Módulo Acelerômetro Digital na Placa DE10-Lite

Autor: **Rhaycen Rodrigues Prates**

Orientador: **Prof. Dr. Valter Fernandes Avelino**

Centro Universitário FEI

Departamento: Engenharia Elétrica

Abril - 2019

Revisão 1: Prof. Dr. Valter Fernandes Avelino (Janeiro/2021)

Índice

1 Objetivo

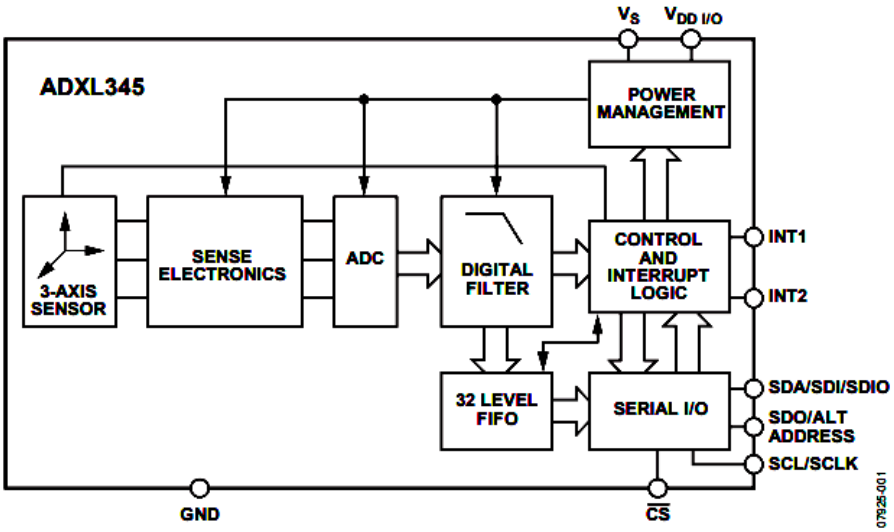
Este documento tem como objetivo mostrar os princípios de funcionamento, as aplicações e a descrição em VHDL do acelerômetro ADXL345 presente na plataforma de desenvolvimento DE10-Lite utilizada nos Laboratórios de Sistemas Digitais I e II do Centro Universitário FEI. Além disso, este documento visa esclarecer como implementar o módulo do acelerômetro em projetos de sistemas digitais de forma a explorar um dos recursos fornecidos pela plataforma DE10-Lite que permite a integração de componentes eletrônicos que auxiliam e inovam os projetos através da metodologia System-on-Chip (SoC), onde os recursos necessários estão presentes na mesma plataforma (AVELINO, 2019), (TERASIC, 2018).

2 Introdução

O acelerômetro digital é um dispositivo capaz de medir a aceleração dinâmica e estática de seus eixos e convertê-la em um valor digital. Com isso possui diversas aplicações que vão desde a instrumentação médica à proteção de disco rígido (HDD), além de aplicações em dispositivos de direção e navegação pessoal e em jogos (ANALOG DEVICES, 2015).

A plataforma de desenvolvimento DE10-Lite contém o acelerômetro digital ADXL345, cujo diagrama de blocos funcional é mostrado na Figura 1. Este acelerômetro é capaz de realizar medições com resolução de ±2.g até ±16.g e fornecer uma saída de dados digitais de 16 bits, em complemento de dois, que podem ser acessados através de uma interface digital SPI (3 ou 4 fios) ou I2C. Quando configurado no modo de alta resolução (ou quando estiver com fundo de escala de ±2.g) é possível realizar medições de variações de inclinação com resolução menor que 1.0º (3,9.mg/LSB) e dessa forma o ADXL345 pode atuar como um sensor de inclinação devido a sua propriedade de medir acelerações estáticas, além de poder ser utilizado em medidas de aceleração dinâmicas (resultado de movimento ou choque) (ANALOG DEVICES, 2015), (TERASIC, 2018).

Figura 1: Diagrama de blocos funcional do ADXL345.



Fonte: ANALOG DEVICES, 2015.

O ADXL345 pode ser utilizado em diversas outras aplicações de sensoriamento como: sensor de atividade e inatividade (onde se detecta a presença de movimento pela comparação da aceleração de seus eixos com um valor limite definido pelo usuário), sensor de toque (pela vibração do sensor pode-se detectar um ou dois toques em qualquer direção), sensor de queda livre (que pode indicar se o dispositivo está caindo), entre outros (ANALOG DEVICES, 2015).

No presente trabalho de iniciação didática são apresentados dois exemplos de aplicação: um sensor de inclinação e um medidor de aceleração da gravidade, bem como o processo de projeto (em VHDL) para a utilização desse dispositivo como um componente do sistema digital.

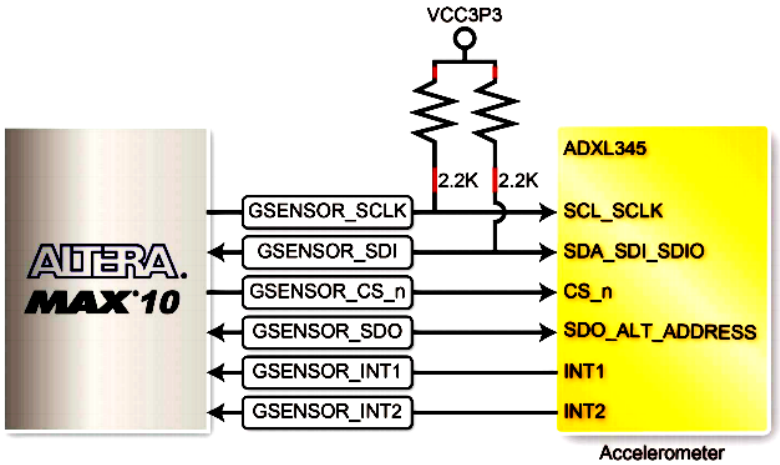
3 Princípio de funcionamento

O sensor é uma estrutura micro-usinada de polissilício construído em cima de um substrato (*wafer*) de silício. A estrutura é suspensa sobre a superfície do *wafer* através de molas de polissilício que fornecem uma resistência contra forças devido à aceleração aplicada. A deflexão da estrutura é medida usando capacitores diferenciais que consistem em placas fixas independentes e outras ligadas à massa em movimento. A aceleração movimenta a massa de prova e desequilíbria o capacitor diferencial, resultando em uma saída do sensor cuja amplitude é proporcional à aceleração. Uma demodulação sensível à fase é usada para determinar a magnitude e a polaridade da aceleração (ANALOG DEVICES, 2015).

A interface entre o acelerômetro ADXL345 e o FPGA MAX 10 da plataforma de desenvolvimento DE10-Lite é feita através de 6 sinais (Figura 2), cujas funções são descritas a seguir: (ANALOG DEVICES, 2015), (TERASIC, 2018).

- **GSENSOR_SCLK:** Sinal de relógio (*SCLK*) utilizado pelo sensor, fornecido pelo FPGA;
- **GSENSOR_SDI:** Entrada de dados seriais do acelerômetro e saída de dados seriais do FPGA. Utilizada para configuração das características necessárias do acelerômetro pelo usuário;
- **GSENSOR_CS_N:** Sinal de habilitação (*Chip_Select*) da troca de dados entre o ADXL345 e o MAX 10. É ativo em estado lógico zero;
- **GSENSOR_SDO:** Saída de dados seriais do acelerômetro e entrada de dados seriais do FPGA. Os dados de aceleração medidos são enviados por esta ligação;
- **GSENSOR_INT1:** Sinal de interrupção para o FPGA (configurável, indica quando um evento ocorreu);
- **GSENSOR_INT2:** Sinal de interrupção para o FPGA (configurável, indica quando um evento ocorreu);

Figura 2: Sinais de interface entre o FPGA MAX 10 e o acelerômetro digital ADXL345.



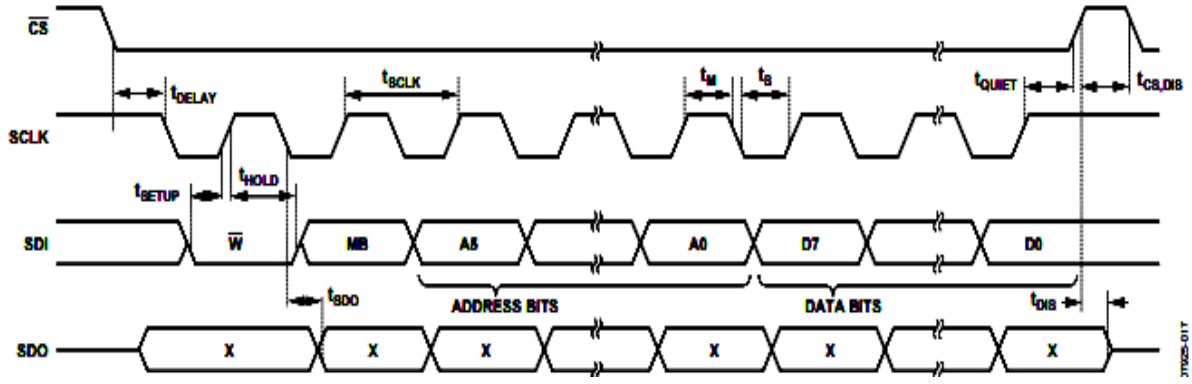
Fonte: TERASIC, 2018.

3.1 Comunicação serial

A transmissão de dados entre o FPGA e o ADXL345 é feita de maneira serial, ou seja, é transmitido um bit a cada ciclo de clock. Para o módulo descrito em VHDL foi utilizado a comunicação serial com 4 fios (SPI 4-wire), onde a saída e entrada de dados são realizadas por duas linhas de comunicação diferentes. Para a configuração do modo de comunicação SPI 4-wire é necessário zerar o bit D6 no registrador DATA_FORMAT (endereço 49) do acelerômetro (ANALOG DEVICES, 2015).

Para realizar a transmissão, ou seja, realizar a escrita nos registros do acelerômetro no modo SPI 4-wire é necessário seguir um protocolo de comunicação. Primeiramente deve-se ativar bit $\overline{CS}$ (*Chip-Select*), em nível lógico zero, para o início da transmissão, em seguida é necessário habilitar o clock de comunicação do acelerômetro que deve ser enviado pelo FPGA através do sinal SCLK, permitindo que a transmissão possa ser realizada. O sinal SDI é responsável pelo envio dos bits de escrita do FPGA para o ADXL345, permitindo a configuração do modo de operação do acelerômetro. O primeiro bit a ser enviado é o $\overline{W}$/R, que determina se está sendo realizada a escrita ou a leitura de dados do sensor, sendo '1' para leitura e '0' para escrita. O segundo bit é o MB (*Multiple-Bit*) que quando está em nível lógico alto realiza o endereçamento do próximo registrador automaticamente depois de cada 8 ciclos de clock. Em seguida estão os bits A5 à A0 que são responsáveis pelo endereçamento do registrador a ser configurado na transmissão. Por último, o byte de dados D7 à D0 que são os bits que configuram o registrador endereçado anteriormente. Para encerrar a transmissão os bits $\overline{CS}$ e SCLK devem retornar ao nível lógico '1'. A forma de onda necessária para realizar a transmissão de dados do FPGA para o ADXL345 é mostrada a seguir na Figura 3 (ANALOG DEVICES, 2015).

Figura 3: Transmissão dos dados do FPGA para o ADXL345 (Write).

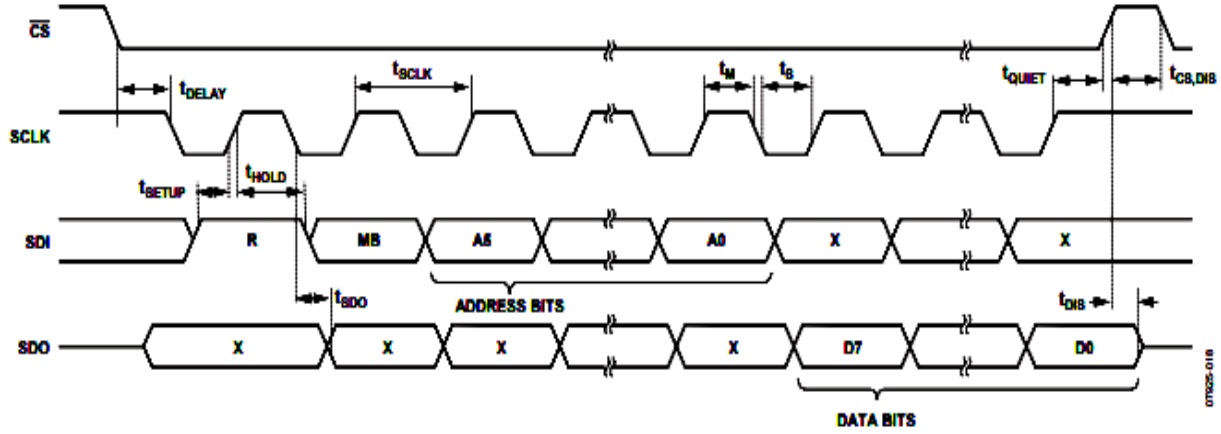


Fonte: ANALOG DEVICES, 2015.

Para realizar a recepção, ou seja, realizar a leitura dos dados de aceleração do acelerômetro no modo SPI 4-wire, é necessário seguir o outro protocolo de comunicação. Novamente deve-se deixar o bit $\overline{CS}$ em nível lógico zero para o início da transmissão, em seguida é necessário habilitar o clock do acelerômetro que deve ser enviado pelo FPGA através do sinal SCLK e com isso a transmissão pode ser realizada. O primeiro bit do sinal SDI a ser

enviado é o $\overline{W}$/R, que deve estar em nível lógico ‘1’ para leitura. O segundo bit é o MB que, quando está em nível lógico alto, realiza o endereçamento do próximo registrador automaticamente depois de cada 8 ciclos de clock. Em seguida estão os bits A5 à A0 que são responsáveis pelo endereçamento do registrador responsável pelo envio dos dados de aceleração. Ao realizar esses passos, o sensor envia através da ligação SDO os bits D7 à D0 (um de cada vez) que correspondem aos dados de um byte com o valor da aceleração do registrador endereçado. Para encerrar a transmissão os bits $\overline{CS}$ e SCLK devem retornar ao nível lógico ‘1’. A forma de onda necessária para realizar a leitura de dados é mostrada a seguir na Figura 4 (ANALOG DEVICES, 2015).

Figura 4: Leitura dos dados do ADXL345 (Read).



Fonte: ANALOG DEVICES, 2015.

Antes de realizar a escrita e leitura dos dados do acelerômetro é necessário definir os parâmetros de aceleração desejados. Os principais são: a faixa de operação (valores de ±2.g até ±16.g), a velocidade de saída dos dados (até 3200 Hz) e o tipo de comunicação (I^2C ou SPI). A seguir é mostrado quais registradores são responsáveis por estas e outras funções. (ANALOG DEVICES, 2015).

3.2 Configuração dos registradores

O acelerômetro digital ADXL345 possui 58 registradores com diversas funções e que devem ser configurados segundo as características adequadas ao projeto em questão. A Tabela 1, extraída do datasheet do componente, apresenta os 58 registradores com seus endereços, características e funções (ANALOG DEVICES, 2015).

Tabela 1: Tabela de registradores do ADXL345.

Address					
Hex	**Dec**	**Name**	**Type**	**Reset Value**	**Description**
0x00	0	DEVID	R	11100101	Device ID
0x01 to 0x1C	1 to 28	Reserved			Reserved; do not access
0x1D	29	THRESH_TAP	R/$\overline{W}$	00000000	Tap threshold
0x1E	30	OFSX	R/$\overline{W}$	00000000	X-axis offset
0x1F	31	OFSY	R/$\overline{W}$	00000000	Y-axis offset
0x20	32	OFSZ	R/$\overline{W}$	00000000	Z-axis offset
0x21	33	DUR	R/$\overline{W}$	00000000	Tap duration
0x22	34	Latent	R/$\overline{W}$	00000000	Tap latency
0x23	35	Window	R/$\overline{W}$	00000000	Tap window
0x24	36	THRESH_ACT	R/$\overline{W}$	00000000	Activity threshold
0x25	37	THRESH_INACT	R/$\overline{W}$	00000000	Inactivity threshold
0x26	38	TIME_INACT	R/$\overline{W}$	00000000	Inactivity time
0x27	39	ACT_INACT_CTL	R/$\overline{W}$	00000000	Axis enable control for activity and inactivity detection
0x28	40	THRESH_FF	R/$\overline{W}$	00000000	Free-fall threshold
0x29	41	TIME_FF	R/$\overline{W}$	00000000	Free-fall time
0x2A	42	TAP_AXES	R/$\overline{W}$	00000000	Axis control for single tap/double tap
0x2B	43	ACT_TAP_STATUS	R	00000000	Source of single tap/double tap
0x2C	44	BW_RATE	R/$\overline{W}$	00001010	Data rate and power mode control
0x2D	45	POWER_CTL	R/$\overline{W}$	00000000	Power-saving features control
0x2E	46	INT_ENABLE	R/$\overline{W}$	00000000	Interrupt enable control
0x2F	47	INT_MAP	R/$\overline{W}$	00000000	Interrupt mapping control
0x30	48	INT_SOURCE	R	00000010	Source of interrupts
0x31	49	DATA_FORMAT	R/$\overline{W}$	00000000	Data format control
0x32	50	DATAX0	R	00000000	X-Axis Data 0
0x33	51	DATAX1	R	00000000	X-Axis Data 1
0x34	52	DATAY0	R	00000000	Y-Axis Data 0
0x35	53	DATAY1	R	00000000	Y-Axis Data 1
0x36	54	DATAZ0	R	00000000	Z-Axis Data 0
0x37	55	DATAZ1	R	00000000	Z-Axis Data 1
0x38	56	FIFO_CTL	R/$\overline{W}$	00000000	FIFO control
0x39	57	FIFO_STATUS	R	00000000	FIFO status

Fonte: ANALOG DEVICES, 2015.

A configuração dos registradores (Tabela 1) definem os parâmetros para as funções a serem executadas pelo acelerômetro. Os principais registradores que podem ser configurados são:

- **Registrador 29:** Define limiar para detecção de toque;
- **Registradores 30 a 32:** Definem o offset do valor de aceleração para os eixos X, Y, Z;
- **Registradores 33 a 35:** Definem a duração, latência de espera para a função de detecção de um toque e a janela de tempo para detecção de dois toques;
- **Registradores 36 a 39:** Definem características para detecção de atividade/inatividade;

- **Registradores 40 e 41:** Definem limiar e tempo para detecção de queda livre;
- **Registradores 42 a 43:** Definem detecção de um ou dois toques e seus eixos de medida;
- **Registrador 44:** Define a taxa de transmissão e o modo de operação (Power/Low_Power);
- **Registrador 45:** Define funções de ativação/desativação para economia de energia;
- **Registradores 46 a 48:** Definem os modos de operação das interrupções;
- **Registrador 49:** Define a formatação dos dados de comunicação serial, a resolução e a faixa de operação do acelerômetro;
- **Registradores 50 a 55:** Fornecem as medidas da aceleração nos três eixos (dois Bytes por eixo);
- **Registradores 56 e 57**: Definem o modo de operação e a sinalização da memória FIFO (*First In, First Out*) interna ao acelerômetro.

Para o **Módulo Acelerômetro** (detalhado na seção 4), foram utilizados os registradores 44 e 49 a 55, pois satisfazem os objetivos básicos do mesmo de extrair os valores de aceleração dos eixos X, Y e Z do sensor (ANALOG DEVICES, 2015).

O registrador 44 é denominado BW_RATE e através dele é possível configurar a velocidade da saída de dados e ativar o modo de baixa energia, conforme a Tabela 2. Os 3 bits mais significativos devem estar em nível lógico baixo pois estes não têm função. O bit D4 é o responsável pela ativação do modo de baixa energia e é ativado em nível lógico ‘1’. A operação do dispositivo em baixo consumo (LOW_POWER) implica em limitação da sua taxa de amostragem em no máximo 400 Hz e aumenta o ruído captado pelo sensor. Como na nossa aplicação não temos a preocupação de redução de consumo de energia, o módulo deve operar em modo de operação normal, portanto o bit D4 deve estar em nível ‘0’. Os bits D3 a D0 configuram a largura de banda (*bandwidth*) e a taxa de saída de dados (*output data rate*) que deve ser combinada com o valor de frequência do clock fornecido ao sensor (SCLK). Para a máxima taxa de saída de dados de 3.200 Hz (que implica em uma largura de banda de 1.600 Hz) é recomendado uma frequência de comunicação serial (SCLK) maior ou igual a 2 MHz. Para taxas de saída de dados menores (abaixo de 1.600 Hz) é necessária uma frequência de comunicação serial (SCLK) pelo menos 500 vezes maior que taxa de saída de dados. Para o **módulo acelerômetro** implementado neste trabalho foi utilizada uma taxa de saída de dados de 100 Hz e foi utilizada uma frequência de comunicação serial (SCLK) de 100 kHz (1000 vezes maior).

A Tabela 3 mostra a relação dos bits D3 a D0 e suas respectivas frequências de saída de dados. Portanto, para o módulo desenvolvido o código do Bit_Rate utilizado nos bits D3 a D0 foi “1010”, correspondendo a taxa de transferência de dados de 100 Hz (largura de banda de 50 Hz) (ANALOG DEVICES, 2015).

Tabela 2: Byte para configuração do registrador BW_RATE (44).

D7	D6	D5	D4	D3	D2	D1	D0
0	0	0	LOW_POWER	Rate			

Fonte: ANALOG DEVICES, 2015.

Tabela 3: Tabela de taxa de dados de saída e seus respectivos códigos binários.

Output Data Rate (Hz)	Bandwidth (Hz)	Rate Code
3200	1600	1111
1600	800	1110
800	400	1101
400	200	1100
200	100	1011
100	50	1010
50	25	1001
25	12.5	1000
12.5	6.25	0111
6.25	3.13	0110
3.13	1.56	0101
1.56	0.78	0100
0.78	0.39	0011
0.39	0.20	0010
0.20	0.10	0001
0.10	0.05	0000

Fonte: ANALOG DEVICES, 2015.

O registrador 49 é denominado DATA_FORMAT e através dele é possível configurar o formato no qual os dados de aceleração contidos nos registradores 50 a 55 serão apresentados, segundo a Tabela 4. Para o módulo foram configurados apenas os bits D6, D1 e D0, os bits D7, D5, D4, D3 e D2 apresentam funções que fogem do escopo do projeto, assim foram mantidos em nível lógico ‘0’. O bit D6 é o seletor do tipo de comunicação serial, SPI 3-wire ou 4-wire, sendo o nível ‘0’ correspondente ao SPI 4-wire e o nível ‘1’ ao SPI 3-wire. Como o módulo utiliza a comunicação serial de 4 fios (SPI 4 -wire), o bit D6 deve estar em nível ‘0’. Os bits D1 e D0 são responsáveis pela configuração do range, ou seja, da faixa de valores de aceleração que se deseja medir, cujos valores são de ±2.g, ±4.g, ±8.g e ±16.g, conforme a Tabela 5 (ANALOG DEVICES, 2015).

Tabela 4: Byte de configuração do registrador DATA_FORMAT (49).

D7	D6	D5	D4	D3	D2	D1	D0
SELF_TEST	SPI	INT_INVERT	0	FULL_RES	Justify	Range	

Fonte: ANALOG DEVICES, 2015.

Tabela 5: Relação entre os bits D1 e D0 e o valor da faixa de aceleração em cada eixo.

Setting		
D1	D0	g Range
0	0	±2 *g*
0	1	±4 *g*
1	0	±8 *g*
1	1	±16 *g*

Fonte: ANALOG DEVICES, 2015.

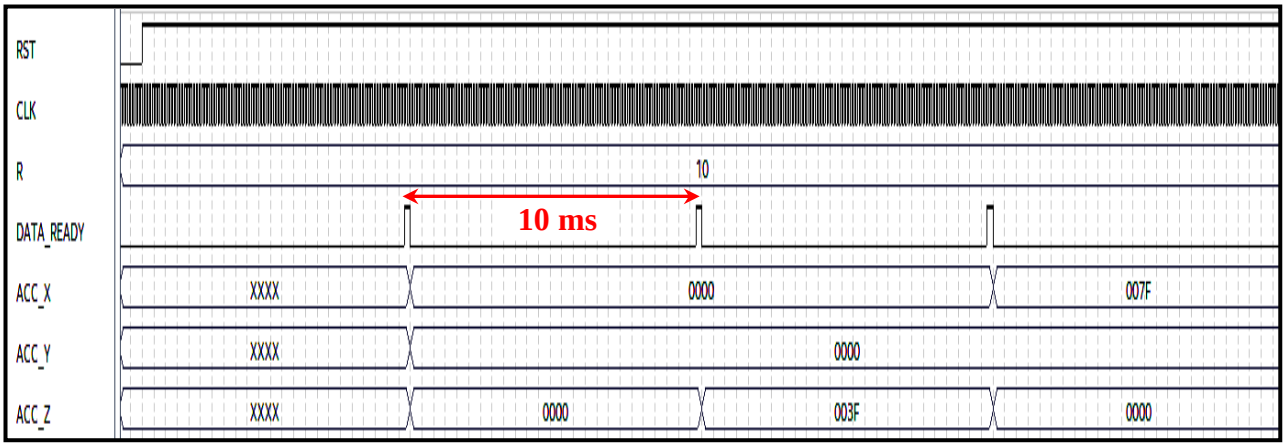
Os registradores 50 a 55 são denominados DATAX0, DATAX1, DATAY0, DATAY1, DATAZ0 e DATAZ1 respectivamente e contém os valores de aceleração medidos pelo ADXL345 para cada eixo. Os dados relativos ao eixo x são mostrados nos registradores 50 e 51, sendo os 8 bits mais significativos os do registrador 51 (DATAX1) e os 8 menos significativos os do registrador 50 (DATAX0), totalizando 16 bits de aceleração correspondente ao eixo x. Os dados relativos ao eixo y são mostrados nos registradores 52 e 53, sendo os 8 bits mais significativos os do registrador 53 (DATAY1) e os 8 menos significativos os do registrador 52 (DATAY0), totalizando 16 bits de aceleração correspondente ao eixo y. Os dados relativos ao eixo z são mostrados nos registradores 54 e 55, sendo os 8 bits mais significativos os do registrador 55 (DATAZ1) e os 8 menos significativos os do registrador 54 (DATAZ0), totalizando 16 bits de aceleração correspondente ao eixo z. Esses registradores são utilizados apenas para leitura dos dados. Os dados contidos nesses registradores são transmitidos do ADXL345 para o FPGA MAX10 através da ligação SDO como mostrado na Figura 4 (ANALOG DEVICES, 2015).

4 Geração do código VHDL

O módulo em VHDL do acelerômetro tem como função fornecer os dados de aceleração dos eixos x, y e z do sensor. O usuário deve ser capaz de configurar o valor do range de aceleração e receber a informação da aceleração nos três eixos do módulo sensor digital ADXL345.

O módulo acelerômetro opera no modo de aquisição contínua de dados ("*free running*") uma vez que tenha sido ativado o reset do módulo. Na Figura 5 é apresentada a temporização básica para configuração de leitura do módulo. Para inicializar o módulo (com o valor da faixa de aceleração em cada eixo, definido pelo vetor R[1..0]) o sinal RST deve ser ativado (em nível lógico zero) e a seguir deve permanecer desativado. A comunicação SPI com o sensor digital ADXL345 executa a configuração do sensor (escrita nos registros 44 e 49) e a leitura dos registros de aceleração (leitura dos registros 50 a 55). A ativação do sinal DATA_READY (DATA_READY='1') por um período do clock de comunicação (10 µs) informa que um conjunto completo de leitura foi realizado (nos três eixos) e que os dados de aceleração podem ser lidos nos registros de saída (ACC_X[15..0], ACC_Y[15..0] e ACC_Z[15..0]). O processo de leitura de dados do sensor é repetido a cada 10 ms (logo, o sinal DATA_READY tem frequência de 100 HZ). Dessa forma, uma vez inicializado, o módulo acelerômetro fornece uma nova leitura de aceleração nos três eixos a cada 10 ms (essa leitura de dados pode ser sincronizada a partir do sinal DATA_READY='1').

Figura 5: Temporização básica do **Módulo Acelerômetro**.



Fonte: Autor.

Deve ser observado que se for necessária a mudança do valor da faixa de aceleração (definido pelo vetor R[1..0]) o sinal RST deve ser novamente ativado para reiniciar os registros de controle do sensor digital ADXL345.

Observa-se também que, a alteração dos sinais de saída ocorre a cada a ativação do sinal DATA_READY, independentemente do valor dos registros de saída terem sido lidos ou não pelo usuário do módulo acelerômetro.

4.1 Sinais de entrada e saída do módulo acelerômetro

O módulo acelerômetro possui os seguintes sinais de entrada:

- **CLK**: Entrada de clock, cuja frequência deve ser de 50 MHz;

- **RST**: Sinal de entrada que habilita o início da troca de dados entre o FPGA e o ADXL345 (configuração dos registradores de controle do sensor digital ADXL345);

- **R[1..0]**: Entrada de dados que seleciona a faixa (range) de aceleração que o sensor deve operar, cuja combinação deve seguir a Tabela 5;

- **SDO**: Sinal de entrada de dados que deve ser conectado ao canal de saída de comunicação serial (SDO- *Serial Data Output*) do sensor digital ADXL345.

O módulo acelerômetro possui os seguintes sinais de saída:

- **DATA_READY**: Sinal que vai a nível lógico alto por 10 µs quando os 2 bytes de aceleração de todos os eixos estão prontos para serem lidos (esse sinal tem periodicidade de 10 ms, correspondendo a uma taxa de leitura de dados de 100 Hz);

- **ACC_X[15..0], ACC_Y[15..0], ACC_Z[15..0]**: Valores de aceleração de cada eixo, no formato de 2 bytes (16 bits), extraídos ao final de cada leitura completa de dados do sensor digital ADXL345;

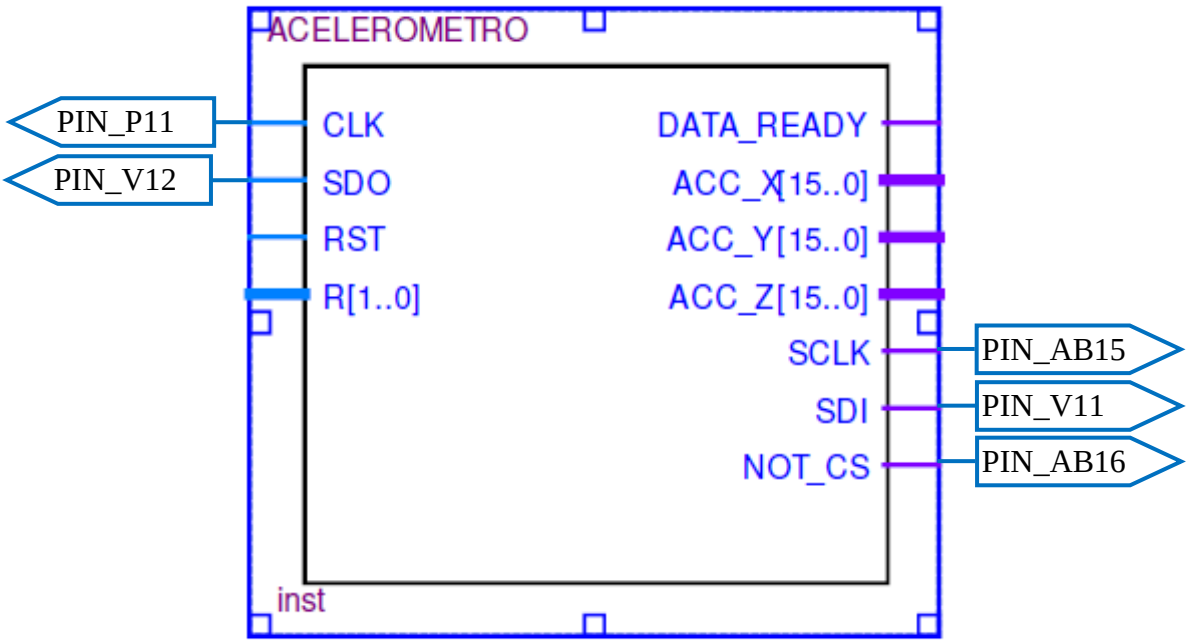
- **SLCK**: Sinal de saída com o clock para comunicação serial (100 kHz). Deve ser conectado ao pino correspondente (de mesmo nome) do sensor digital ADXL345;

- **SDI**: Sinal de saída de dados que deve ser conectado ao canal de entrada de comunicação serial (SDI- *Serial Data Input*) do sensor digital ADXL345 para envio de dados de configuração do módulo acelerador para o sensor;

- **NOT_CS**: Sinal de saída para controle da transmissão serial. Quando esse sinal é colocado em nível lógico ‘0’ inicia-se a transmissão serial SPI. Deve ser conectado ao sinal /CS do sensor digital ADXL345 para que o módulo acelerômetro possa controlar (como mestre) a comunicação serial do canal SPI.

Observa-se que os sinais: SDO, SDI, SCLK e NOT_CS devem ser conectados aos pinos correspondentes do sensor digital ADXL345 contido na plataforma DE10-Lite, conforme numeração indicada na Figura 6). O símbolo do componente do módulo acelerômetro é mostrado a seguir na Figura 6 (ANALOG DEVICES, 2015), (TERASIC, 2018).

Figura 6: Símbolo do **Módulo Acelerômetro**.



Fonte: Autor.

4.2 Descrição VHDL do módulo acelerômetro

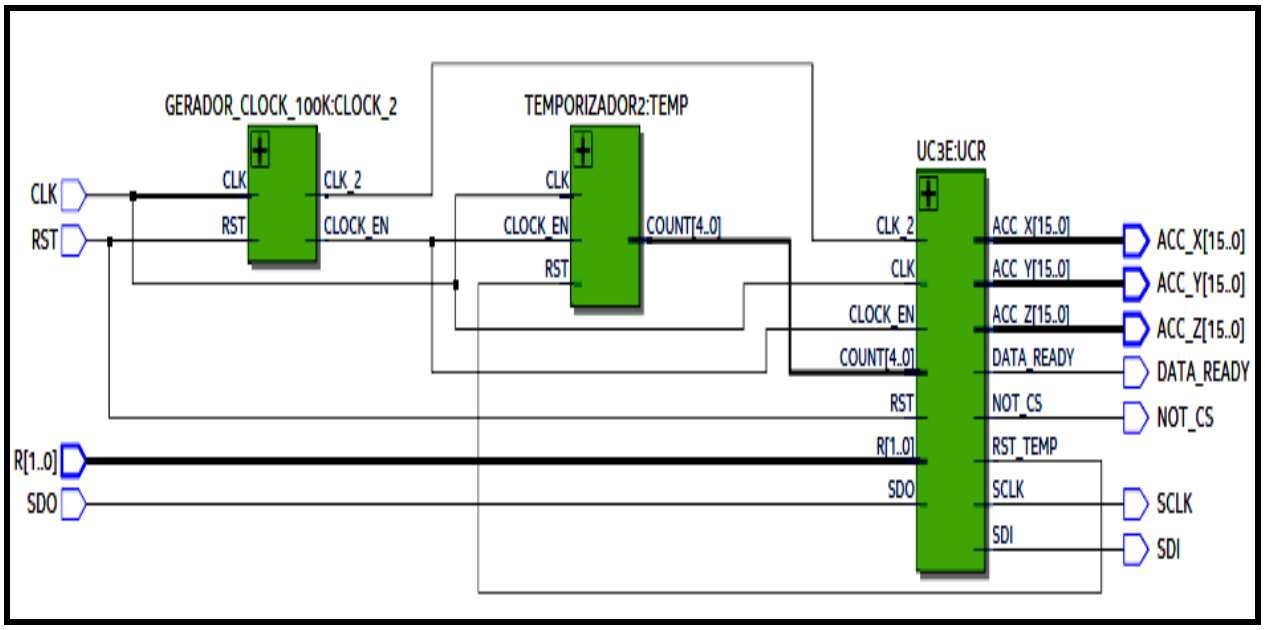
O arquivo principal do código VHDL do módulo acelerômetro (**ACELERÔMETRO.vhd**) consiste na interligação (na descrição no formato estrutural) de três componentes descritos em VHDL: o temporizador (**TEMPORIZADOR2.vhd**), a unidade de controle (**UC3E.vhd**) e o gerador de clock (**GERADOR_CLOCK_100K.vhd**).

Na Figura 7 é apresentada a visão RTL com a interligação dos componentes que compõem esse bloco.

Basicamente, o componente unidade de controle (**UC3E.vhd**) gera os sinais que permitem a comunicação SPI do FPGA MAX10 com o sensor digital ADXL345 (SDO, SDI, SCLK e NOT_CS) e fornece os valores de aceleração em cada eixo através de registradores de 16 bits (ACC_X[15..0], ACC_Y[15..0] e ACC_Z[15..0]). A partir do reset (RST=’0’) a unidade de controle configura do sensor e realiza a leitura contínua de dados de aceleração, que podem ser lidos nos três registradores de 16 bits, a partir da ativação do sinal DATA_READY. O componente temporizador (**TEMPORIZADOR2.vhd**) informa à unidade de controle (através do registrador COUNT[4..0]) quantos dados foram enviados ou recebidos serialmente através da comunicação SPI (esse temporizador é reiniciado pela unidade de controle através do sinal RST_TEMP). O componente gerador de clock (**GERADOR_CLOCK_100K.vhd**) gera um sinal de referência de 200 KHz (CLK_2), o qual será divido por 2 para gerar o sinal de sincronismo da comunicação serial (100KHz). Esse componente também gera um sinal de sincronismo para contagem dos dados seriais (CLK_EN), cuja frequência é 100 KHz, o qual é

utilizado para sincronizar o componente temporizador2 e a unidade de controle, permitindo uma taxa de transferência de registros de dados de 100 leituras/segundo.

Figura 7: Visão RTL do módulo **ACELEROMETRO**.



Fonte: Autor.

A seguir são descritos individualmente como cada um dos componentes do módulo acelerômetro (**ACELERÔMETRO.vhd**) são implementados em VHDL.

O **componente temporizador2** atua como um contador de forma a realizar o sequenciamento da transmissão e recepção dos dados do acelerômetro que ocorre na unidade de controle, já que a troca de dados deve ser realizada de forma serial. Após 16 ciclos de clock o temporizador envia o valor 15 (COUNT= “01111”) para a unidade de controle, para sinalizar que os dezesseis bits da transmissão foram enviados do módulo acelerômetro para o sensor digital ADXL345. Para a parte de leitura de dados do sensor, o temporizador envia o valor 7 (COUNT= “00111”) sinalizando que os 8 primeiros bits da transmissão do sensor foram recebidos e que o acelerômetro enviará o próximo byte de aceleração selecionado a partir do próximo ciclo de clock. Após mais 8 ciclos de clock, o temporizador envia o valor 16 (COUNT= “10000”) para indicar que os 8 bits do valor de aceleração foram lidos do sensor. O código em VHDL do arquivo **TEMPORIZADOR2.vhd** é mostrado na Figura 8.

Figura 8: Código VHDL do **TEMPORIZADOR2.vhd**.

```
1   LIBRARY IEEE;
2   USE IEEE.STD_LOGIC_1164.ALL;
3
4   ENTITY Temporizador2 IS
5      PORT(
6         RST, CLK,CLOCK_EN : IN STD_LOGIC;
7         COUNT : BUFFER STD_LOGIC_VECTOR(4 DOWNTO 0));
8   END Temporizador2;
9
10  ARCHITECTURE FSM OF Temporizador2 IS
11
12  BEGIN
13  PROCESS( CLK, RST )
14     BEGIN
15        IF RST ='0' THEN COUNT <="00000";
16        ELSIF ( RISING_EDGE(CLK) AND CLOCK_EN='1') THEN
17           IF COUNT < "10000" THEN
18              COUNT(0) <= NOT COUNT(0);
19              COUNT(1) <= COUNT(0) XOR COUNT(1);
20              COUNT(2) <= ( COUNT(0) AND COUNT(1) ) XOR COUNT(2);
21              COUNT(3) <= ( COUNT(0) AND COUNT(1) AND COUNT(2) ) XOR COUNT(3);
22              COUNT(4) <= ( COUNT(0) AND COUNT(1) AND COUNT(2) AND COUNT(3)) XOR COUNT(4);
23           END IF;
24        END IF;
25  END PROCESS;
26
27  END FSM;
```

Fonte: Autor.

Outro elemento do **ACELEROMETRO.vhd** é o **componente unidade controle** (**UC3E**) que realiza a troca de dados entre o FPGA e o ADXL345. A lógica da unidade de controle é uma máquina de estados (FSM) com 31 estados (A, B, C, D, E, FX0, GX0, HX0, IX0, FX1, GX1, HX1, IX1, FY0, GY0, HY0, IY0, FY1, GY1, HY1, IY1, FZ0, GZ0, HZ0, IZ0, FZ1, GZ1, HZ1, IZ1, FULL_DATA, J). Essa sequência de estados segue os eventos das formas de onda da Figura 3 e da Figura 4. O sequenciamento de estados do componente unidade de controle **UC3E.vhd** para realizar a escrita no acelerômetro é mostrado no código VHDL da Figura 9.

O estado **A** é o estado inicial (reset) onde a contagem do componente Temporizador2 é zerada (RST_TEMP=’0’) e a transmissão está desabilitada (NOT_CS=’1’ e SET_SCLK=’1’). No estado **B** ocorre o início da contagem do componente Temporizador2 (RST_TEMP=’1’) e a configuração dos dados que serão enviados para o sensor através do sinal SDI. Nesse estado vetor “MEMORY” recebe os dados que devem ser enviados um por vez, para o registrador 44 do sensor. No estado **B**, a transmissão é iniciada (NOT_CS=’0’ e SET_SCLK=’0’) e os dados do vetor “MEMORY” são enviados um a um pelo sinal de saída SDI até que o valor do contador do Temporizador2 seja 15 (COUNT= “01111”). Quando o valor do contador do Temporizador2 for 15, a transmissão é interrompida e a máquina de estados vai ao estado **C**, onde o contador do Temporizador2 é zerado (RST_TEMP=’0’) e a transmissão está desabilitada (NOT_CS=’1’ e SET_SCLK=’1’).

Observe que no estado **B** foi configurado o registrador BW_RATE de acordo com sua tabela de sinais (Tabela 2) (ANALOG DEVICES, 2015), (TERASIC, 2018).

Figura 9: Código VHDL da unidade de controle (**UC3E.vhd**) – Registrador BW_RATE.

```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
USE IEEE.STD_LOGIC_ARITH.ALL;
USE IEEE.STD_LOGIC_UNSIGNED.ALL;

ENTITY UC3E IS
   PORT(
      CLK,RST,SDO,CLOCK_EN,CLK_2 : IN STD_LOGIC;
      SDI : OUT STD_LOGIC;
      SCLK : BUFFER STD_LOGIC;
      NOT_CS, RST_TEMP : OUT STD_LOGIC;
      COUNT : IN STD_LOGIC_VECTOR(4 DOWNTO 0);
      ACC_X0,ACC_X1,ACC_Y0,ACC_Y1,ACC_Z0,ACC_Z1 : BUFFER STD_LOGIC_VECTOR(16 DOWNTO 9);
      ------------------------ENTRADAS E SAÍDAS PARA CONTROLADOR EXTERNO-------------------
      R: IN STD_LOGIC_VECTOR(1 DOWNTO 0);-- SELEÇÃO DE FAIXA DE OPERAÇÃO:2g/4g/8g/16g
      DATA_READY: BUFFER STD_LOGIC; -- AVISO DE LEITURA DE DADOS DOS 3 EIXOS COMPLETA
      ACC_X, ACC_Y, ACC_Z: OUT STD_LOGIC_VECTOR(15 DOWNTO 0)-- DADOS DE ACELERAÇÃO
      );
END UC3E;

ARCHITECTURE FSM OF UC3E IS
SUBTYPE DADOS_SERIAIS IS STD_LOGIC;-- registro de 1 bit (subtype)
TYPE REG_SDI IS ARRAY(0 TO 16) OF DADOS_SERIAIS; -- conjunto de registros da ram
SIGNAL MEMORY: REG_SDI;             --  array de 17 registros

SIGNAL SET_SCLK: STD_LOGIC;

SUBTYPE DADOS_SERIAIS_OUT IS STD_LOGIC;-- registro de 1 bit (subtype)
TYPE REG_SDO IS ARRAY(0 TO 16) OF DADOS_SERIAIS_OUT; -- conjunto de registros da ram
SIGNAL MEMORY_OUT: REG_SDO;       --  array de 17 registros

TYPE ME_STATE IS (A,B,C,D,E,FX0,GX0,HX0,IX0,FX1,GX1,HX1,IX1,FY0,GY0,HY0,IY0,FY1,GY1,
                  HY1,IY1,FZ0,GZ0,HZ0,IZ0,FZ1,GZ1,HZ1,IZ1,FULL_DATA, J);
SIGNAL ST: ME_STATE;

BEGIN
LEITURA_ACELEROMETRO : PROCESS (CLK, RST )
VARIABLE TEMP: STD_LOGIC_VECTOR(13 DOWNTO 0); -- variável de contagem de taxa de dados
BEGIN
IF RST='0' THEN ST <= A; -- reset ou mudança de resolução
             SDI<='0'; NOT_CS<='1'; RST_TEMP<='0';SET_SCLK<='1';DATA_READY<='0';
             TEMP:="00000000000000";
             ACC_X0<="00000000";ACC_X1<="00000000";
             ACC_Y0<="00000000";ACC_Y1<="00000000";
             ACC_Z0<="00000000";ACC_Z1<="00000000";
ELSIF (CLK'EVENT AND CLK='1' AND CLOCK_EN='1') THEN
 CASE ST IS

 -- SEQUÊNCIA DE ESTADOS PARA ESCRITA NO ACELERÔMETRO

   WHEN A =>  ST<=B;
             SDI<='0'; RST_TEMP<='1'; SET_SCLK<='1'; NOT_CS<='0';
             MEMORY(0)<= '0';  --W/R
             MEMORY(1)<= '1';  --MB
             MEMORY(2)<= '1';  --A5
             MEMORY(3)<= '0';  --A4 CONFIGURAÇÃO DO REGISTRADOR 0X2C(44): OUTPUT DATA RATE
             MEMORY(4)<= '1';  --A3
             MEMORY(5)<= '1';  --A2
             MEMORY(6)<= '0';  --A1
             MEMORY(7)<= '0';  --A0
             MEMORY(8)<= '0';  --D7
             MEMORY(9)<= '0';  --D6
             MEMORY(10)<='0';  --D5
             MEMORY(11)<='0';  --D4
             MEMORY(12)<='1';  --D3 TAXA DE COMUNICAÇÃO DE DADOS DE  100 HZ (1010)
             MEMORY(13)<='0';  --D2
             MEMORY(14)<='1';  --D1
             MEMORY(15)<='0';  --D0
             MEMORY(16)<='0';  --BIT NÃO UTILIZADO

   WHEN B => IF COUNT="01111" THEN ST<=C;
             ELSE ST<=B;
             END IF;
             SDI<= MEMORY(conv_integer(unsigned(COUNT)));
             SET_SCLK<='0';NOT_CS<='0';RST_TEMP<='1';

   WHEN C => ST<=D;
             SDI<='0';  RST_TEMP<='0'; SET_SCLK<='1'; NOT_CS<='1';
```

Fonte: Autor.

Outra sequência de escrita importante é a de configuração do registrador DATA_FORMAT (Figura 10). A sequência de estados é semelhante à sequência anterior com a diferença nos valores dos bits armazenados no vetor "MEMORY", que devem seguir a Tabela 4, para configurar o registrador 49. É importante observar que os endereços 14 e 15 do vetor "MEMORY" recebem os bits R1 e R0 que ajustam o valor do range dos dados de aceleração (Tabela 5), sendo estes configurados pelo usuário externo através dos respectivos sinais de entrada R1 e R0 (ANALOG DEVICES, 2015).

Figura 10: Sequência de estados para escrita no registrador DATA_FORMAT do sensor ADXL345.

```
82      WHEN D => ST<=E;
83                RST_TEMP<='1'; NOT_CS<='0';SET_SCLK<='1';
84                MEMORY(0)<= '0';  --W/R
85                MEMORY(1)<= '1';  --MB
86                MEMORY(2)<= '1';  --A5
87                MEMORY(3)<= '1';  --A4 CONFIGURAÇÃO DO REGISTRADOR 0X31(49) PARA DATA_FORMAT
88                MEMORY(4)<= '0';  --A3
89                MEMORY(5)<= '0';  --A2
90                MEMORY(6)<= '0';  --A1
91                MEMORY(7)<= '1';  --A0
92                MEMORY(8)<= '0';  --D7
93                MEMORY(9)<= '0';  --D6 COMUNICAÇÃO EM 4 FIOS (SPI 4-WIRE)
94                MEMORY(10)<='0';  --D5
95                MEMORY(11)<='0';  --D4
96                MEMORY(12)<='0';  --D3
97                MEMORY(13)<='0';  --D2
98                MEMORY(14)<=R(1); --D1 DEFINIÇÃO DO RANGE (±2g A ±16g)
99                MEMORY(15)<=R(0); --D0
100               MEMORY(16)<='0';  --BIT NÃO UTILIZADO
101
102     WHEN E => IF COUNT="01111"  THEN ST<=FX0;
103               ELSE ST<=E;
104               END IF;
105               SDI<= MEMORY(conv_integer(unsigned(COUNT)));
106               SET_SCLK<='0';NOT_CS<='0'; RST_TEMP<='1';
107
108     WHEN FX0 => ST<=GX0;
109               SDI<='0'; RST_TEMP<='0'; SET_SCLK<='1'; NOT_CS<='1'; DATA_READY<='0';
```

Fonte: Autor.

O conjunto de quadros da Sequência de estados para a leitura do registrador DATAX0 do sensor ADXL345.Figura 11 à Figura 16 mostra a descrição em VHDL para o processo de leitura dos dados de aceleração do sensor após a configuração dos registradores realizada acima. A Figura 11 corresponde à leitura do byte menos significativo dos dados de aceleração do eixo x. No estado **GX0** é realizado o início da contagem do Temporizador2 (RST_TEMP='1') e a configuração do vetor "MEMORY_OUT" com os 8 bits de escrita necessários para configurar o registrador DATAX0 e obter os dados desejados. No estado **HX0** é iniciada a transmissão dos 8 bits de escrita através do sinal SDI até que o valor do Temporizador2 seja 7 (COUNT= "00111"), que indica o fim de 8 ciclos de clock necessários para essa transmissão. No estado **IX0**, o sinal ACC_X0 recebe um a um os 8 bits menos significativos de aceleração do eixo x do sensor através do sinal SDO até que o valor do Temporizador2 seja 16 (COUNT= "10000"), que indica a passagem de mais 8 ciclos de clock. No estado **FX1** é desabilitada a transmissão de dados (NOT_CS='1' e SET_SCLK='1'). O mesmo ocorre para os dados de aceleração dos eixos y e z com a diferença apenas no endereço do registrador que contém cada byte de aceleração.

Figura 11: Sequência de estados para a leitura do registrador DATAX0 do sensor ADXL345.

```
111    -- SEQUÊNCIA DE ESTADOS PARA LEITURA DOS DADOS DE ACELERAÇÃO DATAX0
112
113       WHEN GX0 =>  ST<=HX0;
114                RST_TEMP<='1'; NOT_CS<='0';SET_SCLK<='1';
115                MEMORY_OUT(0)<='1';  --W/R
116                MEMORY_OUT(1)<='1';  --MB
117                MEMORY_OUT(2)<='1';  --A5
118                MEMORY_OUT(3)<='1';  --A4
119                MEMORY_OUT(4)<='0';  --A3  CONFIGURAÇÃO DO REGISTRADOR 0X32(50) PARA DATAX0
120                MEMORY_OUT(5)<='0';  --A2
121                MEMORY_OUT(6)<='1';  --A1
122                MEMORY_OUT(7)<='0';  --A0
123                MEMORY_OUT(8)<='0';  --BIT NÃO UTILIZADO
124
125       WHEN HX0 => IF COUNT="00111" THEN ST<=IX0;
126                   ELSE ST<=HX0;
127                   END IF;
128                   SDI<= MEMORY_OUT(conv_integer(unsigned(COUNT)));
129                   SET_SCLK<='0';NOT_CS<='0';RST_TEMP<='1';
130
131       WHEN IX0 => IF COUNT="10000" THEN ST<=FX1;
132                   ELSE ST<=IX0;
133                   END IF;
134                   ACC_X0(conv_integer(unsigned(COUNT)))<=SDO;
135                   SDI<='0';  RST_TEMP<='1'; SET_SCLK<='0'; NOT_CS<='0';
136
137       WHEN FX1 => ST<=GX1;
138                   SDI<='0'; RST_TEMP<='0'; SET_SCLK<='1'; NOT_CS<='1';
```

Fonte: Autor.

Figura 12: Sequência de estados para a leitura do registrador DATAX1 do sensor ADXL345.

```
140    -- SEQUÊNCIA DE ESTADOS PARA LEITURA DOS DADOS DE ACELERAÇÃO DATAX1
141
142       WHEN GX1=>  ST<=HX1;
143                RST_TEMP<='1'; NOT_CS<='0';SET_SCLK<='1';
144                MEMORY_OUT(0)<='1'; --W/R
145                MEMORY_OUT(1)<='1';  --MB
146                MEMORY_OUT(2)<='1';  --A5
147                MEMORY_OUT(3)<='1';  --A4
148                MEMORY_OUT(4)<='0';  --A3  CONFIGURAÇÃO DO REGISTRADOR 0X33(51) PARA DATAX1
149                MEMORY_OUT(5)<='0';  --A2
150                MEMORY_OUT(6)<='1';  --A1
151                MEMORY_OUT(7)<='1';  --A0
152                MEMORY_OUT(8)<='0';  --BIT NÃO UTILIZADO
153
154       WHEN HX1 => IF COUNT="00111" THEN ST<=IX1;
155                   ELSE ST<=HX1;
156                   END IF;
157                SDI<= MEMORY_OUT(conv_integer(unsigned(COUNT)));
158                SET_SCLK<='0';NOT_CS<='0';RST_TEMP<='1';
159
160       WHEN IX1 => IF COUNT="10000"    THEN ST<=FY0;
161                   ELSE ST<=IX1;
162                   END IF;
163                   ACC_X1(conv_integer(unsigned(COUNT)))<=SDO;
164                   SDI<='0';  RST_TEMP<='1'; SET_SCLK<='0'; NOT_CS<='0';
165
166       WHEN FY0 => ST<=GY0;
167                   SDI<='0'; RST_TEMP<='0'; SET_SCLK<='1'; NOT_CS<='1';
```

Fonte: Autor.

Figura 13: Sequência de estados para a leitura do registrador DATAY0 do sensor ADXL345.

```
169    -- SEQUÊNCIA DE ESTADOS PARA LEITURA DOS DADOS DE ACELERAÇÃO DATAY0
170
171        WHEN GY0 =>  ST<=HY0;
172                  RST_TEMP<='1'; NOT_CS<='0';SET_SCLK<='1';
173                  MEMORY_OUT(0)<='1';  --W/R
174                  MEMORY_OUT(1)<='1';  --MB
175                  MEMORY_OUT(2)<='1';  --A5
176                  MEMORY_OUT(3)<='1';  --A4
177                  MEMORY_OUT(4)<='0';  --A3  CONFIGURAÇÃO DO REGISTRADOR 0X34(52) PARA DATAY0
178                  MEMORY_OUT(5)<='1';  --A2
179                  MEMORY_OUT(6)<='0';  --A1
180                  MEMORY_OUT(7)<='0';  --A0
181                  MEMORY_OUT(8)<='0';  --BIT NÃO UTILIZADO
182
183        WHEN HY0 => IF COUNT="00111" THEN ST<=IY0;
184                    ELSE ST<=HY0;
185                    END IF;
186                    SDI<= MEMORY_OUT(conv_integer(unsigned(COUNT)));
187                    SET_SCLK<='0';NOT_CS<='0';RST_TEMP<='1';
188
189        WHEN IY0 => IF COUNT="10000" THEN ST<=FY1;
190                    ELSE ST<=IY0;
191                    END IF;
192                    ACC_Y0(conv_integer(unsigned(COUNT)))<=SDO;
193                    SDI<='0';  RST_TEMP<='1'; SET_SCLK<='0'; NOT_CS<='0';
194
195        WHEN FY1 => ST<=GY1;
196                     SDI<='0'; RST_TEMP<='0'; SET_SCLK<='1'; NOT_CS<='1';
```

Fonte: Autor.

Figura 14: Sequência de estados para a leitura do registrador DATAY1 do sensor ADXL345.

```
198    -- SEQUÊNCIA DE ESTADOS PARA LEITURA DOS DADOS DE ACELERAÇÃO DATAY1
199
200        WHEN GY1=>  ST<=HY1;
201                  RST_TEMP<='1'; NOT_CS<='0';SET_SCLK<='1';
202                  MEMORY_OUT(0)<='1'; --W/R
203                  MEMORY_OUT(1)<='1';  --MB
204                  MEMORY_OUT(2)<='1';  --A5
205                  MEMORY_OUT(3)<='1';  --A4
206                  MEMORY_OUT(4)<='0';  --A3  CONFIGURAÇÃO DO REGISTRADOR 0X35(53) PARA DATAY1
207                  MEMORY_OUT(5)<='1';  --A2
208                  MEMORY_OUT(6)<='0';  --A1
209                  MEMORY_OUT(7)<='1';  --A0
210                  MEMORY_OUT(8)<='0';  --BIT NÃO UTILIZADO
211
212        WHEN HY1 => IF COUNT="00111" THEN ST<=IY1;
213                    ELSE ST<=HY1;
214                    END IF;
215                  SDI<= MEMORY_OUT(conv_integer(unsigned(COUNT)));
216                  SET_SCLK<='0';NOT_CS<='0';RST_TEMP<='1';
217
218        WHEN IY1 => IF COUNT="10000"    THEN ST<=FZ0;
219                    ELSE ST<=IY1;
220                    END IF;
221                    ACC_Y1(conv_integer(unsigned(COUNT)))<=SDO;
222                    SDI<='0';  RST_TEMP<='1'; SET_SCLK<='0'; NOT_CS<='0';
223
224        WHEN FZ0 => ST<=GZ0;
225                    SDI<='0'; RST_TEMP<='0'; SET_SCLK<='1'; NOT_CS<='1';
```

Fonte: Autor.

Figura 15: Sequência de estados para a leitura do registrador DATAZ0 do sensor ADXL345.

```
227   -- SEQUÊNCIA DE ESTADOS PARA LEITURA DOS DADOS DE ACELERAÇÃO DATAZ0
228
229       WHEN GZ0 =>  ST<=HZ0;
230                 RST_TEMP<='1'; NOT_CS<='1';SET_SCLK<='1';
231                 MEMORY_OUT(0)<='1';  --W/R
232                 MEMORY_OUT(1)<='1';  --MB
233                 MEMORY_OUT(2)<='1';  --A5
234                 MEMORY_OUT(3)<='1';  --A4
235                 MEMORY_OUT(4)<='0';  --A3  CONFIGURAÇÃO DO REGISTRADOR 0X36(54) PARA DATAZ0
236                 MEMORY_OUT(5)<='1';  --A2
237                 MEMORY_OUT(6)<='1';  --A1
238                 MEMORY_OUT(7)<='0';  --A0
239                 MEMORY_OUT(8)<='0';  --BIT NÃO UTILIZADO
240
241       WHEN HZ0 => IF COUNT="00111" THEN ST<=IZ0;
242                   ELSE ST<=HZ0;
243                   END IF;
244                   SDI<= MEMORY_OUT(conv_integer(unsigned(COUNT)));
245                   SET_SCLK<='0';NOT_CS<='0';RST_TEMP<='1';
246
247       WHEN IZ0 => IF COUNT="10000" THEN ST<=FZ1;
248                   ELSE ST<=IZ0;
249                   END IF;
250                   ACC_Z0(conv_integer(unsigned(COUNT)))<=SDO;
251                   SDI<='0';  RST_TEMP<='1'; SET_SCLK<='0'; NOT_CS<='0';
252
253       WHEN FZ1 => ST<=GZ1;
254                     SDI<='0'; RST_TEMP<='0'; SET_SCLK<='1'; NOT_CS<='1';
```

Fonte: Autor.

Figura 16: Sequência de estados para a leitura do registrador DATAZ1 do sensor ADXL345.

```
256   -- SEQUÊNCIA DE ESTADOS PARA LEITURA DOS DADOS DE ACELERAÇÃO DATAZ1
257
258       WHEN GZ1=>  ST<=HZ1;
259                 RST_TEMP<='1'; NOT_CS<='0';SET_SCLK<='1';
260                 MEMORY_OUT(0)<='1'; --W/R
261                 MEMORY_OUT(1)<='1';  --MB
262                 MEMORY_OUT(2)<='1';  --A5
263                 MEMORY_OUT(3)<='1';  --A4
264                 MEMORY_OUT(4)<='0';  --A3  CONFIGURAÇÃO DO REGISTRADOR 0X37(55) PARA DATAZ1
265                 MEMORY_OUT(5)<='1';  --A2
266                 MEMORY_OUT(6)<='1';  --A1
267                 MEMORY_OUT(7)<='1';  --A0
268                 MEMORY_OUT(8)<='0';  --BIT NÃO UTILIZADO
269
270       WHEN HZ1 => IF COUNT="00111" THEN ST<=IZ1;
271                   ELSE ST<=HZ1;
272                   END IF;
273                 SDI<= MEMORY_OUT(conv_integer(unsigned(COUNT)));
274                 SET_SCLK<='0';NOT_CS<='0';RST_TEMP<='1';
275
276       WHEN IZ1 => IF COUNT="10000" THEN ST<=FULL_DATA;
277                   ELSE ST<=IZ1;
278                   END IF;
279                   ACC_Z1(conv_integer(unsigned(COUNT)))<=SDO;
280                   SDI<='0';  RST_TEMP<='1'; SET_SCLK<='0'; NOT_CS<='0';
281
282       WHEN FULL_DATA => ST<=J;
283                 SDI<='0'; RST_TEMP<='0';SET_SCLK<='1';DATA_READY<='1';NOT_CS<='1';
284
285       WHEN J => IF TEMP< "00001110000100" THEN TEMP:= TEMP+1;ST<=J; --CONTA ATÉ 900
286                 ELSE TEMP:="00000000000000"; ST<=FX0;
287                 END IF;
288                 DATA_READY<='0';
289
290       WHEN OTHERS => NULL;
291     END CASE;
292   END IF;
293   END PROCESS LEITURA_ACELEROMETRO;
```

Fonte: Autor.

Na Figura 16 além dos estados para a leitura do segundo byte dos dados de aceleração do eixo z, há o estado **FULL_DATA**, no qual o sinal DATA_READY é ativado para sinalizar ao usuário que os dados dos três eixos do sensor foram lidos do sensor e estão disponíveis na saída do módulo através das saídas ACC_X, ACC_Y e ACC_Z (ANALOG DEVICES, 2015). Adicionalmente, no estado **J** é realizada uma temporização, com a contagem de 900 eventos de CLOCK_EN, equivalente à 9 ms (uma vez que o período entre dois pulsos de CLOCK_EN é de 10 µs). Essa temporização faz com que a atualização da leitura dos dados do sensor ocorra a cada 10 ms, resultando uma taxa de leitura de 100 amostras/segundo.

A Figura 17 mostra a parte final do código do componente unidade de controle do módulo acelerômetro. A configuração do sinal de clock para sincronização de dados (SCLK) é realizada com um processo que habilita e desabilita o clock de acordo com a condição da variável SET_SCLK. O sinal SET_SCLK desabilita o clock (SCLK='1') quando ativo. Quando SET_SCLK='0' as bordas de subida do sinal CLK_2 são utilizadas para gerar o clock de comunicação (SCLK) de 100 KHz.

Figura 17: Parte final do arquivo **UC3E.vhd** – Carga de Dados de Aceleração (X, Y, Z).

```
295    -- CONFIGURAÇÃO DO SCLK:
296
297    PROCESS(CLK_2,SET_SCLK)
298    BEGIN
299    IF SET_SCLK='1' THEN SCLK<='1';
300    ELSIF (CLK_2'event and CLK_2='1') THEN SCLK<=NOT(SCLK);
301    END IF;
302    END PROCESS;
303
304    -- CARGA DOS DADOS NOS REGISTROS DE SAÍDA:
305
306    CARREGAR_DADOS : PROCESS(DATA_READY,RST)
307    BEGIN
308    IF RST='0' THEN ACC_X<="0000000000000000";
309                    ACC_Y<="0000000000000000";
310                    ACC_Z<="0000000000000000";
311    ELSIF DATA_READY='1' THEN
312       ACC_X(7)<=ACC_X0(9);  ACC_X(15)<=ACC_X1(9);
313       ACC_X(6)<=ACC_X0(10); ACC_X(14)<=ACC_X1(10);
314       ACC_X(5)<=ACC_X0(11); ACC_X(13)<=ACC_X1(11);
315       ACC_X(4)<=ACC_X0(12); ACC_X(12)<=ACC_X1(12);
316       ACC_X(3)<=ACC_X0(13); ACC_X(11)<=ACC_X1(13);
317       ACC_X(2)<=ACC_X0(14); ACC_X(10)<=ACC_X1(14);
318       ACC_X(1)<=ACC_X0(15); ACC_X(9) <=ACC_X1(15);
319       ACC_X(0)<=ACC_X0(16); ACC_X(8) <=ACC_X1(16);
320       ACC_Y(7)<=ACC_Y0(9);  ACC_Y(15)<=ACC_Y1(9);
321       ACC_Y(6)<=ACC_Y0(10); ACC_Y(14)<=ACC_Y1(10);
322       ACC_Y(5)<=ACC_Y0(11); ACC_Y(13)<=ACC_Y1(11);
323       ACC_Y(4)<=ACC_Y0(12); ACC_Y(12)<=ACC_Y1(12);
324       ACC_Y(3)<=ACC_Y0(13); ACC_Y(11)<=ACC_Y1(13);
325       ACC_Y(2)<=ACC_Y0(14); ACC_Y(10)<=ACC_Y1(14);
326       ACC_Y(1)<=ACC_Y0(15); ACC_Y(9) <=ACC_Y1(15);
327       ACC_Y(0)<=ACC_Y0(16); ACC_Y(8) <=ACC_Y1(16);
328       ACC_Z(7)<=ACC_Z0(9);  ACC_Z(15)<=ACC_Z1(9);
329       ACC_Z(6)<=ACC_Z0(10); ACC_Z(14)<=ACC_Z1(10);
330       ACC_Z(5)<=ACC_Z0(11); ACC_Z(13)<=ACC_Z1(11);
331       ACC_Z(4)<=ACC_Z0(12); ACC_Z(12)<=ACC_Z1(12);
332       ACC_Z(3)<=ACC_Z0(13); ACC_Z(11)<=ACC_Z1(13);
333       ACC_Z(2)<=ACC_Z0(14); ACC_Z(10)<=ACC_Z1(14);
334       ACC_Z(1)<=ACC_Z0(15); ACC_Z(9) <=ACC_Z1(15);
335       ACC_Z(0)<=ACC_Z0(16); ACC_Z(8) <=ACC_Z1(16);
336    END IF;
337    END PROCESS CARREGAR_DADOS;
338
339    END FSM;
```

Fonte: Autor.

O processo Carregar_Dados (Figura 17) realiza a montagem dos dados de saída a partir da ativação do sinal DATA_READY, permitindo a leitura dos valores de aceleração quando houver a aquisição completa dos registros do sensor. Observa-se que a contagem do índice de leitura dos vetores ACC_X0, ACC_X1, ACC_Y0, ACC_Y1, ACC_Z0, ACC_Z1 é crescente, mas o envio dos dados do sensor de aceleração ocorre do bit mais significativo para o bit menos significativo (Figura 4), logo é necessário um ajuste de posição na transferência dos bits para os registradores ACC_X, ACC_Y, ACC_Z que é realizado no processo Carregar_Dados, conforme representado na Figura 17.

O código VHDL do componente gerador de clock de 100 KHz (Figura 18) é composto de um contador binário de 9 bits (variável de contagem: temp[8..0]), portanto com módulo 500. Esse contador retorna a zero a cada 500 bordas de subida do clock de 50 MHz (disponível na plataforma DE10-Lite). Dessa forma, o ciclo do contador se repete na frequência de 100 KHz (50 MHz/500).

Figura 18: Código VHDL do arquivo **GERADOR_CLOCK_100K.vhd**.

```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;

ENTITY GERADOR_CLOCK_100K IS PORT(
   CLK       : IN STD_LOGIC;
   RST       : IN STD_LOGIC;
   CLOCK_EN, CLK_2   : OUT STD_LOGIC);
END GERADOR_CLOCK_100K ;

ARCHITECTURE COMPORTAMENTAL OF GERADOR_CLOCK_100K  IS
SIGNAL temp: STD_LOGIC_VECTOR( 8 DOWNTO 0 );

BEGIN
GERANDO_CLK_2: PROCESS( CLK, RST )
   BEGIN
      IF RST ='0'  THEN
         temp <="000000000"  ; -- carrega valor inicial: 0
      ELSIF rising_edge(CLK) THEN
         IF temp < "111110011" THEN -- verifica se temp < 499
         temp(0) <= NOT temp(0);
         temp(1) <= temp(0) XOR temp(1);
         temp(2) <= ( temp(0) AND temp(1) ) XOR temp(2);
         temp(3) <= ( temp(0) AND temp(1) AND temp(2) ) XOR temp(3);
         temp(4) <= ( temp(0) AND temp(1) AND temp(2) AND temp(3)) XOR temp(4);
         temp(5) <= ( temp(0) AND temp(1) AND temp(2) AND temp(3) AND temp(4))
                    XOR temp(5);
         temp(6) <= ( temp(0) AND temp(1) AND temp(2) AND temp(3) AND temp(4)
                    AND temp(5))XOR temp(6);
         temp(7) <= ( temp(0) AND temp(1) AND temp(2) AND temp(3) AND temp(4)
                    AND temp(5) AND temp(6))XOR temp(7);
         temp(8) <= ( temp(0) AND temp(1) AND temp(2) AND temp(3) AND temp(4)
                    AND temp(5) AND temp(6) AND temp(7))XOR temp(8);
         ELSE  temp <="000000000";
         END IF;
      END IF;
END PROCESS;

-- Atualiza saida de CLK_2 (200 KHz) para gerar SCLK de 100 KHz:
CLK_2<= temp(7);

-- Atualiza pulso de clock enable:
CLOCK_EN <= '1' WHEN temp = "111110011"  ELSE '0';

END COMPORTAMENTAL;
```

Fonte: Autor.

O componente gerador de clock de 100 KHz fornece um sinal de referência (**CLK_2**) para obtenção do clock de comunicação (**SCLK**). O sinal CLK-2 é obtido a partir do segundo bit mais significativo do contador (temp(7)), logo o sinal CLK_2 tem frequência de 200 KHz (que é divido por dois no componente **UC3E** para a obtenção do sinal **SCLK**). O componente gerador de clock de 100 KHz fornece também o sinal de habilitação de clock (**CLOCK_EN**) que é gerado quando o contador atinge a contagem máxima (500) e tem duração de um período do clock de 50 MHz. Esse sinal é utilizado pelo componente **Temporizador2** para contagem do número de bits transferidos no canal de comuicação SPI e pelo componente unidade de controle (**UC3E**) para controle da máquina de estados de escrita e leitura dos registros do sensor digital ADXL345. O arquivo é mostrado na Figura 18 (TERASIC, 2018).

O código VHDL do módulo do acelerômetro é a junção desses três componentes descritos no formato estrutural, conforme a Figura 19 (cuja visão RTL foi mostrada na Figura 7).

Os sinais de comunicação SPI com o sensor ADXL345 presente na plataforma DE10-Lite (SDO, SDI, SCLK e NOT_CS) devem ser **obrigatoriamente conectados** a pinos específicos do FPGA MAX10, conforme indicado na Tabela 6 (ANALOG DEVICES, 2015), (TERASIC, 2018).

Tabela 6: Associação de pinos obrigatória para do módulo **ACELEROMETRO**.

Node Name, Direction	Location
CLK, Input	PIN_P11
NOT_CS, Output	PIN_AB16
SCLK, Output	PIN_AB15
SDI, Output	PIN_V11
SDO, Input	PIN_V12

Fonte: TERASIC, 2018.

Observação: No Quartus Prime a atribuição de pinos realizada em um projeto (utilizando a opção *Assignments> Pin Planner*) pode ser transferida para outra pasta de projeto (desde que o arquivo do projeto tenha o mesmo nome) utilizando as opções: *Assignments> Export Assignments* e *Assignments> Import Assignments.* Os arquivos contendo a configuração de pinos possuem a extensão: "**.qsf**" (por exemplo: **ACELEROMETRO.qsf**) e podem ser encontrados na principal de cada projeto.

Figura 19: Arquivo VHDL completo do módulo **ACELEROMETRO**.

```
--*****************************************************************************
-- CENTRO UNIVERSITARIO FEI
-- Projeto de Iniciação Didática de Sistemas Digitais
-- Autor: Rhaycen R. Prates - 04/2019
-- Revisor: Prof. Valter F. Avelino - 01/2021
-- Componente VHDL: ACELERÔMETRO => Acelerometro.vhd
-- Rev. 0
-- Especificações: Entradas: CLK, RST, R[1..0], LOAD_DATA
--                 Saidas: DATA_READY, ACC_X[15..0], ACC_Y[15..0],ACC_Z[15..0]
--                 Interface com ADXL345: SDO, SCLK, SDI, NOT_CS
-- Esse código realiza a comunicação com o sensor ADXL345, permitindo o controle
-- da leitura do acelerômetro presente na placa DE10-Lite (Terasic).
-- Esse código foi desenvolvido para ser utilizado na disciplina de
-- Laboratório de Sistemas Digitais II do Centro Universitário FEI.
--*****************************************************************************
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;

ENTITY ACELEROMETRO IS
   PORT(
      CLK: IN STD_LOGIC; -- CLOCK DE 50 MHZ
      SDO: IN STD_LOGIC; -- ENTRADA DE DADOS DO SENSOR ADXL345
      RST: IN STD_LOGIC; -- SINAL DE CONTROLE DE CONFIGURAÇÃO DE REGISTROS
      R: IN STD_LOGIC_VECTOR(1 DOWNTO 0); -- SINAIS DE CONFIGURAÇÃO DE RANGE
      DATA_READY: OUT STD_LOGIC; -- SINAL DE DADOS PRONTOS
      ACC_X, ACC_Y, ACC_Z: OUT STD_LOGIC_VECTOR(15 DOWNTO 0);-- DADOS DE ACELERAÇÃO
      SCLK, SDI, NOT_CS: OUT STD_LOGIC -- SINAIS DE COMUNICAÇÃO COM SENSOR ADXL345
       );
END ACELEROMETRO ;

ARCHITECTURE ESTRUTURAL OF  ACELEROMETRO  IS
SIGNAL COUNT: STD_LOGIC_VECTOR(4 DOWNTO 0);
SIGNAL RST_TEMP, CLOCK_EN,CLK_2: STD_LOGIC;
SIGNAL ACC_X0, ACC_X1, ACC_Y0, ACC_Y1, ACC_Z0, ACC_Z1: STD_LOGIC_VECTOR(16 DOWNTO 9);

COMPONENT UC3E PORT(
      CLK,RST,SDO,CLOCK_EN,CLK_2 : IN STD_LOGIC;
      NOT_CS, SCLK,RST_TEMP : OUT STD_LOGIC;
      COUNT : IN STD_LOGIC_VECTOR(4 DOWNTO 0);
      ACC_X0, ACC_X1, ACC_Y0, ACC_Y1, ACC_Z0, ACC_Z1: OUT STD_LOGIC_VECTOR(16 DOWNTO 9);
      SDI : OUT STD_LOGIC;
      ------------------------ENTRADAS E SAÍDAS PARA CONTROLE EXTERNO--------------
      R: IN STD_LOGIC_VECTOR(1 DOWNTO 0);-- SELEÇÃO DE RANGE (2g A 16g)
      DATA_READY: OUT STD_LOGIC; -- SINAL DE LEITURA DE DADOS DOS 3 EIXOS COMPLETADA
      ACC_X, ACC_Y, ACC_Z: OUT STD_LOGIC_VECTOR(15 DOWNTO 0) );-- DADOS DE ACELERAÇÃO
END COMPONENT UC3E;

COMPONENT TEMPORIZADOR2  PORT(
      RST, CLK,CLOCK_EN : IN STD_LOGIC;
      COUNT : BUFFER STD_LOGIC_VECTOR(4 DOWNTO 0) );
END COMPONENT TEMPORIZADOR2;

COMPONENT GERADOR_CLOCK_100K PORT(
   CLK, RST : IN STD_LOGIC;
   CLOCK_EN, CLK_2: OUT STD_LOGIC );
END COMPONENT GERADOR_CLOCK_100K;

BEGIN
   UCR : UC3E PORT MAP( CLK=>CLK, RST=>RST, SDO=>SDO, CLOCK_EN=>CLOCK_EN, CLK_2=> CLK_2,
         NOT_CS=>NOT_CS, SCLK=>SCLK, RST_TEMP=>RST_TEMP, COUNT=>COUNT, SDI=>SDI, R=>R,
         ACC_X0=>ACC_X0, ACC_X1=>ACC_X1, ACC_Y0=>ACC_Y0, ACC_Y1=>ACC_Y1,ACC_Z0=>ACC_Z0,
         ACC_Z1=>ACC_Z1, ACC_X=>ACC_X, ACC_Y=>ACC_Y,ACC_Z=>ACC_Z,DATA_READY=>DATA_READY);
   TEMP: TEMPORIZADOR2 PORT MAP(CLK=>CLK, COUNT=>COUNT,RST=>RST_TEMP,CLOCK_EN=>CLOCK_EN);
   CLOCK_2:GERADOR_CLOCK_100K PORT MAP(CLK=>CLK,RST=>RST,CLOCK_EN=>CLOCK_EN,CLK_2=>CLK_2);

END ESTRUTURAL;
```

Fonte: Autor.

5 Exemplos de Projeto

Para facilitar a aplicação do módulo acelerômetro nesta seção são apresentados dois exemplos de aplicação básica: um medidor de inclinação e um medidor de aceleração da gravidade.

Como o objetivo é permitir uma visualização da utilização do componente em projetos práticos nesta seção é apresentado o processo de desenvolvimento desses exemplos de projetos, aplicando os recursos da linguagem VHDL, utilizada nos Laboratórios de Sistemas Digitais I e II do Centro Universitário FEI (AVELINO, 2019).

5.1 Projeto 1: Medidor de Inclinação

O primeiro exemplo de aplicação do módulo do acelerômetro é um medidor de inclinação, denominado de **DISPLAY_XYZ.vhd**, cuja saída são os seis displays (HEX5, HEX4, HEX3, HEX2, HEX1 e HEX0) e os dez Led´s (LED[9..0]) presentes na plataforma DE10-Lite. Os displays de sete segmentos mostram o ângulo de inclinação da placa DE10-Lite em cada um dos eixos: eixo X (HEX1 e HEX0), eixo Y (HEX3 e HEX2) e eixo Z (HEX5 e HEX4). Ao se movimentar a plataforma em torno de cada eixo o valor da inclinação angular (na faixa de -90° a +90°) é apresentado em graus no respectivo par de displays (no formato de dezena e unidade). Ao se alterar a posição do sensor ADXL345 em torno do eixo X os dez Led´s da placa DE10-Lite acendem progressivamente de acordo com o ângulo que o acelerômetro estiver com relação ao eixo horizontal, correspondendo a uma apresentação no formato de gráfico de barras (Bargraph) com dez Led's.

Através do módulo do acelerômetro é possível realizar a leitura de 16 bits que são fornecidos pelo ADXL345. Entretanto, apenas os 10 bits menos significativos (D[9..0]) representam o valor da aceleração em código binário (complemento de 2), sendo 9 bits de valor (D[8..0]) e um bit de sinal (D[9]). Neste projeto considerou-se que o range de referência utilizado seria de ±8.g (para reduzir ruído e operar com valores de até dois dígitos de representação). Nessa faixa de operação a sensibilidade do sensor ADXL345 para a inclinação em cada eixo é de 64 LSB/g. Como a aceleração estática do ADXL345 equivalente a 1g para uma inclinação de 90° (conforme representado na Figura 20, então, uma variação de 90° provoca resulta em uma variação de 64 níveis no valor da aceleração.

A Tabela 7 mostra os valores típicos esperados para os dez bits obtidos com a inclinação do eixo X entre -90° e +90°. Observou-se que os três bits menos significativos dos dados de aceleração de cada eixo apresentavam muita variabilidade (devido à sensibilidade do sensor), com isso os valores das medidas apresentam variações frequentes. A numeração equivalente na coluna valor em decimal (e hexadecimal) representada na Tabela 7 considera que os valores binários estão em complemento de 2 e que os valores com inclinação negativa (com a rotação do sensor no sentido horário) são números negativos. A partir dessa tabela foi realizada a conversão de valores de inclinação em graus para ser apresentada nos displays do medidor de inclinação (ANALOG DEVICES, 2015), (TERASIC, 2018).

Figura 20: Sensibilidade do sensor ADXL345 em relação aos eixos de aceleração.

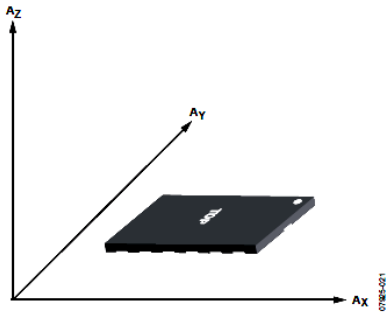


Figure 57. Axes of Acceleration Sensitivity (Corresponding Output Voltage Increases When Accelerated Along the Sensitive Axis)

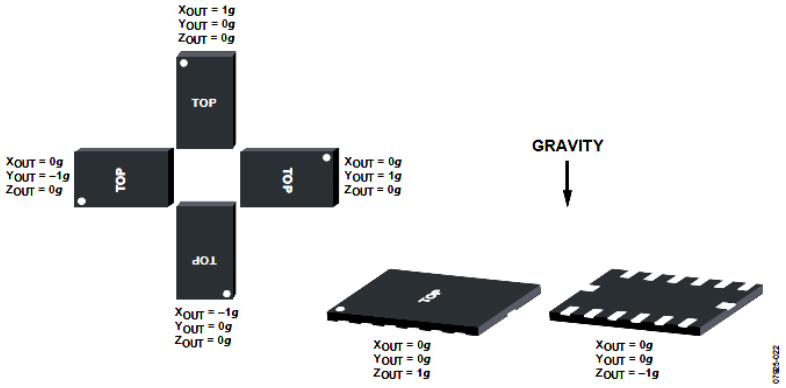


Figure 58. Output Response vs. Orientation to Gravity

Fonte: ANALOG DEVICES, 2015.

Tabela 7: Relação entre os valores lidos do ADXL345 e a inclinação sensor no eixo X.

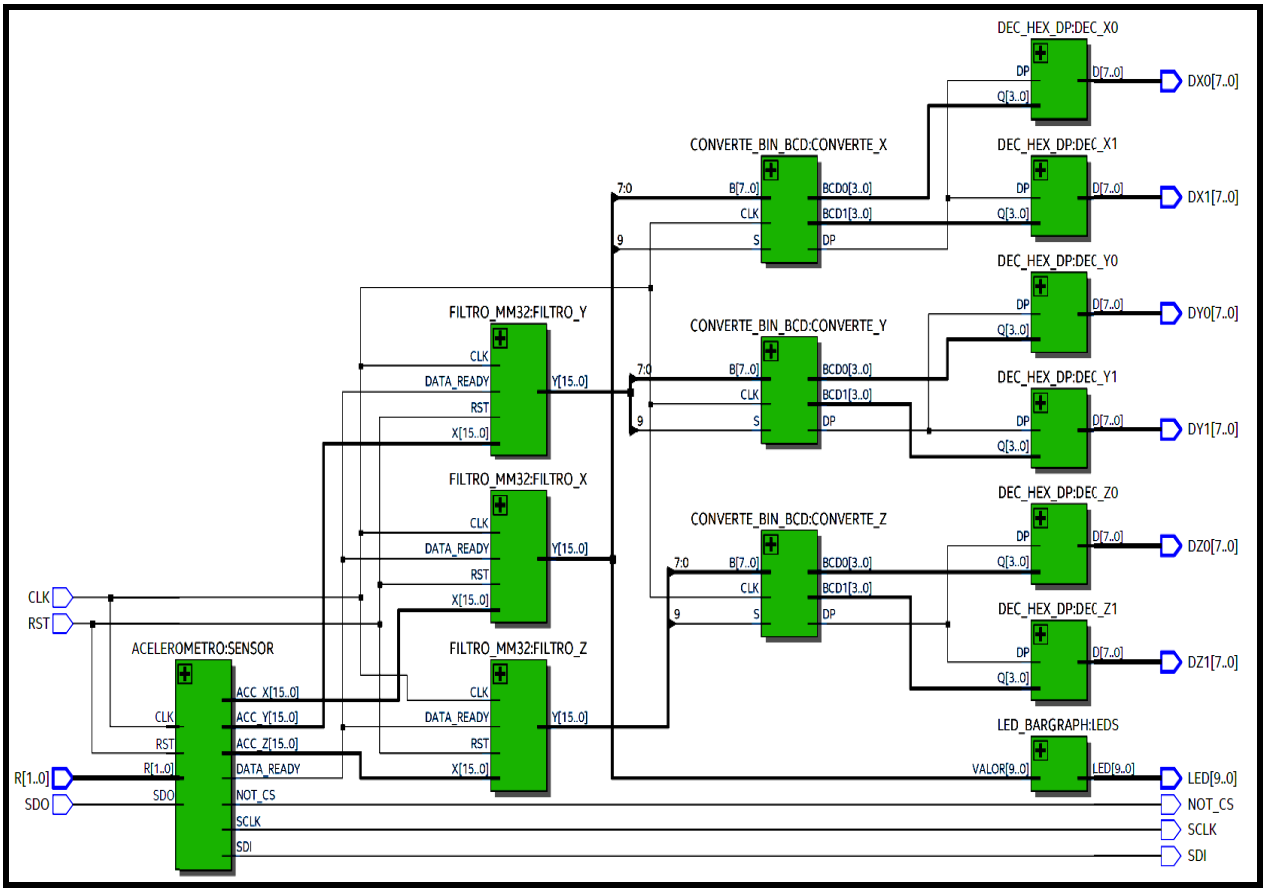
ÂNGULO (º)	Dados lidos do sensor ADXL345 no eixo X (D[9..0])										Valor Decimal	Valor Hexa
	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0		
90	0	0	0	1	0	0	0	0	0	0	64	040h
60	0	0	0	0	1	0	1	0	1	0	42	02Ah
30	0	0	0	0	0	1	0	1	0	1	21	015h
10 (anti-horário)	0	0	0	0	0	0	0	1	1	1	7	007h
00	0	0	0	0	0	0	0	0	0	0	0	000h
-10 (horário)	1	1	1	1	1	1	1	0	0	1	-7	3F9h
-30	1	1	1	1	1	0	1	0	1	1	-21	3EBh
-60	1	1	1	1	0	1	0	1	1	0	-42	3D6h
-90	1	1	1	1	0	0	0	0	0	0	-64	3C0h

Fonte: Autor.

Para a representação da inclinação em graus (nos displays) foi utilizado um fator de escala de 1,4, de modo que o valor decimal 64 representa 90° (e o valor -64 corresponde a -90°). Já para a representação do valor no Bargraph de Led's os valores binários foram somados ao valor decimal 64, fazendo com que a escala de -90° a +90° seja representada em uma faixa de valores decimais entre 0 e 128.

O código VHDL do medidor de inclinação (**DISPLAY_XYZ.vhd**) é constituído de cinco tipos de componentes: o módulo acelerômetro (**ACELEROMETRO.vhd**), um filtro de média móvel de ordem 32 (**FILTRO_MM32.vhd**), um conversor de código binário para BCD (**CONVERTE_BIN_BCD.vhd**), um decodificador para display de sete segmentos (**DEC_HEX_DP.vhd**) e um decodificador para bargraph (**LED_BARGRAPH.vhd**). A visão RTL da interligação desses componentes é mostrada na Figura 21.

Figura 21: Visão RTL do projeto **DISPLAY_XYZ**.



Fonte: Autor.

O módulo acelerômetro (**ACELEROMETRO.vhd**) comunica-se com o sensor ADXL345 através dos sinais de comunicação serial SPI (SDO, SDI, SCLCK, NOT_CS) e fornece os valores de leitura de aceleração em cada um dos eixos (ACC_X[15..0], ACC_Y[15..0], ACC_Z[15..0]). Conforme descrito na **seção 4.2**, esse módulo recebe o sinal de clock principal (CLK) de 50 MHz e está configurado para realizar a leitura dos dados de aceleração na taxa de 100 leituras/segundo. A faixa de medida da aceleração (*Range*) pode ser

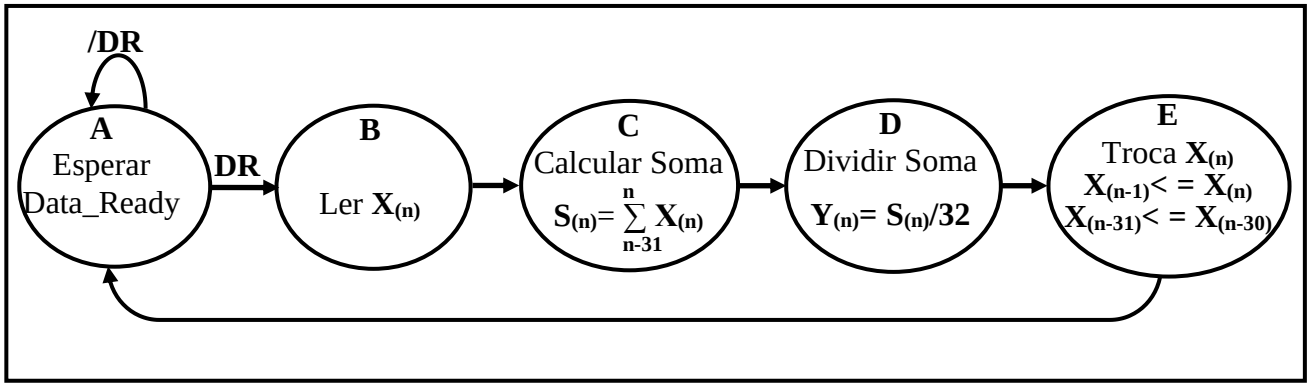
definida através dos sinais de entrada R[1..0], sendo que o projeto do medidor de inclinação considera que a faixa de medida da aceleração é de ±8.g (correspondendo à condição das entradas: R[1] = '1' e R[0] = '0'). A cada leitura de dados realizada o sinal DATA_READY é ativado por um período do clock de transmissão de dados (10 µs)

Cada sinal lido de um dos eixos do acelerômetro é enviado a um componente que realiza um filtro de média móvel de ordem 32 (**FILTRO_MM32.vhd**). A função desse componente é atenuar as flutuações (ruídos) dos sinais de aceleração lidos do sensor, sendo que para isso realiza a média dos últimos 32 valores lidos. Esse componente implementa a expressão:

$$Y_{(n)}=(x_{(n)} +x_{(n-1)} +x_{(n-2)} + ...+x_{(n-30)} +x_{(n-31)})/ 32$$

A realização dessa função é implementada com uma máquina de estados de cinco estados (representada na Figura 22).

Figura 22: Diagrama de estados do filtro de média móvel de ordem 32.



Fonte: Autor.

No estado **A** o componente aguarda o pulso de DATA_READY, que informa a leitura de um novo valor de aceleração (amostra $X_{(n)}$). No estado **B** o valor da amostra atual é inserido no registrador $X_{(n)}$. No estado **C** calcula-se a somatória de todas as últimas 32 amostras (leituras de aceleração) realizadas ($S_{(n)}= X_{(n)} +X_{(n-1)} +X_{(n-2)} +...+ X_{(n-31)}$). No estado **D** a somatória é dividida por 32 ($Y_{(n)}= S_{(n)}/32$), sendo esse o valor de saída do filtro. Essa operação é realizada com o deslocamento do vetor binário $S_{(n)}$ de quatro posições à direita, tendo o cuidado de completar os bits mais significativos com zeros (se $S_{(n)}$ for positivo) ou com uns (se $S_{(n)}$ for negativo). No estado **E** os valores de todos os 31 últimos registros são atualizados de modo que os valores de $X_{(n-1)}$ a $X_{(n-31)}$ devem corresponder aos valores anteriores de $X_{(n)}$ a $X_{(n-30)}$. A máquina de estados retorna ao estado **A** para aguardar novo valor de $X_{(n)}$. O arquivo com o código VHDL do componente filtro de média móvel de ordem 32 é mostrado na Figura 23.

Figura 23: Código VHDL do arquivo **FILTRO_MM32.vhd**.

```
13   LIBRARY IEEE;
14   USE IEEE.STD_LOGIC_1164.ALL;
15   USE IEEE.STD_LOGIC_ARITH.ALL;
16   USE IEEE.STD_LOGIC_UNSIGNED.ALL;
17
18   ENTITY FILTRO_MM32 IS
19      PORT(
20         CLK, RST : IN STD_LOGIC;
21         DATA_READY : IN STD_LOGIC;
22         X : IN STD_LOGIC_VECTOR(15 DOWNTO 0);
23         Y : OUT STD_LOGIC_VECTOR(15 DOWNTO 0) );
24   END FILTRO_MM32;
25
26   ARCHITECTURE FSM OF FILTRO_MM32 IS
27   TYPE ME_ESTADO IS (A,B,C,D,E);
28   SIGNAL ST: ME_ESTADO;
29   SIGNAL YN,SX,XN,XN1,XN2,XN3,XN4,XN5,XN6,XN7: STD_LOGIC_VECTOR(15 DOWNTO 0);
30   SIGNAL XN8,XN9,XN10,XN11,XN12,XN13,XN14,XN15: STD_LOGIC_VECTOR(15 DOWNTO 0);
31   SIGNAL XN16,XN17,XN18,XN19,XN20,XN21,XN22,XN23: STD_LOGIC_VECTOR(15 DOWNTO 0);
32   SIGNAL XN24,XN25,XN26,XN27,XN28,XN29,XN30,XN31: STD_LOGIC_VECTOR(15 DOWNTO 0);
33
34   BEGIN
35   PROCESS(CLK, RST)
36   BEGIN
37   IF RST='0' THEN ST <= A;
38            YN <= (OTHERS =>'0');  SX <= (OTHERS =>'0');  XN <= (OTHERS =>'0');
39            XN1 <= (OTHERS =>'0'); XN2 <= (OTHERS =>'0'); XN3 <= (OTHERS =>'0');
40            XN4 <= (OTHERS =>'0'); XN5 <= (OTHERS =>'0'); XN6 <= (OTHERS =>'0');
41            XN7 <= (OTHERS =>'0'); XN8 <= (OTHERS =>'0'); XN9 <= (OTHERS =>'0');
42            XN10 <= (OTHERS =>'0');XN11 <= (OTHERS =>'0');XN12 <= (OTHERS =>'0');
43            XN13 <= (OTHERS =>'0');XN14 <= (OTHERS =>'0');XN15 <= (OTHERS =>'0');
44            XN16 <= (OTHERS =>'0');XN17 <= (OTHERS =>'0');XN18 <= (OTHERS =>'0');
45            XN19 <= (OTHERS =>'0');XN20 <= (OTHERS =>'0');XN21 <= (OTHERS =>'0');
46            XN22 <= (OTHERS =>'0');XN23 <= (OTHERS =>'0');XN24 <= (OTHERS =>'0');
47            XN25 <= (OTHERS =>'0');XN26 <= (OTHERS =>'0');XN27 <= (OTHERS =>'0');
48            XN28 <= (OTHERS =>'0');XN29 <= (OTHERS =>'0');XN30 <= (OTHERS =>'0');
49            XN31 <= (OTHERS =>'0');
50
51   ELSIF (CLK'EVENT AND CLK='1') THEN
52    CASE ST is
53      WHEN A => IF DATA_READY='0' THEN ST<=A; --espera Data_Ready
54                ELSE ST<=B;
55                END IF;
56
57      WHEN B => XN<=X;                    -- leitura do valor Xn
58                ST<= C;
59
60      WHEN C => SX<= XN +XN1 +XN2 +XN3 +XN4 +XN5 +XN6 +XN7 +XN8 +XN9 +XN10
61                     +XN11 +XN12 +XN13 +XN14 +XN15 +XN16 +XN17 +XN18 +XN19
62                     +XN20 +XN21 +XN22 +XN23 +XN24 +XN25 +XN26 +XN27 +XN28
63                     +XN29 +XN30 +XN31;       --soma valores de X
64                ST<=D;
65
66      WHEN D => IF SX(15)='1' THEN YN<= "11111" & SX(15 DOWNTO 5);--média negativa
67                ELSE YN <= "00000" & SX(15 DOWNTO 5); -- calcula média positiva
68                END IF;
69                ST<=E;
70
71      WHEN E => XN1 <= XN; XN2 <= XN1; XN3 <= XN2; -- atualiza valores de X
72                XN4 <= XN3; XN5 <= XN4; XN6 <= XN5; XN7 <= XN6;
73                XN8 <= XN7; XN9 <= XN8; XN10 <= XN9; XN11 <= XN10;
74                XN12 <= XN11; XN13 <= XN12; XN14 <= XN13; XN15 <= XN14;
75                XN16 <= XN15; XN17 <= XN16; XN18 <= XN17; XN19 <= XN18;
76                XN20 <= XN19; XN21 <= XN20; XN22 <= XN21; XN23 <= XN22;
77                XN24 <= XN23; XN25 <= XN24; XN26 <= XN25; XN27 <= XN26;
78                XN28 <= XN27; XN29 <= XN28; XN30 <= XN29; XN31 <= XN30;
79                ST<=A;
80
81      WHEN OTHERS => NULL;
82    END CASE;
83   END IF;
84   END PROCESS;
85
86   -- ATUALIZA SAÍDA:
87
88   Y <= YN;
89
90   END FSM;
```

Fonte: Autor.

Para apresentar o valor da aceleração, no formato decimal, nos displays de sete segmentos é necessário converter o valor binário em um valor decimal codificado em binário (BCD). Essa função é realizada pelo componente conversor de código binário para BCD (**CONVERTE_BIN_BCD.vhd**). Esse componente recebe um vetor de entrada (B[7..0]) proveniente dos últimos 8 bits da saída do filtro de média móvel (Y[7..0]), pois o valor decimal máximo desse vetor deve ser de 64 para a faixa de medida da aceleração de ±8.g. Além disso, esse componente recebe também na entrada "S" o valor do décimo bit da saída do filtro (Y[9]) que representa o sinal do número binário.

A conversão para BCD ocorre em três etapas: inicialmente o vetor binário de entrada tem o seu sinal verificado para a obtenção do módulo do valor binário (BM[7..0]). Caso S=1 o valor é negativo e os oito bits tem seu valor invertido (complemento do número negativo). Em seguida o valor de entrada é multiplicado 1,4 (resultado da multiplicação por 7, somado com um offset de 4, e divisão por 5) para que o valor 64 possa corresponder a 90 graus.

Finalmente o valor binário é convertido para BCD utilizando o algoritmo "Desloca e Soma 3" (*double dabble algorithm*). Esse algoritmo realiza diversas iterações. Em cada iteração, qualquer dígito BCD (unidade, dezena, centena) que seja pelo maior ou igual a cinco é incrementado de três e, em seguida, todos os bits são deslocados de uma posição à esquerda. O incremento garante que um valor de um dígito BCD igual a cinco, incrementado e deslocado para a esquerda, se torne dezesseis, gerando assim um "vai um" para o próximo dígito BCD. Logo, se algum dígito for cinco ou mais, o valor três é adicionado para garantir que esse dígito gere o "vai um" equivalente na base dez (uma vez que o valor cinco dobrado deve gerar o "vai um" para o próximo dígito decimal).

Nesse algoritmo possui as seguintes etapas (considerado que o valor binário a ser convertido está nos últimos oito bits de um vetor de 18 bits Z[17..0]):

1 – Deslocar o vetor de 18 bits uma posição à esquerda;

2 – Verificar se qualquer coluna que representa a dezena e a unidade do número BCD (Z[15..12] ou Z[11..8]) é maior que 4 (0100);

3 – Nas colunas em que o valor for maior que 4 deve-se adicionar 3;

4 – Voltar ao passo 1 até que oito deslocamentos tenham sido realizados;

5 – Depois de oito deslocamentos os nibbles mais significativos do vetor (Z[17..16], Z[15..12] e Z[11..8]) devem conter respectivamente os valores da centena, dezena e unidade do valor binário original em BCD.

O arquivo com o código VHDL do módulo conversor de código binário para BCD é mostrado na Figura 24.

O componente decodificador para display de sete segmentos (**DEC_HEX_DP.vhd**) é utilizado para representar o valor BCD de cada dígito (Dezena e Unidade) correspondente ao valor da aceleração de cada eixo. Esse módulo recebe como entrada, além dos quatro bits do código BCD, um bit de sinal (DP) que é utilizado para acender o ponto decimal do display de sete segmentos para indicar que o número é negativo. Dessa forma, esse decodificador tem oito bits de saída, onde o bit mais significativo deve ser conectado ao ponto decimal do display de

sete segmentos. O arquivo com o código VHDL do módulo decodificador para display de sete segmentos é mostrado na Figura 25.

Figura 24: Código VHDL do arquivo **CONVERTE_BIN_BCD.vhd**.

```
14    LIBRARY IEEE;
15    USE IEEE.STD_LOGIC_1164.ALL;
16    USE IEEE.STD_LOGIC_ARITH.ALL;
17    USE IEEE.STD_LOGIC_UNSIGNED.ALL;
18    USE IEEE.NUMERIC_STD.ALL; -- necessário p/ conversão de inteiro para std_logic
19
20    ENTITY CONVERTE_BIN_BCD IS     -- declaracao da entidade CONVERTE_BIN_BCD
21       PORT
22       (  CLK: IN STD_LOGIC;       -- sinais de controle
23          S  : IN STD_LOGIC;       -- bit de sinal
24          B  : IN STD_LOGIC_VECTOR(7 DOWNTO 0); -- vetor de 8 bits de entrada
25          BCD2,BCD1,BCD0 : OUT STD_LOGIC_VECTOR(3 DOWNTO 0);-- dígitos BCD de saida
26          DP : OUT STD_LOGIC);     -- ponto decimal
27    END CONVERTE_BIN_BCD;
28
29    ARCHITECTURE CoNVERSOR OF CONVERTE_BIN_BCD IS
30    SIGNAL BM: STD_LOGIC_VECTOR(7 DOWNTO 0);  -- valor binário em módulo
31    SIGNAL BI,AUX,VG: INTEGER range 0 to 511; -- valores auxiliares inteiros
32    SIGNAL BG: STD_LOGIC_VECTOR(7 DOWNTO 0);  -- valor convertdo em graus (binário)
33
34    BEGIN
35    -- Ajusta Fator de Conversão para medida em graus (-90 a +90 graus)
36    BM <= NOT(B) WHEN S='1' ELSE B; -- inverte números negativos (valor em módulo)
37    DP<= S;                   -- mostra valor negativo em ponto decimal do  display
38    BI <= CONV_INTEGER(BM); -- converte valor do módulo para valor inteiro
39    AUX <= (BI*7+4);          -- ajusta fator de escala para graus: 64 -> 90 graus
40    VG <= AUX/5;              -- com a expressão: (64*7+4)/5 = 90,4 => 90 graus
41    BG <= STD_LOGIC_VECTOR(TO_UNSIGNED(VG,8)); -- converte inteiro para binário
42
43    -- Algoritmo: "Desloca e Soma 3" (converte Binário (8 bits) -> 3 dígitos BCD):
44    -- 1 - Se qualquer coluna (Centena, Dezena, Unidade) for maior que 4 (0100),
45    --     adicionar  3  a  essa  coluna;
46    -- 2 - Deslocar todos os bits uma posição à esquerda;
47    -- 3 - Avaliar cada coluna BCD;
48    -- 4 - Voltar ao passo 1 até que oito deslocamentos tenham sido realizados.
49    -- 5 - Depois de oito deslocamentos os 3 nibbles mais significativos devem
50    --     conter os valores dos três dígitos em BCD.
51
52    PROCESS (CLK)
53    VARIABLE Z: STD_LOGIC_VECTOR(17 DOWNTO 0); -- variável de conversão
54       -- Converte binário de 8 bits para BCD {|F|F|(hex) => |2|5|5|(bcd) }
55    BEGIN
56       IF CLK'EVENT AND CLK='1' THEN
57          Z := (OTHERS=>'0');     -- zera vetor auxiliar  z
58          Z(10 DOWNTO 3) := BG;   -- equivale a deslocar à esquerda 3 vezes
59       -- Converte binário de 8 bits para BCD {|F|F|(hex) => |2|5|5|(bcd) }
60          FOR I IN 0 TO 4 LOOP  -- desloca 5 vezes somando 3 se coluna BCD > 4
61             IF Z(11 DOWNTO 8) > 4 THEN          -- verifica coluna 0 BCD
62                Z(11 DOWNTO 8) := Z(11 DOWNTO 8) + 3;
63             END IF;
64             IF Z(15 DOWNTO 12) > 4 THEN         -- verifica coluna 1 BCD
65                Z(15 DOWNTO 12) := Z(15 DOWNTO 12) + 3;
66             END IF;
67             Z(17 DOWNTO 1) := Z(16 DOWNTO 0);   -- desloca 1 bit à esquerda
68          END LOOP;
69          BCD0 <= Z(11 DOWNTO 8);                -- dígito BCD da coluna 0
70          BCD1 <= Z(15 DOWNTO 12);               -- dígito BCD da coluna 1
71          BCD2 <= "00" & Z(17 DOWNTO 16);        -- dígito BCD da coluna 2
72       END IF;
73    END PROCESS;
74
75    END CONVERSOR;
```

Fonte: Autor.

Figura 25: Código VHDL do arquivo **DEC_HEX_DP.vhd**.

```
14    LIBRARY IEEE;
15    USE IEEE.STD_LOGIC_1164.ALL;
16
17    ENTITY DEC_HEX_DP IS                    -- declaracao da entidade DEC_HEX_DP
18       PORT
19       (  DP : IN STD_LOGIC;                -- bit de ponto decimal
20          Q  : IN STD_LOGIC_VECTOR(3 DOWNTO 0);  -- vetor de entrada Q[3] Q[2] Q[1] Q[0]
21          D  : OUT STD_LOGIC_VECTOR(7 DOWNTO 0) ); -- vetor de saida D(7..0)
22    END DEC_HEX_DP;
23
24    ARCHITECTURE SELETOR OF DEC_HEX_DP IS
25
26    BEGIN
27
28    D(7)<= NOT(DP);
29
30    WITH Q SELECT
31    D(6 DOWNTO 0)<="1000000" WHEN "0000",   -- display 0
32                   "1111001" WHEN "0001",   -- display 1
33                   "0100100" WHEN "0010",   -- display 2
34                   "0110000" WHEN "0011",   -- display 3
35                   "0011001" WHEN "0100",   -- display 4
36                   "0010010" WHEN "0101",   -- display 5
37                   "0000010" WHEN "0110",   -- display 6
38                   "1111000" WHEN "0111",   -- display 7
39                   "0000000" WHEN "1000",   -- display 8
40                   "0010000" WHEN "1001",   -- display 9
41                   "0001000" WHEN "1010",   -- display A
42                   "0000011" WHEN "1011",   -- display b
43                   "1000110" WHEN "1100",   -- display C
44                   "0100001" WHEN "1101",   -- display d
45                   "0000110" WHEN "1110",   -- display E
46                   "0001110" WHEN "1111",   -- display F
47                   "1111111" WHEN OTHERS;   -- display Apagado
48    END SELETOR;
```

Fonte: Autor.

O componente decodificador para bargraph de Led's (**LED_BARGRAPH.vhd**) é utilizado para representar o valor da rotação no eixo X no formato de um gráfico de barras (Bargraph) composto pelos dez Led's da placa DE10-Lite. Esse módulo recebe como entrada os oito bits menos significativos do valor da aceleração no eixo X (que varia entre -64 e +64) e soma 64, gerando um deslocamento da escala (*offset*) para a faixa de 0 a 128. Essa faixa é dividida em dez intervalos, sendo que em cada intervalo um conjunto de Led's acende progressivamente (como representado na Tabela 8). Dessa forma, o número de Led's acesos é proporcional ao ângulo de rotação da placa no eixo X. O arquivo com o código VHDL do módulo para bargraph de Led's é mostrado na Figura 26.

O arquivo principal do medidor de inclinação, denominado de **DISPLAY_XYZ.vhd**, é descrito no formato estrutural com os componentes: módulo acelerômetro (**ACELEROMETRO.vhd**), filtro de média móvel de ordem 32 (**FILTRO_MM32.vhd**), conversor de código binário para BCD (**CONVERTE_BIN_BCD.vhd**), decodificador para display de sete segmentos (**DEC_HEX_DP.vhd**) e decodificador para bargraph (**LED_BARGRAPH.vhd**). O arquivo VHDL com essa descrição é apresentado na Figura 27.

Tabela 8: Relação entre a inclinação sensor no eixo X e os valores do sensor com offset de 64.

ÂNGULO (º)	Valores de aceleração no eixo X (D[9..0]) somados com 64										Valor Decimal	Valor Bargraph LED[9..0]
	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0	128	
+90	0	0	1	0	0	0	0	0	0	0	128	1111111111
+72	0	0	0	1	1	1	0	0	1	1	115 (a 127)	1111111111
+54	0	0	0	1	1	0	0	1	1	0	102 (a 114)	0111111111
+36	0	0	0	1	0	1	1	0	1	0	90 (a 101)	0011111111
+18	0	0	0	1	0	0	1	1	0	0	76 (a 89)	0001111111
0	0	0	0	1	0	0	0	0	0	0	64 (a 75)	0000111111
-18	0	0	0	0	1	1	0	0	1	1	51 (a 63)	0000011111
-36	0	0	0	0	1	0	0	1	1	0	38 (a 50)	0000001111
-54	0	0	0	0	0	1	1	0	0	1	25 (a 37)	0000000111
-72	0	0	0	0	0	0	1	1	0	1	13 (a 24)	0000000011
-90	0	0	0	0	0	0	0	0	0	0	0 (a 12)	0000000001

Fonte: Autor.

Figura 26: Código VHDL do arquivo **LED_BARGRAPH.vhd**.

```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
USE IEEE.STD_LOGIC_ARITH.ALL;
USE IEEE.STD_LOGIC_UNSIGNED.ALL;

ENTITY LED_BARGRAPH IS
   PORT(
      VALOR: IN STD_LOGIC_VECTOR(9 DOWNTO 0);
      LED : OUT STD_LOGIC_VECTOR(9 DOWNTO 0) );
END LED_BARGRAPH;

ARCHITECTURE SELETOR OF LED_BARGRAPH IS
SIGNAL MODULO: STD_LOGIC_VECTOR(7 DOWNTO 0);  -- valor binário em módulo

BEGIN
-- Ajusta módulo do valor (-64 a +64) para (0 a 128)
MODULO <= VALOR(7 DOWNTO 0) + "01000000"; -- soma  64 ao valor

WITH MODULO SELECT
LED <= "1111111111" WHEN "11000000" DOWNTO "01110100", -- > 115
       "0111111111" WHEN "01110011" DOWNTO "01100110", -- 115 a 102
       "0011111111" WHEN "01100101" DOWNTO "01011010", -- 101 a 90
       "0001111111" WHEN "01011001" DOWNTO "01001100", -- 89 a 76
       "0000111111" WHEN "01001011" DOWNTO "01000000", -- 75 a 64
       "0000011111" WHEN "00111111" DOWNTO "00110011", -- 63 a 51
       "0000001111" WHEN "00110010" DOWNTO "00100110", -- 50 a 38
       "0000000111" WHEN "00100101" DOWNTO "00011001", -- 37 a 25
       "0000000011" WHEN "00011000" DOWNTO "00001101", -- 24 a 13
       "0000000001" WHEN OTHERS;                       -- < 13
END SELETOR;
```

Fonte: Autor.

Figura 27: Arquivo VHDL medidor de inclinação (**DISPLAY_XYZ.vhd**).

```
17  LIBRARY IEEE;
18  USE IEEE.STD_LOGIC_1164.ALL;
19
20  ENTITY DISPLAY_XYZ IS
21     PORT(
22        CLK, RST, SDO: IN STD_LOGIC;
23        R: IN STD_LOGIC_VECTOR(1 DOWNTO 0);
24        DX1,DX0,DY1,DY0, DZ1, DZ0 : OUT STD_LOGIC_VECTOR(7 DOWNTO 0);
25        SCLK, SDI, NOT_CS : OUT STD_LOGIC;
26        LED : OUT STD_LOGIC_VECTOR(9 DOWNTO 0)
27        );
28  END DISPLAY_XYZ;
29
30  ARCHITECTURE ESTRUTURAL OF  DISPLAY_XYZ IS
31  SIGNAL NX0,NX1,NY0,NY1, NZ0, NZ1: STD_LOGIC_VECTOR(3 DOWNTO 0);
32  SIGNAL SX,SY,SZ, DR: STD_LOGIC;
33  SIGNAL ACC_X, ACC_Y, ACC_Z: STD_LOGIC_VECTOR(15 DOWNTO 0);
34  SIGNAL EIXO_X, EIXO_Y, EIXO_Z: STD_LOGIC_VECTOR(9 DOWNTO 0);
35
36  COMPONENT ACELEROMETRO IS PORT(
37        CLK, RST, SDO: IN STD_LOGIC;
38        R: IN STD_LOGIC_VECTOR(1 DOWNTO 0);
39        DATA_READY: OUT STD_LOGIC;
40        ACC_X, ACC_Y, ACC_Z: OUT STD_LOGIC_VECTOR(15 DOWNTO 0);
41        SCLK, SDI, NOT_CS: OUT STD_LOGIC );
42  END COMPONENT ACELEROMETRO ;
43
44  COMPONENT FILTRO_MM32 IS PORT (
45        CLK, RST : IN STD_LOGIC;
46        DATA_READY : IN STD_LOGIC;
47        X : IN STD_LOGIC_VECTOR(15 DOWNTO 0);
48        Y : OUT STD_LOGIC_VECTOR(15 DOWNTO 0) );
49  END COMPONENT FILTRO_MM32;
50
51  COMPONENT CONVERTE_BIN_BCD IS PORT (
52        CLK: IN STD_LOGIC;
53        S  : IN STD_LOGIC;
54        B  : IN STD_LOGIC_VECTOR(7 DOWNTO 0);
55        BCD2, BCD1, BCD0  : OUT STD_LOGIC_VECTOR(3 DOWNTO 0);
56        DP : OUT STD_LOGIC );
57  END COMPONENT CONVERTE_BIN_BCD;
58
59  COMPONENT DEC_HEX_DP  IS PORT (
60        DP : IN STD_LOGIC;
61        Q  : IN STD_LOGIC_VECTOR(3 DOWNTO 0);
62        D  : OUT STD_LOGIC_VECTOR(7 DOWNTO 0) );
63  END COMPONENT DEC_HEX_DP;
64
65  COMPONENT LED_BARGRAPH  IS PORT (
66        VALOR : IN STD_LOGIC_VECTOR(9 DOWNTO 0);
67        LED  : OUT STD_LOGIC_VECTOR(9 DOWNTO 0) );
68  END COMPONENT LED_BARGRAPH;
69
70  BEGIN
71     SENSOR : ACELEROMETRO PORT MAP(CLK=>CLK,RST=>RST,SDO=>SDO,R=>R,
72              DATA_READY=>DR, SCLK=>SCLK, SDI=>SDI,
73              NOT_CS=>NOT_CS, ACC_X=>ACC_X,ACC_Y=>ACC_Y,ACC_Z=>ACC_Z);
74     FILTRO_X: FILTRO_MM32 PORT MAP(CLK=>CLK,RST=>RST,DATA_READY=>DR,
75              X=>ACC_X, Y(9 DOWNTO 0)=>EIXO_X);
76     FILTRO_Y: FILTRO_MM32 PORT MAP(CLK=>CLK,RST=>RST,DATA_READY=>DR,
77              X=>ACC_Y, Y(9 DOWNTO 0)=>EIXO_Y);
78     FILTRO_Z: FILTRO_MM32 PORT MAP(CLK=>CLK,RST=>RST,DATA_READY=>DR,
79              X=>ACC_Z, Y(9 DOWNTO 0)=>EIXO_Z);
80     CONVERTE_X: CONVERTE_BIN_BCD PORT MAP(CLK=>CLK, S=>EIXO_X(9),
81              B=>EIXO X(7 DOWNTO 0). BCD1=>NX1. BCD0=>NX0. DP=>SX):
82     CONVERTE_Y: CONVERTE_BIN_BCD PORT MAP(CLK=>CLK, S=>EIXO_Y(9),
83              B=>EIXO_Y(7 DOWNTO 0), BCD1=>NY1, BCD0=>NY0, DP=>SY);
84     CONVERTE_Z: CONVERTE_BIN_BCD PORT MAP(CLK=>CLK, S=>EIXO_Z(9),
85              B=>EIXO_Z(7 DOWNTO 0), BCD1=>NZ1, BCD0=>NZ0, DP=>SZ);
86     DEC_X0 : DEC_HEX_DP PORT MAP(DP=> SX, Q=>NX0, D=>DX0);
87     DEC_X1 : DEC_HEX_DP PORT MAP(DP=> SX, Q=>NX1, D=>DX1);
88     DEC_Y0 : DEC_HEX_DP PORT MAP(DP=> SY, Q=>NY0, D=>DY0);
89     DEC_Y1 : DEC_HEX_DP PORT MAP(DP=> SY, Q=>NY1, D=>DY1);
90     DEC_Z0 : DEC_HEX_DP PORT MAP(DP=> SZ, Q=>NZ0, D=>DZ0);
91     DEC_Z1 : DEC_HEX_DP PORT MAP(DP=> SZ, Q=>NZ1, D=>DZ1);
92     LEDS   : LED_BARGRAPH PORT MAP(VALOR => EIXO_X, LED=>LED);
93
94  END ESTRUTURAL;
```

Fonte: Autor.

Para teste do circuito os sinais de entrada e saída foram associados a pinos da placa DE10-Lite, conforme a Tabela 9.

Tabela 9: Associação de pinos do medidor de inclinação.

Node Name, Direction	Location	Node Name, Direction	Location
CLK, Input	PIN_P11	DZ0[7], Output	PIN_F17
DX0[7], Output	PIN_D15	DZ0[6], Output	PIN_F20
DX0[6], Output	PIN_C17	DZ0[5], Output	PIN_F19
DX0[5], Output	PIN_D17	DZ0[4], Output	PIN_H19
DX0[4], Output	PIN_E16	DZ0[3], Output	PIN_J18
DX0[3], Output	PIN_C16	DZ0[2], Output	PIN_E19
DX0[2], Output	PIN_C15	DZ0[1], Output	PIN_E20
DX0[1], Output	PIN_E15	DZ0[0], Output	PIN_F18
DX0[0], Output	PIN_C14	DZ1[7], Output	PIN_L19
DX1[7], Output	PIN_A16	DZ1[6], Output	PIN_N20
DX1[6], Output	PIN_B17	DZ1[5], Output	PIN_N19
DX1[5], Output	PIN_A18	DZ1[4], Output	PIN_M20
DX1[4], Output	PIN_A17	DZ1[3], Output	PIN_N18
DX1[3], Output	PIN_B16	DZ1[2], Output	PIN_L18
DX1[2], Output	PIN_E18	DZ1[1], Output	PIN_K20
DX1[1], Output	PIN_D18	DZ1[0], Output	PIN_J20
DX1[0], Output	PIN_C18	LED[9], Output	PIN_B11
DY0[7], Output	PIN_A19	LED[8], Output	PIN_A11
DY0[6], Output	PIN_B22	LED[7], Output	PIN_D14
DY0[5], Output	PIN_C22	LED[6], Output	PIN_E14
DY0[4], Output	PIN_B21	LED[5], Output	PIN_C13
DY0[3], Output	PIN_A21	LED[4], Output	PIN_D13
DY0[2], Output	PIN_B19	LED[3], Output	PIN_B10
DY0[1], Output	PIN_A20	LED[2], Output	PIN_A10
DY0[0], Output	PIN_B20	LED[1], Output	PIN_A9
DY1[7], Output	PIN_D22	LED[0], Output	PIN_A8
DY1[6], Output	PIN_E17	NOT_CS, Output	PIN_AB16
DY1[5], Output	PIN_D19	R[1], Input	PIN_C11
DY1[4], Output	PIN_C20	R[0], Input	PIN_C10
DY1[3], Output	PIN_C19	RST, Input	PIN_B8
DY1[2], Output	PIN_E21	SCLK, Output	PIN_AB15
DY1[1], Output	PIN_E22	SDI, Output	PIN_V11
DY1[0], Output	PIN_F21	SDO, Input	PIN_V12

Fonte: TERASIC, 2018.

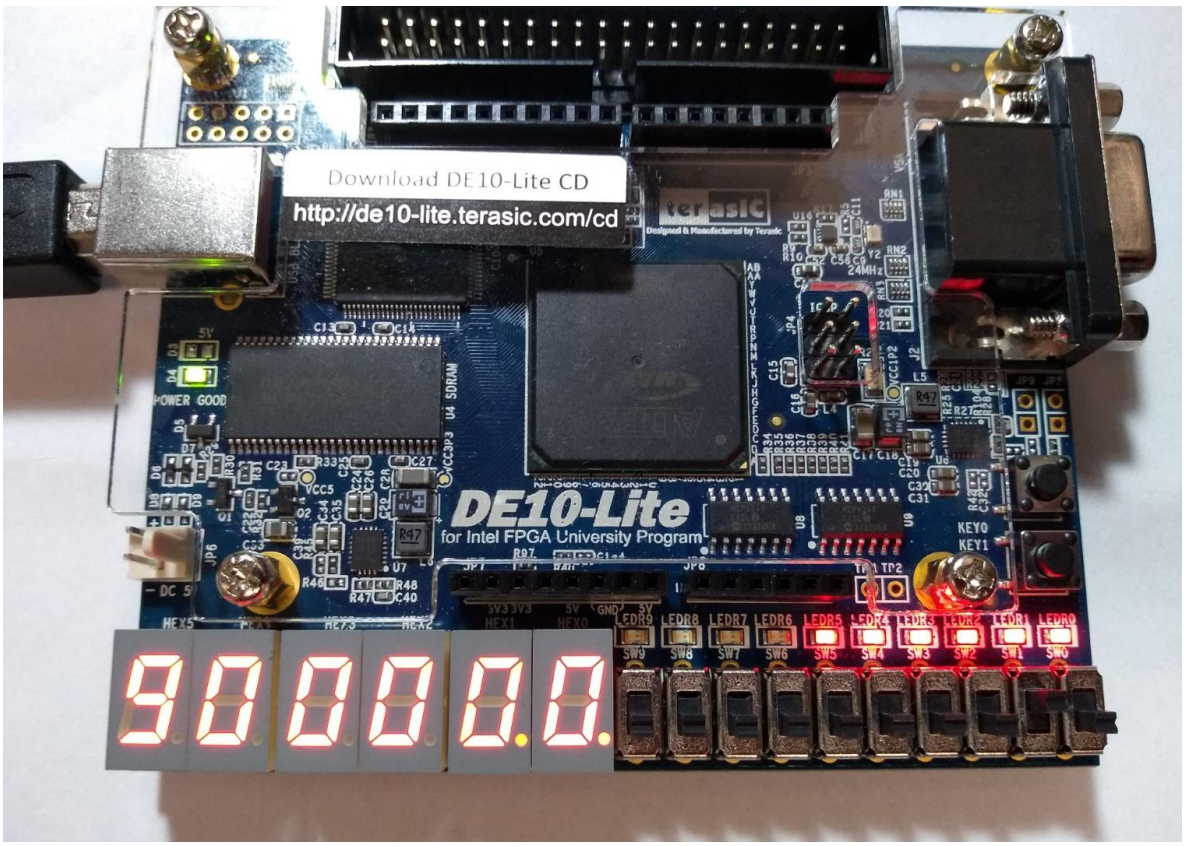
Os sinais de interface entre o acelerômetro digital e o FPGA (SDO, SDI, NOT_CS e SCLK) são ligados aos pinos de conexão correspondentes aos sinais de mesmo nome do

ADXL345. O sinal CLK é ligado ao clock de 50 MHz da placa DE10-Lite. O sinal RST é ligado a uma chave de contato momentâneos (KEY0). Os sinais de seleção da faixa de operação do sensor (sensibilidade) R(1) e R(0) estão conectados a duas chaves deslizantes (SW1 e SW0). Por fim as saídas para os seis displays de sete segmentos (HEX5 a HEX0) e ao conjunto de dez Led's estão associados aos respectivos pinos, conforme o Manual de Usuário da Placa DE10-Lite (TERASIC, 2018).

Observação: No Quartus Prime a atribuição de pinos realizada em um projeto (utilizando a opção *Assignments> Pin Planner*) pode ser transferida para outra pasta de projeto (desde que o arquivo do projeto tenha o mesmo nome) utilizando as opções: *Assignments> Export Assignments* e *Assignments> Import Assignments.* Os arquivos contendo a configuração de pinos possuem a extensão: "**.qsf**" (por exemplo: **DISPLAY_XYZ.qsf**) e podem ser encontrados na principal de cada projeto.

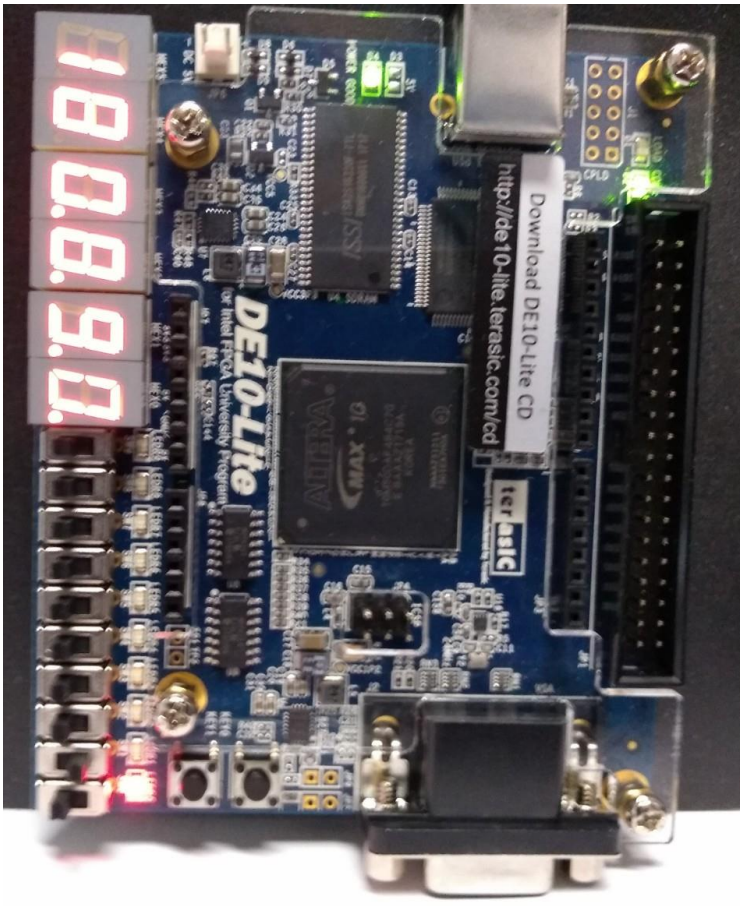
Após a configuração do FPGA foi possível observar o acionamento dos Leds e displays em função da inclinação da placa DE10-Lite como definido para o projeto, conforme Tabela 9. A Figura 28 mostra os displays HEX1 e HEX0 com valor 0 e os Led's 0 a 5 acesos, o que representa 0 graus de inclinação (horizontal). Na Figura 29 os displays HEX1 e HEX0 mostram o valor -90 (com ponto decimal aceso) e o Led 0 aceso, o que representa 90 graus no sentido horário (vertical).

Figura 28: Medidor de inclinação - Plataforma DE10-Lite na posição horizontal.



Fonte: Autor.

Figura 29: Medidor de Inclinação - Plataforma DE10-Lite na posição vertical.



Fonte: Autor.

5.2 Projeto 2: Medidor de Aceleração da Gravidade

Outro exemplo de projeto que mostra a aplicação do módulo do acelerômetro é o medidor de aceleração da gravidade cuja função é fornecer a medida da aceleração estática da gravidade que atua sobre o ADXL345. Para transformar o valor binário extraído do acelerômetro para um valor de aceleração gravitacional é preciso transformar o valor binário em decimal e multiplicar pelo fator de escala. Considerando-se a faixa de medida de maior resolução (*range*) de ±2.g a sensibilidade do sensor pode variar entre 3,5.mg/LSB a 4,3.mg/LSB, com valor típico de 3,9.mg/LSB, conforme indicado na Tabela 10. Como a aceleração estática da gravidade equivale a 1.g, para o desenvolvimento do sistema é utilizada a faixa valores de ±2.g, que é faixa de maior resolução. O valor de aceleração gravitacional é mostrado nos displays de 7 segmentos e um conjunto de dez Led's é utilizado para mostrar a contagem das amostras realizadas. Esses Led's e displays estão disponíveis na plataforma DE10-Lite (ANALOG DEVICES, 2015), (TERASIC, 2018).

Tabela 10: Tabela de sensibilidade dos dados de saída em função do range de aceleração.

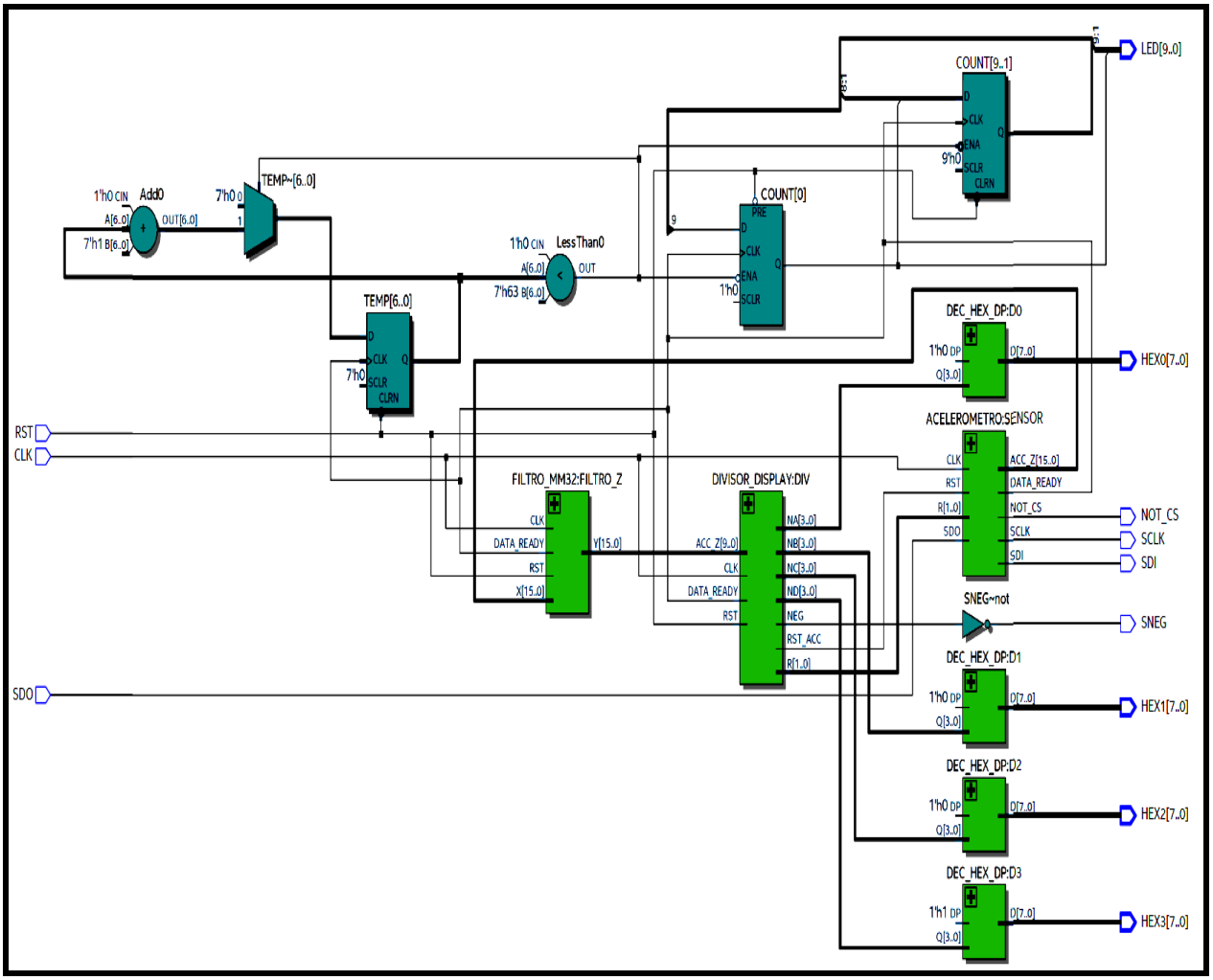
SENSITIVITY	Each axis				
Sensitivity at X_{OUT}, Y_{OUT}, Z_{OUT}	All *g*-ranges, full resolution	230	256	282	LSB/*g*
	±2 *g*, 10-bit resolution	230	256	282	LSB/*g*
	±4 *g*, 10-bit resolution	115	128	141	LSB/*g*
	±8 *g*, 10-bit resolution	57	64	71	LSB/*g*
	±16 *g*, 10-bit resolution	29	32	35	LSB/*g*
Sensitivity Deviation from Ideal	All *g*-ranges		±1.0		%
Scale Factor at X_{OUT}, Y_{OUT}, Z_{OUT}	All *g*-ranges, full resolution	3.5	3.9	4.3	mg/LSB
	±2 *g*, 10-bit resolution	3.5	3.9	4.3	mg/LSB
	±4 *g*, 10-bit resolution	7.1	7.8	8.7	mg/LSB
	±8 *g*, 10-bit resolution	14.1	15.6	17.5	mg/LSB
	±16 *g*, 10-bit resolution	28.6	31.2	34.5	mg/LSB
Sensitivity Change Due to Temperature			±0.01		%/°C

Fonte: ANALOG DEVICES, 2015.

No desenvolvimento do projeto foi observado que os 2 bits menos significativos do byte menos significativo dos dados de aceleração do eixo z estavam muito instáveis, o que dificultava a leitura do valor da aceleração gravitacional nos displays e por isso esses bits foram desprezados, reduzindo a sensibilidade do sensor. O sensor digital ADXL345 mede a aceleração da gravidade no seu eixo Z (equivale a 1.g ou 9,806 m/s^2 ao nível do mar). Foi adotada a faixa de medida de ±2.g isso significa que quando estiver sendo medido 1.g o sensor deve indicar o valor 256 (em 9 bits) no eixo Z. Como foram desprezados os dois bits menos significativos da medida esse valor é representado como 64 (em 7 bits). Portanto, com a redução da sensibilidade, o fator de escala utilizado no projeto foi de 9.806/64 = 153/LSB, onde o LSB considerado é o terceiro bit do valor dos dados (ANALOG DEVICES, 2015).

O código VHDL do medidor de aceleração da gravidade (**DISPLAY_GRAVIDADE.vhd**) é constituído de quatro tipos de componentes: o módulo acelerômetro (**ACELEROMETRO.vhd**), o filtro de média móvel de ordem 32 (**FILTRO_MM32.vhd**), um divisor do código binário em quatro displays (**DIVISOR_DISPLAY.vhd**), um decodificador para display de sete segmentos (**DEC_HEX_DP.vhd**). A visão RTL da interligação desses componentes é mostrada na Figura 30.

Figura 30: Visão RTL do projeto **DISPLAY_GRAVIDADE**.



Fonte: Autor.

O módulo acelerômetro (**ACELEROMETRO.vhd**) comunica-se com o sensor ADXL345 através dos sinais de comunicação serial SPI (SDO, SDI, SCLCK, NOT_CS) e fornece os valores de leitura de aceleração em cada um dos eixos (ACC_X[15..0], ACC_Y[15..0], ACC_Z[15..0]). Conforme descrito na **seção 4.2**, esse módulo recebe o sinal de clock principal (CLK) de 50 MHz e está configurado para realizar a leitura dos dados de aceleração na taxa de 100 leituras/segundo. A faixa de medida da aceleração (*Range*) para o medidor de aceleração é ±2.g, logo os sinais de entrada R[1..0] são fixados em: R[1] = ’0’ e R[0]

= ’0’. A cada leitura de dados realizada o sinal DATA_READY é ativado por um período do clock de transmissão de dados (10 µs)

A medida de aceleração do eixo Z (ACC_Z[15..0]) é enviada a um componente que realiza um filtro de média móvel de ordem 32 (**FILTRO_MM32.vhd**). A função desse componente é atenuar as flutuações (ruídos) dos sinais de aceleração lidos do sensor, sendo que para isso realiza a média dos últimos 32 valores lidos. Esse componente implementa a expressão:

$$Y_{(n)}=(x_{(n)} +x_{(n-1)} +x_{(n-2)} + ...+x_{(n-30)} +x_{(n-31)})/ 32$$

Esse componente já foi descrito na seção anterior (Figura 23).

O componente divisor do código binário em quatro displays (**DIVISOR_DISPLAY.vhd**) realiza o ajuste do fator de escala da medida e a conversão de um código binário de 15 bits em quatro valores BCD (NA, NB, NC, ND) através de uma máquina de estados. No estado **INICIO** o módulo acelerômetro é iniciado (RST_ACC=’0’) de modo a configurar o sensor para a faixa de operação de ±2.g. No estado **ESPERA_DADOS** a lógica espera a ativação do sinal DATA_READY. No estado DIVISOR_SUCESSIVO o valor (em módulo) do eixo Z do acelerômetro (ACC_M[9..0]) é limitado a sete bits e convertido para inteiro (ACC_Z_INT). Esse valor é multiplicado por 153 e novamente convertido para um sinal binário de 15 bits (GRAV_STD[14..0]. Finalmente o valor binário de 15 bits é convertido para quatro valores BCD utilizando o algoritmo “Desloca e Soma3”. Esse algoritmo já foi descrito na Figura 24). Os valores BCD resultam nos valores NA[3..0], NB[3..0], NC[3..0] e ND[3..0]. O bit 9 do valor da aceleração (ACC_Z(9)) é utilizado para gerar a sinalização de sinal negativo (NEG) que deve acender o segmento intermediário do display mais significativo. O código VHDL desse componente é apresentado na Figura 31.

O componente decodificador para display de sete segmentos (**DEC_HEX_DP.vhd**) é utilizado para representar o valor BCD de cada dígito (Dezena e Unidade) correspondente ao valor da aceleração de cada eixo. Esse módulo recebe como entrada, além dos quatro bits do código BCD, um bit de sinal (DP) que é utilizado para acender o ponto decimal do display de sete segmentos para indicar que o número é negativo. Dessa forma, esse decodificador tem oito bits de saída, onde o bit mais significativo deve ser conectado ao ponto decimal do display de sete segmentos. O arquivo com o código VHDL do módulo decodificador para display de sete segmentos é mostrado na Figura 25.

Figura 31: Código VHDL do arquivo **DIVISOR_DISPLAY.vhd**.

```
16  LIBRARY IEEE;
17  USE IEEE.STD_LOGIC_1164.ALL;
18  USE IEEE.STD_LOGIC_UNSIGNED.ALL;
19  USE IEEE.NUMERIC_STD.ALL;
20
21  ENTITY DIVISOR_DISPLAY IS
22     PORT (
23        CLK,RST,DATA_READY : IN STD_LOGIC;
24        ACC_Z: IN STD_LOGIC_VECTOR(9 DOWNTO 0);
25        RST_ACC, NEG: OUT STD_LOGIC;
26        R   : OUT STD_LOGIC_VECTOR(1 DOWNTO 0);
27        NA,NB,NC,ND : OUT STD_LOGIC_VECTOR(3 DOWNTO 0) );
28  END ENTITY;
29
30  ARCHITECTURE RTL OF DIVISOR_DISPLAY IS
31  SIGNAL GRAV_STD: STD_LOGIC_VECTOR(14 DOWNTO 0);
32  SIGNAL ACC_M: STD_LOGIC_VECTOR(9 DOWNTO 0);
33  SIGNAL GRAV: INTEGER RANGE 0 TO 32767;
34  SIGNAL ACC_Z_INT: INTEGER RANGE 0 TO 511;
35  TYPE ESTADO_ME IS (INICIO, ESPERA_DADOS,DIVISOR_SUCESSIVO);
36  SIGNAL ST: ESTADO_ME;
37
38  BEGIN
39
40  ACC_M <= not(ACC_Z) WHEN ACC_Z(9)='1' ELSE ACC_Z;
41
42  PROCESS(CLK,RST,DATA_READY)
43  VARIABLE Z: STD_LOGIC_VECTOR(34 DOWNTO 0); -- variável de conversão
44      BEGIN
45      IF RST='0' THEN NA<="0000";NB<="0000";NC<="0000";ND<="0000";
46                      R<="00";RST_ACC<='0';ST<=INICIO;
47      ELSIF RISING_EDGE(CLK) THEN
48        CASE ST IS
49           WHEN INICIO => NA<="0000";NB<="0000";NC<="0000";ND<="0000";
50                R<="00";RST_ACC<='0';ST<=ESPERA_DADOS;
51
52           WHEN ESPERA_DADOS =>
53                RST_ACC<='1';                -- ativa acelerômetro
54                IF DATA_READY='1' THEN   ST<= DIVISOR_SUCESSIVO;
55                ELSE ST<=ESPERA_DADOS;
56                END IF;
57
58           WHEN DIVISOR_SUCESSIVO => ST<=ESPERA_DADOS;
59                -- Ajuste do fator de escala:
60                ACC_Z_INT<=to_integer(UNSIGNED(ACC_M(8 DOWNTO 2)));
61                GRAV<=ACC_Z_INT*153;
62                GRAV_STD<=std_logic_vector(TO_UNSIGNED(GRAV,15));
63
64                -- Converte binário de 15 bits para BCD {|7|F|F|F|(hex) =>
    |3|2|7|6|7|(bcd) }
65                Z := (OTHERS=>'0');      -- zera vetor auxiliar  z
66                Z(17 DOWNTO 3) := GRAV_STD;   -- equivale a deslocar à esquerda 3 vezes
67                FOR i IN 0 TO 11 LOOP -- desloca 12 vezes somando 3 se coluna BCD > 4
68                IF Z(18 DOWNTO 15) > 4 THEN          -- verifica coluna 0 BCD
69                   Z(18 DOWNTO 15) := Z(18 DOWNTO 15) + 3;
70                END IF;
71                IF Z(22 DOWNTO 19) > 4 THEN          -- verifica coluna 1 BCD
72                   Z(22 DOWNTO 19) := Z(22 DOWNTO 19) + 3;
73                END IF;
74                IF Z(26 DOWNTO 23) > 4 THEN          -- verifica coluna 2 BCD
75                   Z(26 DOWNTO 23) := Z(26 DOWNTO 23) + 3;
76                END IF;
77                IF Z(30 DOWNTO 27) > 4 THEN          -- verifica coluna 3 BCD
78                   Z(30 DOWNTO 27) := Z(30 DOWNTO 27) + 3;
79                END IF;
80                IF Z(34 DOWNTO 31) > 4 THEN          -- verifica coluna 4 BCD
81                   Z(34 DOWNTO 31) := Z(34 DOWNTO 31) + 3;
82                END IF;
83                Z(34 DOWNTO 1) := Z(33 DOWNTO 0);   -- desloca 1 bit à esquerda
84                END LOOP;
85
86                -- Distribuição de valores nos displays:
87                NA <= Z(18 DOWNTO 15);                -- dígito BCD da coluna 0
88                NB <= Z(22 DOWNTO 19);                -- dígito BCD da coluna 1
89                NC <= Z(26 DOWNTO 23);                -- dígito BCD da coluna 2
90                ND <= Z(30 DOWNTO 27);                -- dígito BCD da coluna 3
91                NEG<=ACC_Z(9);
92
93           WHEN OTHERS => NULL;
94        END CASE;
95     END IF;
96  END PROCESS;
97
98  END RTL;
```

Fonte: Autor.

O arquivo principal do medidor de aceleração da gravidade, denominado de **DISPLAY_GRAVIDADE.vhd**, é descrito no formato estrutural com os componentes: módulo acelerômetro (**ACELEROMETRO.vhd**), filtro de média móvel de ordem 32 (**FILTRO_MM32.vhd**), divisor do código binário em quatro displays (**DIVISOR_DISPLAY.vhd**) e decodificador para display de sete segmentos (**DEC_HEX_DP.vhd**). A parte inicial do arquivo VHDL com essa descrição é apresentado na Figura 32.

Figura 32: Parte inicial do arquivo VHDL **DISPLAY_GRAVIDADE.vhd**.

```
17  LIBRARY IEEE;
18  USE IEEE.STD_LOGIC_1164.ALL;
19  USE IEEE.STD_LOGIC_ARITH.ALL;
20  USE IEEE.STD_LOGIC_UNSIGNED.ALL;
21
22  ENTITY DISPLAY_GRAVIDADE IS
23     PORT(
24        CLK, RST,SDO: IN STD_LOGIC;
25        HEX0,HEX1,HEX2,HEX3 : OUT STD_LOGIC_VECTOR(7 DOWNTO 0);
26        SCLK, SDI, NOT_CS, SNEG : OUT STD_LOGIC;
27        LED : OUT STD_LOGIC_VECTOR(9 DOWNTO 0)
28        );
29  END DISPLAY_GRAVIDADE;
30
31  ARCHITECTURE ESTRUTURAL OF DISPLAY_GRAVIDADE IS
32  SIGNAL NA,NB,NC,ND: STD_LOGIC_VECTOR(3 DOWNTO 0);
33  SIGNAL R: STD_LOGIC_VECTOR(1 DOWNTO 0);
34  SIGNAL RST_ACC,LOAD_DATA, DR, SINAL: STD_LOGIC;
35  SIGNAL ACC_Z: STD_LOGIC_VECTOR(15 DOWNTO 0);
36  SIGNAL Z: STD_LOGIC_VECTOR(9 DOWNTO 0);
37  SIGNAL COUNT: STD_LOGIC_VECTOR(9 DOWNTO 0);
38
39  COMPONENT ACELEROMETRO IS PORT(
40        CLK, RST, SDO: IN STD_LOGIC;
41        R: IN STD_LOGIC_VECTOR(1 DOWNTO 0);
42        DATA_READY: OUT STD_LOGIC;
43        ACC_X, ACC_Y, ACC_Z: OUT STD_LOGIC_VECTOR(15 DOWNTO 0);
44        SCLK, SDI, NOT_CS: OUT STD_LOGIC );
45  END COMPONENT ACELEROMETRO;
46
47  COMPONENT FILTRO_MM32 IS PORT (
48        CLK, RST : IN STD_LOGIC;
49        DATA_READY : IN STD_LOGIC;
50        X : IN STD_LOGIC_VECTOR(15 DOWNTO 0);
51        Y : OUT STD_LOGIC_VECTOR(15 DOWNTO 0) );
52  END COMPONENT FILTRO_MM32;
53
54  COMPONENT DIVISOR_DISPLAY IS PORT (
55        CLK,RST,DATA_READY : IN STD_LOGIC;
56        ACC_Z: IN STD_LOGIC_VECTOR(9 DOWNTO 0);
57        RST_ACC, NEG: OUT STD_LOGIC;
58        R  : OUT STD_LOGIC_VECTOR(1 DOWNTO 0);
59        NA,NB,NC,ND : OUT STD_LOGIC_VECTOR(3 DOWNTO 0) );
60  END COMPONENT DIVISOR_DISPLAY;
61
62  COMPONENT DEC_HEX_DP  IS PORT (
63        DP : IN STD_LOGIC;
64        Q  : IN STD_LOGIC_VECTOR(3 DOWNTO 0);
65        D  : OUT STD_LOGIC_VECTOR(7 DOWNTO 0) );
66  END COMPONENT DEC_HEX_DP;
67
68  BEGIN
69     SENSOR : ACELEROMETRO PORT MAP(CLK=>CLK,RST=>RST_ACC,SDO=>SDO,R=>R,
70               DATA_READY=>DR, SCLK=>SCLK,SDI=>SDI, NOT_CS=>NOT_CS,
71               ACC_Z=>ACC_Z);
72     FILTRO_Z: FILTRO_MM32 PORT MAP(CLK=>CLK,RST=>RST,DATA_READY=>DR,
73                X=>ACC_Z, Y(9 DOWNTO 0)=>Z);
74     DIV   : DIVISOR_DISPLAY PORT MAP(CLK=>CLK,RST=>RST,DATA_READY=>DR,
75              ACC_Z=>Z,RST_ACC=>RST_ACC,R=>R,
76              NA=>NA, NB=>NB, NC=>NC, ND=>ND, NEG=>SINAL);
77     D0    : DEC_HEX_DP PORT MAP(DP=>'0',Q=>NA,D=>HEX0);
78     D1    : DEC_HEX_DP PORT MAP(DP=>'0',Q=>NB,D=>HEX1);
79     D2    : DEC_HEX_DP PORT MAP(DP=>'0',Q=>NC,D=>HEX2);
80     D3    : DEC_HEX_DP PORT MAP(DP=>'1',Q=>ND,D=>HEX3);
```

Fonte: Autor.

A parte final do arquivo principal do medidor de aceleração da gravidade (Figura 33) descreve a atribuição do sinal negativo à saída SNEG, assim como mostra a implementação de um processo para mostrar através da ativação sequencial de dez Led's o número de amostras lidas (cada Led é ativado quando cem amostras foram lidas). A implementação é no formado de um contador em anel (utilizando um registrador de deslocamento). Essa visualização tem como objetivo mostrar que a frequência da aquisição das amostras (100 amostras/segundo).

Figura 33: Parte final do arquivo VHDL **DISPLAY_GRAVIDADE.vhd**.

```
82       -- Atualiza display de sinal:
83       SNEG <= NOT(SINAL);
84
85       PROCESS (DR, RST)
86       VARIABLE TEMP: STD_LOGIC_VECTOR(6 DOWNTO 0); -- temporizador de amostras
87       BEGIN
88          IF RST ='0' THEN COUNT <="0000000001"; TEMP:="0000000";
89          ELSIF ( RISING_EDGE(DR) AND DR='1') THEN
90             IF TEMP< "1100011" THEN TEMP:= TEMP+1; -- contador de 100 amostras
91             ELSE  TEMP:="0000000";
92                   -- Registrador de Deslocamento:
93                      COUNT(0) <= COUNT(9);
94                      COUNT(1) <= COUNT(0);
95                      COUNT(2) <= COUNT(1);
96                      COUNT(3) <= COUNT(2);
97                      COUNT(4) <= COUNT(3);
98                      COUNT(5) <= COUNT(4);
99                      COUNT(6) <= COUNT(5);
100                     COUNT(7) <= COUNT(6);
101                     COUNT(8) <= COUNT(7);
102                     COUNT(9) <= COUNT(8);
103            END IF;
104         END IF;
105      END PROCESS;
106
107      --Atualiza Led's de contagem de 100 amostras:
108      LED<=COUNT;
109
110   END ESTRUTURAL;
```

Fonte: Autor.

Para a implementação do circuito na placa DE10-Lite, foi realizada a associação de pinos do projeto, conforme a Tabela 11. Os sinais SDO, SDI, SCLK e NOT_CS foram ligados aos pinos de interface entre o ADXL345 e o MAX10. O sinal de entrada CLK foi conectado ao clock de 50 MHz da placa, enquanto o sinal RST foi ligado a um botão de contato momentâneo (KEY0). Os vetores de saída de 7 bits DA, DB, DC e DD foram associados aos pinos dos displays hexadecimais, assim como os bits NEG e POINT sendo estes relativos ao sinal de negativo da aceleração e ao ponto decimal, respectivamente, conforme o Manual de Usuário da Placa DE10-Lite (TERASIC, 2018).

Tabela 11: Associação de pinos do medidor de aceleração da gravidade.

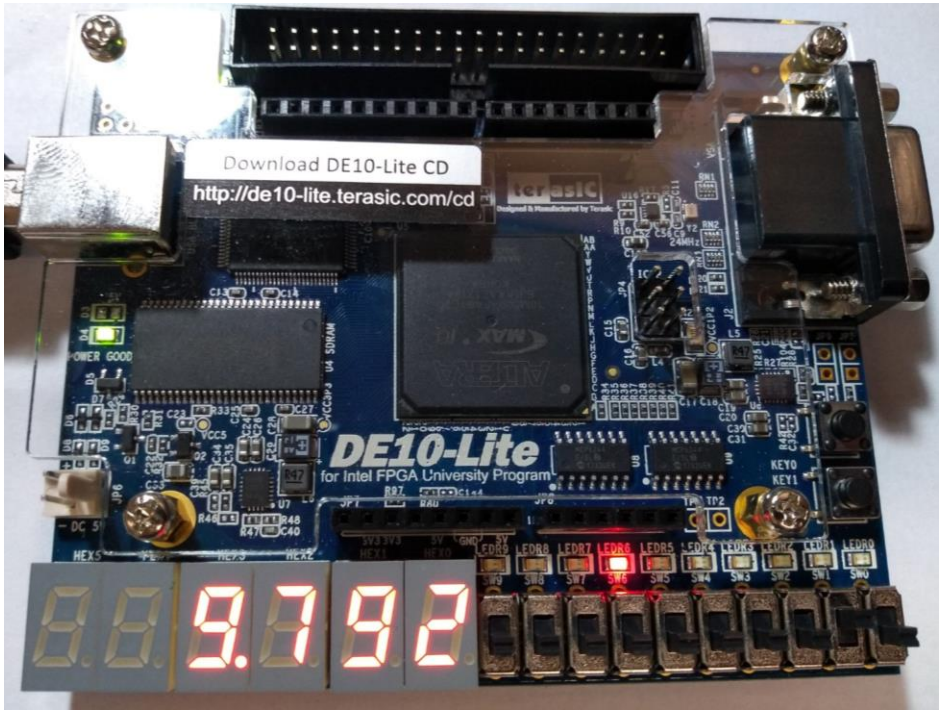
Node Name, Direction	Location	Node Name, Direction	Location
CLK, Input	PIN_P11	HEX3[7], Output	PIN_D22
HEX0[7], Output	PIN_D15	HEX3[6], Output	PIN_E17
HEX0[6], Output	PIN_C17	HEX3[5], Output	PIN_D19
HEX0[5], Output	PIN_D17	HEX3[4], Output	PIN_C20
HEX0[4], Output	PIN_E16	HEX3[3], Output	PIN_C19
HEX0[3], Output	PIN_C16	HEX3[2], Output	PIN_E21
HEX0[2], Output	PIN_C15	HEX3[1], Output	PIN_E22
HEX0[1], Output	PIN_E15	HEX3[0], Output	PIN_F21
HEX0[0], Output	PIN_C14	SNEG, Output	PIN_F20
HEX1[7], Output	PIN_A16	LED[9], Output	PIN_B11
HEX1[6], Output	PIN_B17	LED[8], Output	PIN_A11
HEX1[5], Output	PIN_A18	LED[7], Output	PIN_D14
HEX1[4], Output	PIN_A17	LED[6], Output	PIN_E14
HEX1[3], Output	PIN_B16	LED[5], Output	PIN_C13
HEX1[2], Output	PIN_E18	LED[4], Output	PIN_D13
HEX1[1], Output	PIN_D18	LED[3], Output	PIN_B10
HEX1[0], Output	PIN_C18	LED[2], Output	PIN_A10
HEX2[7], Output	PIN_A19	LED[1], Output	PIN_A9
HEX2[6], Output	PIN_B22	LED[0], Output	PIN_A8
HEX2[5], Output	PIN_C22	NOT_CS, Output	PIN_AB16
HEX2[4], Output	PIN_B21	RST, Input	PIN_B8
HEX2[3], Output	PIN_A21	SCLK, Output	PIN_AB15
HEX2[2], Output	PIN_B19	SDI, Output	PIN_V11
HEX2[1], Output	PIN_A20	SDO, Input	PIN_V12
HEX2[0], Output	PIN_B20		

Fonte: TERASIC, 2018.

Observação: No Quartus Prime a atribuição de pinos realizada em um projeto (utilizando a opção *Assignments> Pin Planner*) pode ser transferida para outra pasta de projeto (desde que o arquivo do projeto tenha o mesmo nome) utilizando as opções: *Assignments> Export Assignments* e *Assignments> Import Assignments.* Os arquivos contendo a configuração de pinos possuem a extensão: "**.qsf**" (por exemplo: **DISPLAY_GRAVIDADE.qsf**) e podem ser encontrados na principal de cada projeto.

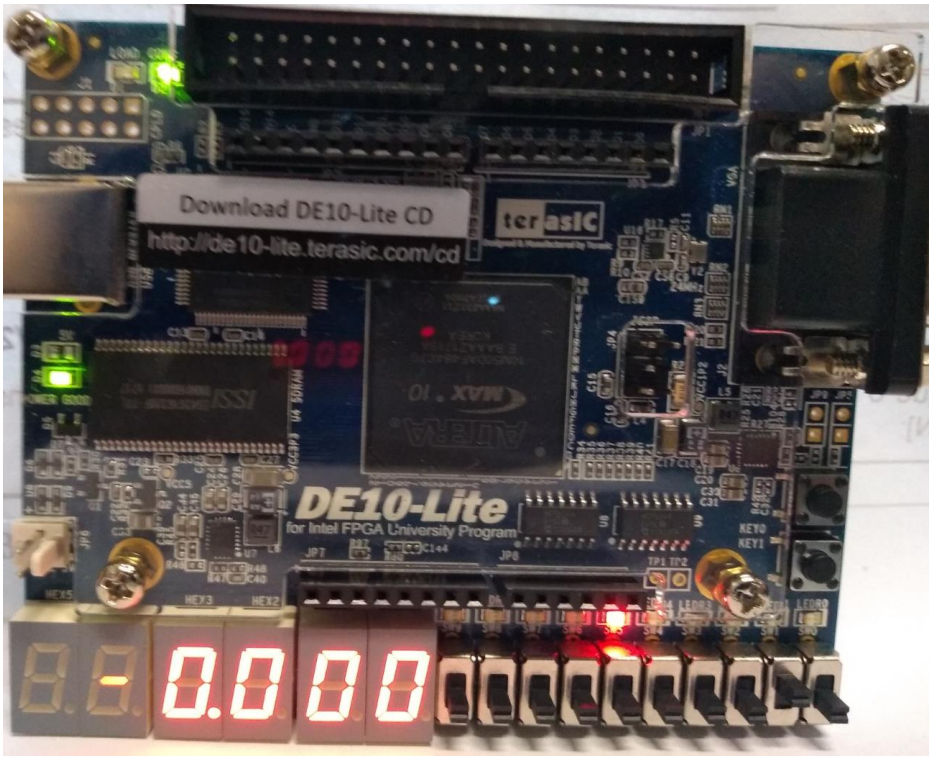
As imagens da Figura 34 e Figura 35 mostram os pontos de máxima aceleração gravitacional (aproximadamente 9,8 m/s^2) quando o sensor está na posição horizontal e o de mínima aceleração gravitacional quando o sensor está na posição vertical (0 m/s^2).

Figura 34: Valor máximo da aceleração gravitacional (posição horizontal).



Fonte: Autor.

Figura 35: Valor mínimo da aceleração gravitacional (posição vertical).



Fonte: Autor.

6 Referências bibliográficas

TERASIC. DE 10-Lite User Manual, Taiwan, 2018. Disponível em: <https://www.terasic.com.tw/cgi-bin/page/archive.pl?Language=English&No=1021>. Acesso em: 17 abr. 2019.

ANALOG DEVICES. Datasheet: ADXL345 DIGITAL ACCELEROMETER. 2015. Disponível em: <https://www.analog.com/media/en/technical-documentation/data-sheets/ADXL345.pdf>. Acesso em: 3 abr. 2019.

AVELINO, V. F., Aula1/Aula2/Aula3/Aula4_EL 6720_1S19, Centro Universitário FEI, Sistemas Digitais II – 1º Semestre de 2019