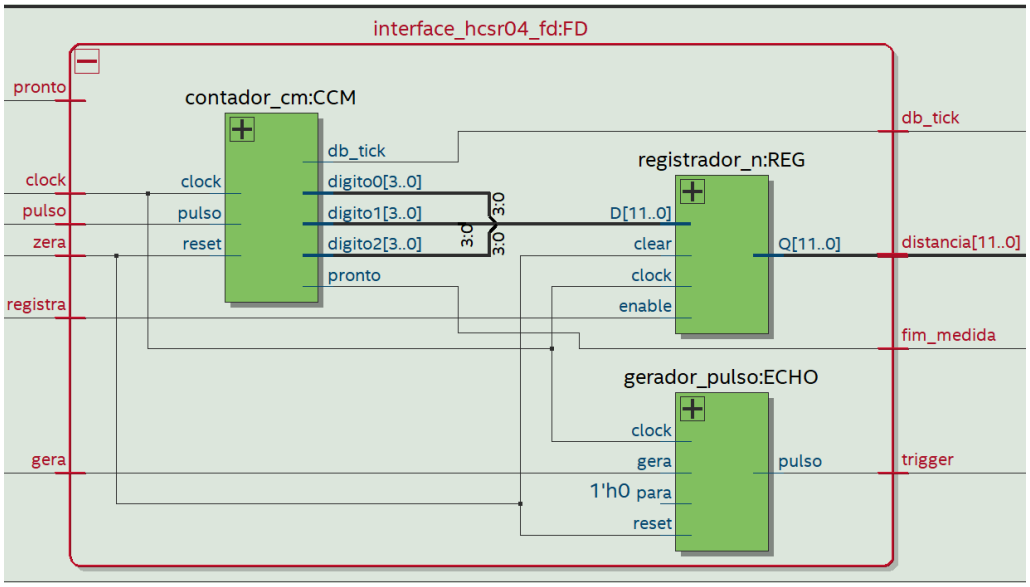


Figura 6: Diagrama RTL do Fluxo de Dados do circuito de interface com o sensor.

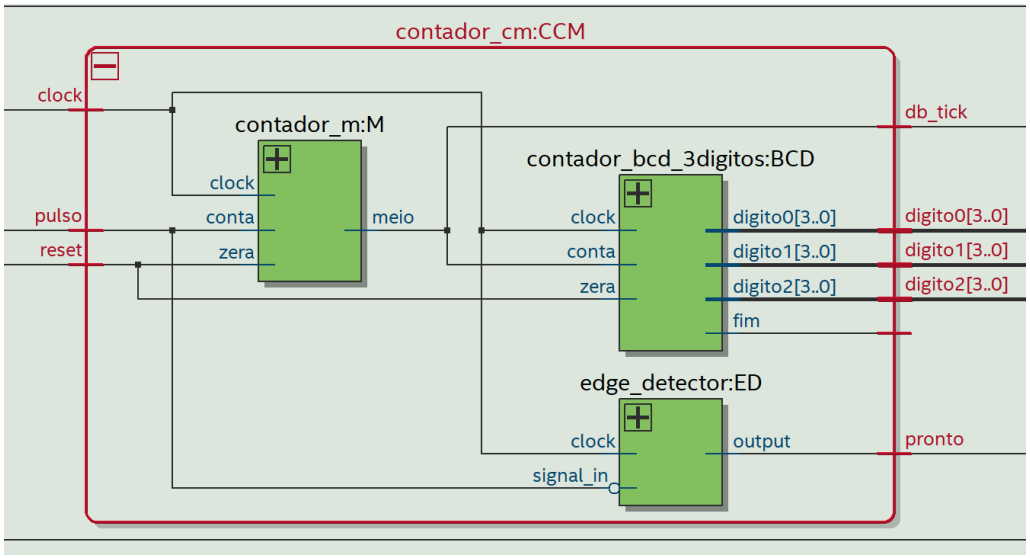
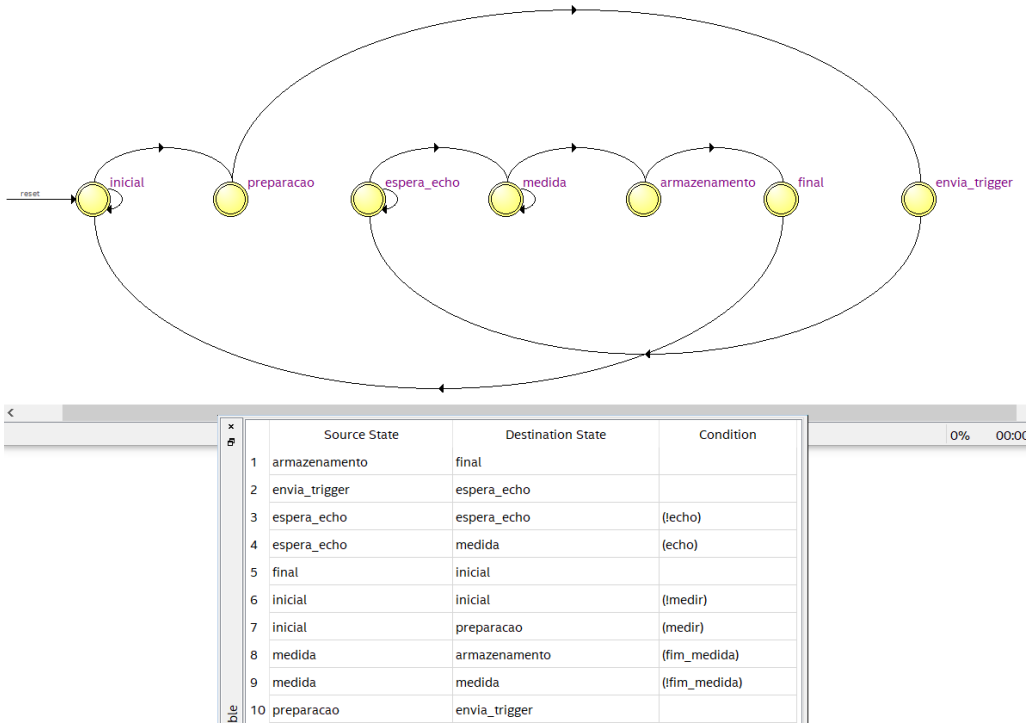


Figura 7: Diagrama RTL do funcionamento interno do *contador_cm*.

Após verificar os diagramas de blocos e a conformidade do Fluxo de Dados, a ferramenta interna *State Machine Viewer* foi utilizada para conferir a estrutura da máquina de estados que rege a Unidade de Controle do projeto, assim como as transições entre estados e os sinais de condição que influenciam no funcionamento. Esses aspectos estão representados na Figura 8.



	Source State	Destination State	Condition
1	armazenamento	final	
2	envia_trigger	espera_echo	
3	espera_echo	espera_echo	(!echo)
4	espera_echo	medida	(echo)
5	final	inicial	
6	inicial	inicial	(!medir)
7	inicial	preparacao	(medir)
8	medida	armazenamento	(fim_medida)
9	medida	medida	(!fim_medida)
10	preparacao	envia_trigger	

Figura 8: Máquina de estados da Unidade de Controle e tabela com seus sinais de transição

Por fim, o projeto da interface com o sensor ultrassônico HC-SR04 foi encapsulado em um novo módulo *exp3_sensor* de forma a permitir os testes de funcionamento práticos em laboratório. Nesse sentido, a entrada "medir"foi ligada a um novo componente (*edge_detector*) para possibilitar sua ativação por meio de um botão físico.

Na parte das saídas, foram adicionados quatro módulos do componente *hexa7seg*, que realiza a conversão de sinais binários de 4 bits para que sejam visualizados em *displays* de sete segmentos em forma de dígito hexadecimal. Além disso, foram mantidos e vinculados para as saídas outros sinais de depuração, tendo em mente a interação prática com o circuito. Esse novo módulo foi sintetizado no *Quartus*, por onde foi possível obter o diagrama RTL de blocos ilustrado na Figura 9.

Finalmente, foi configurada a pinagem conforme a Tabela 2 descrita na Seção 3 e o projeto foi exportado para um arquivo de extensão *qar* que poderá ser diretamente utilizado em laboratório para elaboração na placa FPGA.

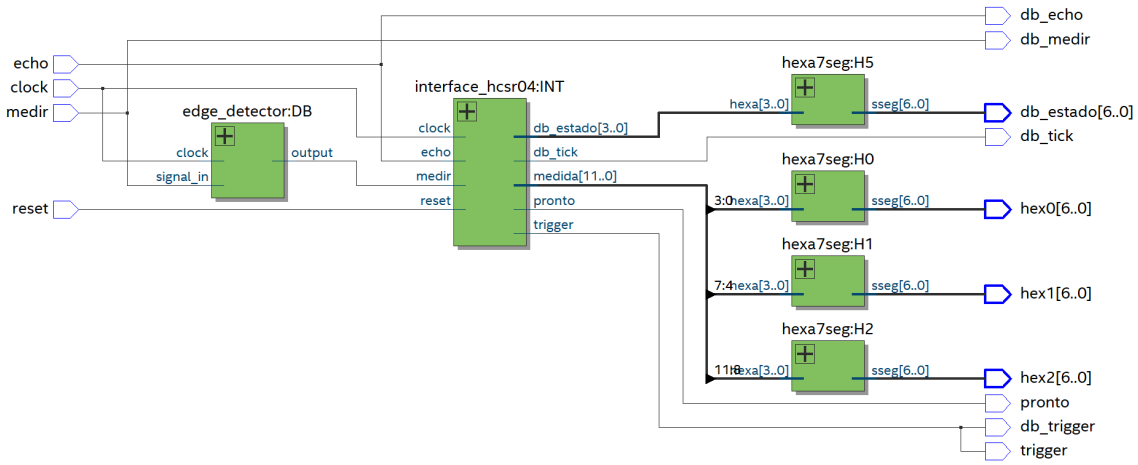


Figura 9: Diagrama RTL do funcionamento interno do circuito completo encapsulado.

3 PLANEJAMENTO DA AULA PRÁTICA

Em laboratório, será realizada a síntese do projeto arquivado (extensão *qar*) segundo a pinagem da Tabela 2.

Inicialmente, todos os componentes externos ao circuito projetado (Analog Discovery, módulo do sensor HC-SR04 e circuito conversor de nível de tensão) serão referenciados em um mesmo potencial da placa programada (GND).

Em seguida, os pinos de alimentação da FPGA serão ligados a esses componentes de acordo com o nível de tensão esperado: 3,3 V de referencial na placa conversora; e 5 V para o funcionamento do sensor ultrassônico.

Com isso, os sinais de interface do circuito podem ser conectados. A entrada *trigger* na ponte da FPGA pode ser ligada diretamente ao pino de mesmo nome no sensor, o *echo* do sensor deve ser ligado à entrada de um par entrada-saída do CI 7HC4050 e coletado em seu dual para o sinal *echo* efetivo do circuito na porta B12. Por fim, os sinais de depuração - *db_trigger* e *db_echo* - serão associados a canais do Analog Discovery (relações pino-canal: G12-1 e K16-2, respectivamente).

A partir desse momento, a placa seria ligada e programada com o projeto em questão para a realização dos testes propostos.

Os sinais ligados aos canais do Analog Discovery serão devidamente configurados para comparar as larguras de pulso esperadas com os resultados obtidos pelos *displays*.

Com auxílio da ferramenta *Scope* do WaveForms, os alcances dos canais serão postos como 1 V/div e o *trigger* do osciloscópio será testado na faixa de 1 V a 2 V, visto que a tensão nominal esperada é de 3,3 V.

Em uma primeira medida, as divisões de base serão mantidas em 0,01 ms ($i.e.$ 10 μs), de forma a assegurar a largura de pulso do *trigger*. Para as medições subsequentes se fará uso de 400 us/div para a análise por cursores da largura do pulso de *echo* e validar o dado apresentado nos *displays* pela Equação 1, em que L é a largura do pulso em μs e D o valor em cm apresentado.

Sinal	Ligação na placa FPGA	Pino na FPGA	Analog Discovery
clock	CLOCK_50	PIN_M9	-
reset	chave SW0	PIN_U13	-
medir	chave SW1	PIN_V13	-
echo	GPIO_1_D3	PIN_B12	-
trigger	GPIO_1_D1	PIN_A12	-
hex0	display HEX0	PIN_U21 PIN_V21 PIN_W22 PIN_W21 PIN_Y22 PIN_Y21 PIN_AA22	-
hex1	display HEX1	PIN_AA20 PIN_AB20 PIN_AA19 PIN_AA18 PIN_AB18 PIN_AA17 PIN_U22	-
hex2	display HEX2	PIN_Y19 PIN_AB17 PIN_AA10 PIN_Y14 PIN_V14 PIN_AB22 PIN_AB21	-
pronto	led LEDR[0]	PIN_AA2	-
db_medir	led LEDR[1]	PIN_AA1	-
db_trigger	GPIO_1_D33	PIN_G12	CH1+ (Scope)
db_echo	GPIO_1_D35	PIN_K16	CH2+ (Scope)
db_estado	display HEX5	PIN_N9 PIN_M8 PIN_T14 PIN_P14 PIN_C1 PIN_C2 PIN_W19	-
db_tick	led LEDR[2]	PIN_W2	-

Tabela 2: Pinagem para a montagem experimental.

4 ATIVIDADES EXPERIMENTAIS

Com a devida montagem do circuito e verificação da largura do pulso de *trigger* registrada pela ferramenta *Scope*, os testes propostos na Tabela 1 (exceto o caso de teste 4) serão amostrados 10 vezes e registrados na Tabela 3.

Os valores calculados vêm da relação entre tempo de pulso por *cm* medido (Equação 1) e serão comparados diretamente ao apresentado nos *displays*, junto à média verificada em cada caso, em *cm*.

Medida	Posição 1 (30.41cm)		Posição 2 (60.78cm)		Posição 3 (100cm)	
	calculada	medida	calculada	medida	calculada	medida
1						
2						
3						
4						
5						
6						
7						
8						
9						
10						
média						

Tabela 3: Comparação entre os valores experimentais calculados e os gerados.

APÊNDICES

A Declaração de entidades da interface

A.1 Entidade principal

```
1     entity interface_hcsr04 is
2         port (
3             clock      : in std_logic;
4             reset      : in std_logic;
5             medir      : in std_logic;
6             echo       : in std_logic;
7             trigger    : out std_logic;
8             medida     : out std_logic_vector(11 downto 0);
9             pronto     : out std_logic;
10            db_estado  : out std_logic_vector(3 downto 0);
11            db_tick    : out std_logic
12        );
13    end entity interface_hcsr04;
```

A.2 Unidade de controle

```
1     entity interface_hcsr04_uc is
2         port (
3             clock       : in  std_logic;
4             reset       : in  std_logic;
5             medir       : in  std_logic;
6             echo        : in  std_logic;
7             fim_medida  : in  std_logic;
8             zera        : out std_logic;
9             gera        : out std_logic;
10            registra    : out std_logic;
11            pronto      : out std_logic;
12            db_estado   : out std_logic_vector(3 downto 0)
13        );
14    end interface_hcsr04_uc;
```

A.3 Fluxo de dados

```
1     entity interface_hcsr04_fd is
2         port (
3             clock       : in std_logic;
4             gera        : in std_logic;
5             pulso       : in std_logic;
6             registra    : in std_logic;
7             zera        : in std_logic;
8             pronto      : in std_logic;
9             trigger     : out std_logic;
10            fim_medida  : out std_logic;
```

```
11              distancia   : out std_logic_vector(11 downto 0);
12              db_tick     : out std_logic
13          );
14      end entity interface_hcsr04_fd;
```

A.4 Contador de distância

```
1       entity contador_cm is
2           generic (
3               constant R : integer := 50;
4               constant N : integer := 6
5           );
6           port (
7               clock    : in  std_logic;
8               reset    : in  std_logic;
9               pulso    : in  std_logic;
10              digito0  : out std_logic_vector(3 downto 0);
11              digito1  : out std_logic_vector(3 downto 0);
12              digito2  : out std_logic_vector(3 downto 0);
13              fim      : out std_logic;
14              pronto   : out std_logic;
15              db_tick  : out std_logic
16          );
17      end entity contador_cm;
```

B Implementação de arquiteturas

B.1 Contador de distância

```
1   architecture estrutural of contador_cm is
2
3     component contador_m
4         generic (
5             constant M : integer;
6             constant N : integer
7         );
8         port (
9             clock : in  std_logic;
10            zera  : in  std_logic;
11            conta : in  std_logic;
12            Q     : out std_logic_vector (N-1 downto 0);
13            fim   : out std_logic;
14            meio  : out std_logic
15        );
16    end component;
17
18    component contador_bcd_3digitos
19        port (
20            clock   : in  std_logic;
21            zera    : in  std_logic;
22            conta   : in  std_logic;
23            digito0 : out std_logic_vector(3 downto 0);
24            digito1 : out std_logic_vector(3 downto 0);
25            digito2 : out std_logic_vector(3 downto 0);
26            fim     : out std_logic
27        );
28    end component;
29
30    component edge_detector
31        port (
32            clock     : in  std_logic;
33            signal_in : in  std_logic;
34            output    : out std_logic
35        );
36    end component;
37
38    signal s_pulso_negado, s_tick : std_logic;
39
40    -- saidas
41    signal s_fim_pulso : std_logic;
42
43  begin
44
45    s_pulso_negado <= not pulso;
46
47    BCD: contador_bcd_3digitos
48        port map (
```

```
49              clock   => clock,
50              zera    => reset,
51              conta   => s_tick,
52              digito0 => digito0,
53              digito1 => digito1,
54              digito2 => digito2,
55              fim     => fim
56          );
57
58      -- ticks a cada 1 cm
59      M: contador_m
60          generic map (
61              M => 2941,
62              N => 12
63          )
64          port map (
65              clock => clock,
66              zera  => reset,
67              conta => pulso,
68              Q     => open,
69              fim   => open,
70              meio  => s_tick
71          );
72
73      ED: edge_detector
74          port map (
75              clock      => clock,
76              signal_in  => s_pulso_negado,
77              output     => s_fim_pulso
78          );
79
80      pronto  <= s_fim_pulso;
81      db_tick <= s_tick;
82
83  end architecture estrutural;
```

B.2 Unidade de controle

```
1   architecture fsm_arch of interface_hcsr04_uc is
2     type tipo_estado is (inicial, preparacao, envia_trigger,
3                          espera_echo, medida, armazenamento, final);
4     signal Eatual, Eprox: tipo_estado;
5   begin
6
7     -- estado
8     process (reset, clock)
9     begin
10        if reset = '1' then
11            Eatual <= inicial;
12        elsif clock'event and clock = '1' then
13            Eatual <= Eprox;
```

```
14          end if;
15      end process;
16
17      -- logica de proximo estado
18      process (medir, echo, fim_medida, Eatual)
19      begin
20        case Eatual is
21          when inicial =>          if medir='1' then Eprox <= preparacao;
22                                   else              Eprox <= inicial;
23                                   end if;
24          when preparacao =>       Eprox <= envia_trigger;
25          when envia_trigger =>    Eprox <= espera_echo;
26          when espera_echo =>      if echo='0' then Eprox <= espera_echo;
27                                   else             Eprox <= medida;
28                                   end if;
29          when medida =>           if fim_medida='1' then Eprox <=
                armazenamento;
30                                   else                   Eprox <= medida;
31                                   end if;
32          when armazenamento =>    Eprox <= final;
33          when final =>            Eprox <= inicial;
34          when others =>           Eprox <= inicial;
35        end case;
36      end process;
37
38    -- saidas de controle
39    with Eatual select
40        zera <= '1' when preparacao, '0' when others;
41    with Eatual select
42        gera <= '1' when envia_trigger, '0' when others;
43    with Eatual select
44        registra <= '1' when armazenamento, '0' when others;
45    with Eatual select
46        pronto <= '1' when final, '0' when others;
47
48    with Eatual select
49        db_estado <= "0000" when inicial,
50                     "0001" when preparacao,
51                     "0010" when envia_trigger,
52                     "0011" when espera_echo,
53                     "0100" when medida,
54                     "0101" when armazenamento,
55                     "1111" when final,
56                     "1110" when others;
57
58    end architecture fsm_arch;
```