SISTEMA CORRELACIONADOR DIGITAL EM FPGA

RELATÓRIO FINAL DE PROJETO DE INICIAÇÃO CIENTÍFICA
(PIBIC/CNPq/INPE)

Andre Treno Ricarte (DAS , Bolsista PIBIC/CNPq)
Email : andre.ricarte@gmail.com

Dr. José Roberto Cecatto (DAS/INPE, Orientador)
E-mail: jrc@das.inpe.br

COLABORADOR

Dr. Cláudio Faria (DAS/INPE)

Julho de 2006

SISTEMA CORRELACIONADOR DIGITAL EM FPGA

BOLSISTA: André Treno Ricarte

ORIENTADOR: Dr. José Roberto Cecatto

Resumo

O presente projeto consistiu no desenvolvimento de uma interface de hardware e software para controle e aquisição de dados provenientes de um circuito correlacionador digital implementado com dispositivos lógicos programáveis FPGA. O circuito correlacionador desenvolvido é um dispositivo eletrônico capaz de obter o coeficiente de correlação cruzada entre dois sinais de entrada, possuindo aplicações em diversas áreas da ciência como, por exemplo, na área de radiointerferometria, onde são utilizados na realização de medidas da função visibilidade de uma fonte astronômica em arranjos radiointerferométricos. A função visibilidade é a medida obtida por um interferômetro a partir do sinal da fonte observada incidente em cada par de antenas do instrumento. Um computador é responsável pela programação e aquisição dos dados, referentes às integrações dos sinais obtidos pelo correlacionador. A comunicação entre o computador e o circuito correlacionador é feito através da interface paralela utilizando um cabo IDE (*Integrated Drive Electronics*) adaptado. Foi também desenvolvido uma interface computacional na linguagem C, com o auxílio da ferramenta de desenvolvimento GLADE, que consiste em um *front-end* para a biblioteca GTK (*GIMP toolkit*) que consiste em um conjunto de bibliotecas de componentes gráficos para desenvolvimento de aplicações. A interface desenvolvida é responsável pela programação do correlacionador e manipulação de um conjunto de dados que modelam as correlações realizadas. As medidas de correlação obtidas através da integração dos dois canais do circuito desenvolvido são adquiridas e visualizadas pela interface gráfica, esta interface executa a visualização dos dados por meio de uma biblioteca do software GNUPLOT compilada juntamente ao código fonte do sistema. Serão apresentados os principais resultados deste trabalho.

Palavras Chave: Radio Intererômetria, Correlacionador.

SUMÁRIO

1 INTRODUÇÃO

O presente projeto consiste no desenvolvimento de um circuito correlacionador digital utilizando dispositivos lógicos programáveis FPGA (Field Programmable Gate Array), os quais possibilitam uma grande flexibilidade na implementação de circuitos lógicos relativamente complexos. O circuito correlacionador proposto é um dispositivo eletrônico capaz de computar a função de correlação cruzada entre dois sinais, possuindo aplicações em diversas áreas da ciência como, por exemplo, na área de radio-interferometria, onde são utilizadas para medidas da função visibilidades (Wohlenben, et al.,1991) de uma fonte astronômica em arranjos rádio interferométricos que são utilizados para aumento da resolução espacial (Kellerman; Moran, 2001; Crutcher, 1994)..

A interferometria, ou mais especificadamente a rádio interferometria, tem como principio básico o aumento da resolução angular através do uso de dois ou mais radiotelescópios observando simultaneamente a mesma fonte. Em geral, vários telescópios são utilizados e combinados convenientemente, dois a dois, de forma a se obter um arranjo interferométrico, ou rádio interferômetro.

Os sinais provenientes de um par de radio telescópios são combinados por um circuito correlacionador, produzindo um padrão de interferência chamado função visibilidade. A relação entre a função visibilidade e a imagem da fonte observada é descrita por uma transformada de Fourier, assim sendo, é possível, a princípio, a partir do conjunto de visibilidades amostradas pelo arranjo obter a distribuição de brilho da fonte observada.

Os correlacionadores foram tradicionalmente desenvolvidos utilizando multiplicadores analógicos e filtros passa baixa, mas com o desenvolvimento das ultimas décadas da tecnologia digital, esta se tornou mais atrativa para o desenvolvimento dos mesmos, devido às suas vantagens, principalmente quanto ao custo e estabilidade do sistema. Mais recentemente, algumas tentativas têm sido realizadas para desenvolver os circuitos correlacionadores em dispositivos lógicos programáveis.

Neste contexto o presente trabalho consiste em um esforço de um desenvolvimento de um correlacionador digital em dispositivos FPGA (Field Programable Gate Array) para futuras

aplicações em observações rádio-astronômicas na faixa decimétrica. Assim sendo este correlacionador digital de um bit de quantização está sendo baseado no sistema atual utilizado pelo Brazilian Decimetric Array (BDA) (Sawant et al.), o qual foi desenvolvido no Indian Institute of Astrophisics (IIA) na Índia, que utilizou os chips correlacionadores projetados para o Radio Heliógrafo Nobeyama (Nakajima, et al, 1994) e para o Gauribidanur Radioheliograph (Ramesh, et al,1999).

O circuito correlacionador de BDA permite a correlacao de até 6 sinais de RF com largura de banda de 2.5 Mhz, digitalizados com um A/D 790 de 1 bit de quantização e amostrado a uma freqüência de 5 Mhz por meio de um flip-flop tipo D. Os sinais quantizados e amostrados passam por um unidade de atraso digital, construída por meio de registradores de deslocamento 74LS164, para ajuste de fase entre os sinais previamente escolhidos por um software. Os sinais, após o ajuste de fase, seguem para serem correlacionados em um chip correlacionador desenvolvido para o Radio Heliógrafo Nobeyama, o qual é responsável por produzir a medida de correlação entre os sinais. Os coeficientes de correlação obtidos são então lidos, por um software de aquisição, para posterior processamento.

A comunicação entre o computador e o circuito correlacionador é realizada através da interface paralela utilizando um cabo IDE (*Integrated Drive Electronics*) adaptado, esta comunicação é possível devido à implementação de uma interface computacional, na linguagem C, com o auxílio da ferramenta de desenvolvimento GLADE, que consiste em um *front-end* para a biblioteca GTK (*GIMP toolkit*) onde um conjunto de bibliotecas de componentes gráficos é disponibilizado para desenvolvimento de aplicações. A interface desenvolvida é responsável pela programação do correlacionador e manipulação de um conjunto de dados que modelam as observações obtidas pelo correlacionador. Os coeficientes das correlações obtidas através da integração dos dois canais do circuito desenvolvido são adquiridas e visualizadas pela interface gráfica, esta interface executa a visualização dos dados por meio de uma biblioteca do software GNUPLOT compilada juntamente ao código fonte do sistema

2 TÉCNICAS DE CORRELAÇÃO DE SINAIS

Correlação significa dependência, em termos de sinais, a correlação é a medida de dependência ou similaridade entre um sinal em relação ao outro. A correlação cruzada entre dois sinais contínuos $x(t)$ e $y(t)$, pode ser discutida pela seguinte equação (2.1):

$$R_{xy} = \lim_{T->\infty} \frac{1}{2T} \int_{-T}^{T} x(t).y(t+\tau)d\tau \tag{2.1}$$

onde T é o intervalo em que se deseja integrá-los $x(t)$ o sinal sem deslocamento e $y(t+\tau)$ o segundo sinal acrescido do deslocamento.

A correlação cruzada contem informação sobre a diferença da fase entre os dois sinais $x(t)$ e $y(t+\tau)$, ou seja, se correlacionarmos x (equação 2.2) e y (equação 2.3) temos (equação 2.4).

$$x = A.\sin(\omega t + \phi_1) \tag{2.2}$$

$$y = B.\sin(\omega t + \phi_2) \tag{2.3}$$

$$R_{xy}(t) = A.B.\cos(\omega t + \phi_1 - \phi_2) \tag{2.4}$$

Quando dois sinais de mesma freqüência são correlacionados, o resultado é um sinal de mesma freqüência e fase o qual é a diferença de fase entre os dois sinais correlacionados.

2.1 Aplicação em Rádio-Interferometria

A utilização de uma única antena (rádio telescópio) na observação de uma fonte emissora limita a resolução espacial (limite de difração da antena), ao passo que, com a utilização de elementos mais sofisticados tecnicamente, que têm sido desenvolvidos para combinar elementos simples em arranjos de múltiplos elementos, o qual trabalhando juntos formam um único radio telescópio, permitindo, com isso, um aumento da resolução espacial. Nestes instrumentos a resolução não é mais determinada pelo limite de difração da antena, mas sim pela distância que separam os radio telescópios, esta distância é chamada de linha de base, onde a linha de base pode se estender para Km.

Um radio-interferômetro básico composto por dois radio-telescópios observando simultâneamente um plano de onda proveniente de uma fonte no céu, como o ilustrado na

FIGURA 2.1. Este par de antenas possui um espaço físico entre si, que recebe o nome de linha de base, considerando que este arranjo esta observando uma fonte emissora de radiação com um comprimento de onda λ, esta fonte forma um ângulo θ em relação a linha de base B.

A radiação captada pelas antenas deve ser combinada de modo conveniente pelo circuito correlacionador gerando uma medida correspondente à similaridade entre os sinais. Devido à diferença de percurso δ existe uma atraso τ entre os sinais que chegam ao correlacionador.

A diferença de percurso δ é influenciada pelo ângulo de incidência da radiação, de acordo com a equação 2.4 temos:

$$\delta = B.\sin\theta \tag{2.4}$$

Caso δ seja um valor múltiplo inteiro de λ as radiações detectadas pelas antenas estarão com uma interferência construtiva entre seus sinais que consiste na sobreposição dos sinais.

Para determinarmos a diferença de fase entre os sinais temos a seguinte equação 2.5

$$\phi = (B.\sin\theta)2\pi / \lambda \tag{2.5}$$

Como demonstrado na equação 2.5, a diferença de fase entre os sinais é determinada pela variação do ângulo θ. Considerando os sinais provenientes das antenas ilustradas na FIGURA 2.1 temos os sinais provenientes das antenas V1(equação 2.6) e V2 (equação 2.7)

$$V_1(t) = E.\cos(\omega t) \tag{2.6}$$

$$V_2(t) = E.\cos(\omega t - \tau_g) \tag{2.7}$$

Onde τ_g é o atraso sofrido pelo sinal proveniente da antena a direita

A multiplicação realizada em um determinado período de tempo pelo correlacionador corresponde a seguinte equação 2.8

$$\frac{E^2}{T}\int_0^T \cos(\omega t).\cos(\omega t - \tau) \tag{2.8}$$

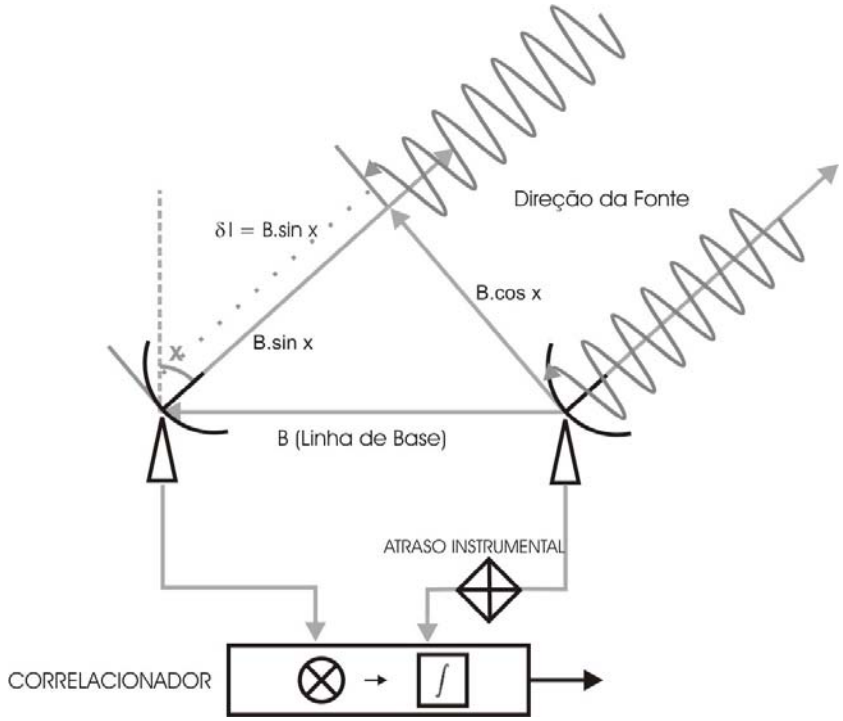

FIGURA 2.1 – Interferômetro de dois elementos, a diferença de percurso, o atraso instrumental, e o correlacionador consistindo de um circuito multiplicador e integrador.

Assim, esse interferômetro produz duas medidas de correlação correspondentes às componentes cosseno e seno da fonte pontual observada:

$$R(u)_R \propto I_\theta \cos(2\pi u\theta) \tag{2.9}$$

$$R(u)_I \propto I_\theta \sin(2\pi u\theta) \tag{2.10}$$

onde R_R é a componente real (cosseno) e R_I a componente imaginária (seno) observada. Na FIGURA 2.3 tem-se um exemplo de possíveis franjas interferométricas cosseno e seno do radiointerferômetro complexo.

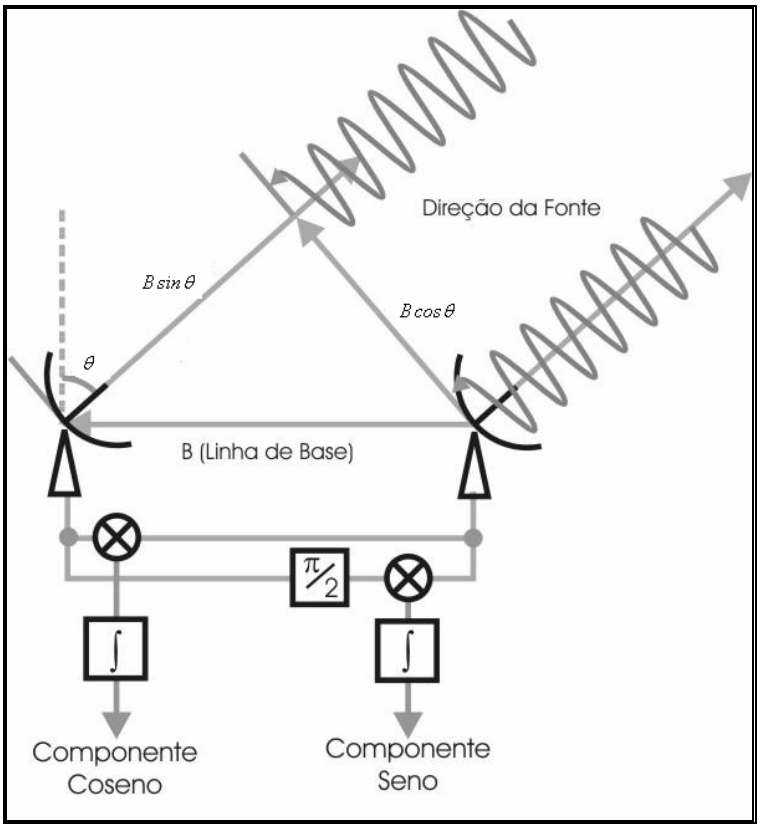

FIGURA 2.2 - Radiointerferômetro correlacionador complexo

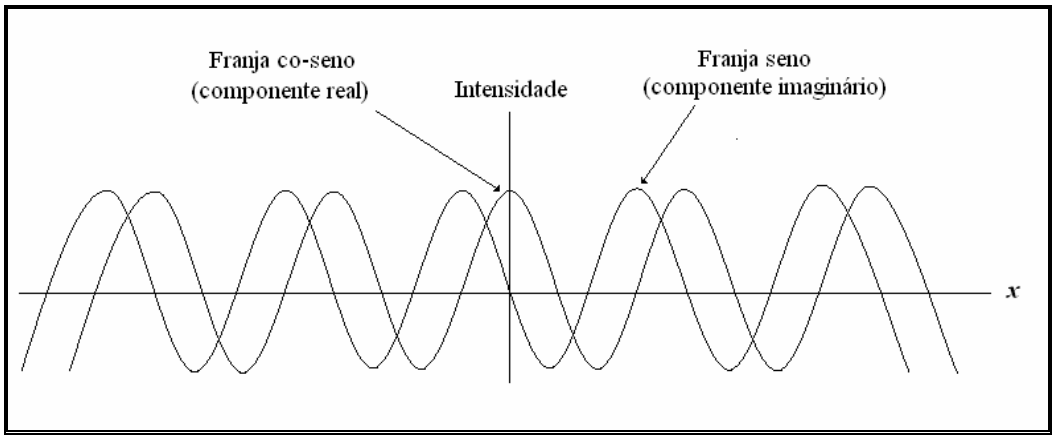

FIGURA 2.3 - Franjas interferométricas obtidas por um radiointerferômetro correlacionador complexo

A partir das medidas dessas componentes é possível recuperar a amplitude (*I*) e fase (φ) da fonte utilizando-se as seguintes equações:

$$I = \sqrt{R_R{}^2 + R_i{}^2} \tag{2.11}$$

$$\varphi = -tan^{-1}\left(R_I / R_R\right) \tag{2.12}$$

Utilizando-se uma representação complexa na forma exponencial para definir a resposta do correlacionador, tem-se que:

$$R(u) \propto I_\theta \, exp\left(i\, 2\pi\, u\, \theta\right) \tag{2.13}$$

Resposta para uma Distribuição de Brilho Contínua

A resposta do radiointerferômetro também pode ser expressa considerando-se uma fonte unidimensional com extensão angular finita *S*, cuja intensidade da distribuição de brilho pode representada por uma função real $I(x)$, onde x é o eixo relativo à posição angular da fonte no céu com origem no centro de fase do arranjo, tal como o ângulo θ. Decompondo a distribuição de brilho por um conjunto de fontes infinitesimais com dimensão dx, e integrando-se a contribuição de cada resposta individual para toda a dimensão *S* da fonte, tem-se que a resposta a uma distribuição de brilho unidimensional é dada por:

$$V(u) = \int_S I(x)\, exp\left(i\, 2\pi\, u\, x\right) dx \tag{2.14}$$

onde *V(u)* é usualmente chamada de visibilidade complexa da fonte, a qual é medida considerando-se uma componente de freqüência espacial *u*. A equação 2. corresponde à transformada de Fourier da distribuição de brilho da fonte.

Para o caso de uma distribuição de brilho bidimensional definida por *I(x,y)*, onde *y* é a dimensão espacial ortogonal a *x*, a equação do interferômetro torna-se:

$$V(u,v) = \int I(x,y)\, exp\left(i 2\pi (ux + vx)\right) dxdy \tag{2.15}$$

onde o plano *xy* define a superfície de brilho da fonte e *u* e *v* são as freqüências espaciais associadas aos eixos *x* e *y* respectivamente.

A equação 2. corresponde à transformada de Fourier de distribuição de brilho da fonte, e assim sendo, um radiointerferômetro é um instrumento que permite realizar uma medida específica da visibilidade complexa da fonte em um ponto específico *(u,v)* do plano de

Fourier, também referenciado na literatura como plano *uv*, que é dependente do vetor linha de base do arranjo, do comprimento de onda observado e da posição angular da fonte (centro de referência de fase).

Em um arranjo radiointerferométrico vários radiotelescópios são utilizados e combinados, dois a dois, de forma a se obter uma medida da função visibilidade para cada par interferométrico. Assim torna-se possível através do conjunto de visibilidades amostradas estimar a distribuição de brilho da fonte observada utilizando uma transformada inversa de Fourier, em um processo conhecido como Síntese de Fourier. Assim torna-se possível, a partir do conjunto de visibilidades amostradas, estimar a distribuição de brilho da fonte observada utilizando uma transformada inversa de Fourier.

3 CIRCUITO DIGITALIZADOR

Uma das funções de um amplificador operacional é a de trabalhar como um comparador. Em inúmeras situações surge a necessidade de comparar dois sinais, sendo que um destes sinais é uma referência para o outro sinal.

Um exemplo desta aplicação prática: Quando temos um sinal analógico proveniente de uma antena e temos a necessidade de saber quando o sinal passa pelo nível de tensão zero. Em uma das entradas do amplificador colocamos o sinal a ser comparado (A) e no outro terminal colocamos o sinal de referência (B). O comparador deverá emitir um sinal de saída quando sinal A ultrapassar o sinal B e se restabelecer imediatamente quando o sinal A for menor que o sinal B.

Os comparadores produzem saídas na forma de quadrádica (forma de pulsos) em função do nível do sinal aplicado. O que acontece realmente é que a saída do comparador está sempre em um valor alto, denominado saturação positiva (+Vsat), ou em um valor baixo, denominado saturação negativa (-Vsat).

Temos dois tipos de comparadores os Inversores e os Não Inversores. O circuito utilizado no projeto para comparar o sinal proveniente de uma antena é o Não Inversor para manter uma certa similaridade do sinal.

O funcionamento de um comparador é bastante simples: devido ao alto ganho do amplificador a diferença de tensão existente entre as entradas inversora e não inversora leva a saída para +Vsat ou –Vsat conforme a diferença positiva ou negativa entre as entradas.

De acordo com a FIGURA 3.1 temos as seguintes equações 3.1 e 3.2:

$$V_0 = +\text{Vsat, quando vi} > 0 \qquad (3.1)$$

ou

$$V_0 = -\text{Vsat, quando vi} < 0 \qquad (3.2).$$

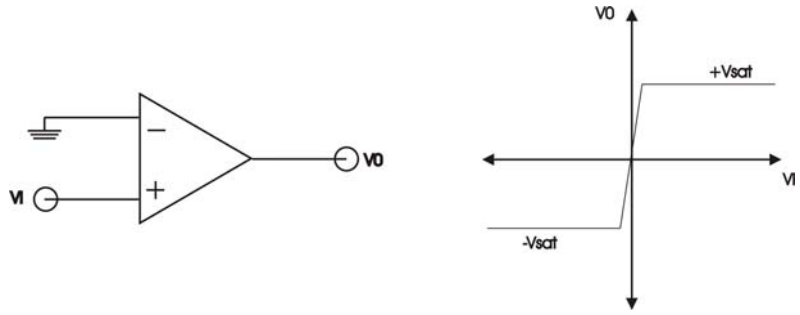

FIGURA 3.1 – Tensão de Saturação de acordo com voltagem aplicada em VI
FONTE: Pertence Junior, Antonio, Eletrônica analógica: amplificadores operacionais e filtros ativos: teoria, projetos, aplicações e laboratório / Antonio Pertence Júnior. Porto Alegre: Bookman,2003,85 p.

3.1 Limitando a tensão de saída

Para limitarmos a tensão de saída do comparador basta adicionar um diodo Zener (Dz) de valor Vz = 5,1 V (exemplo: 1N751, 1N4733) e um resistor (R) de 330 Ohm, para limitar a corrente sobre o diodo, de acordo com a FIGURA 3.2, e teremos uma tensão de saída compatível com circuitos digitais da família TTL.

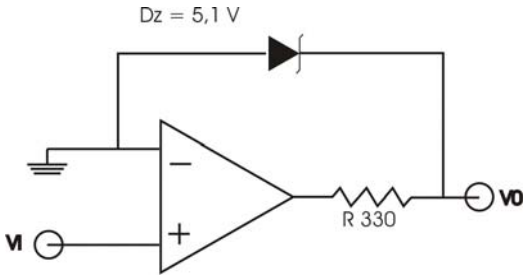

FIGURA 3.2 – Esquema do circuito digitalizador utilizado
FONTE: Pertence Junior, Antonio, Eletrônica analógica: amplificadores operacionais e filtros ativos: teoria, projetos, aplicações e laboratório / Antonio Pertence Júnior. Porto Alegre: Bookman,2003,87 p.

A FIGURA 3.3 a seguir mostra uma senóide de entrada no comparador com uma amplitude de sinal de 18v e na saída do comparador um sinal de nível lógico quadrádico

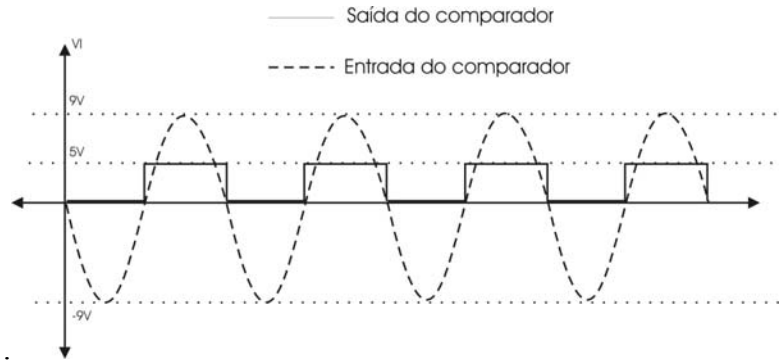

FIGURA 3.3 - Resultado obtido da digitalização de um sinal analógico.

FONTE: Pertence Junior, Antonio, Eletrônica analógica: amplificadores operacionais e filtros ativos: teoria, projetos, aplicações e laboratório / Antonio Pertence Júnior. Porto Alegre: Bookman,2003,85 p.

4 DESENVOLVIMENTO DO CORRELACIONADOR EM FPGA

4.1 Ambiente de Desenvolvimento FPGA

Para o desenvolvimento e teste do circuito correlacionador digital foi utilizado o Kit FPGA FLEX 10K da marca Altera (FIGURA 4.1) devido a sua capacidade ser suficiente e a facilidade em utilizar os componentes eletrônicos inclusos no kit como: leds, jumpers, displays de 7 segmentos, chaves e principalmente os barramentos de I/O que são facilmente conectados aos pinos de I/O do chip FPGA Flex10K. Esta conexão é feita de maneira transparente pelo software de desenvolvimento MAX PLUS II (FIGURA 4.2).

FIGURA 4.1– Foto do Kit FPGA utilizado para confecção do circuito correlacionador

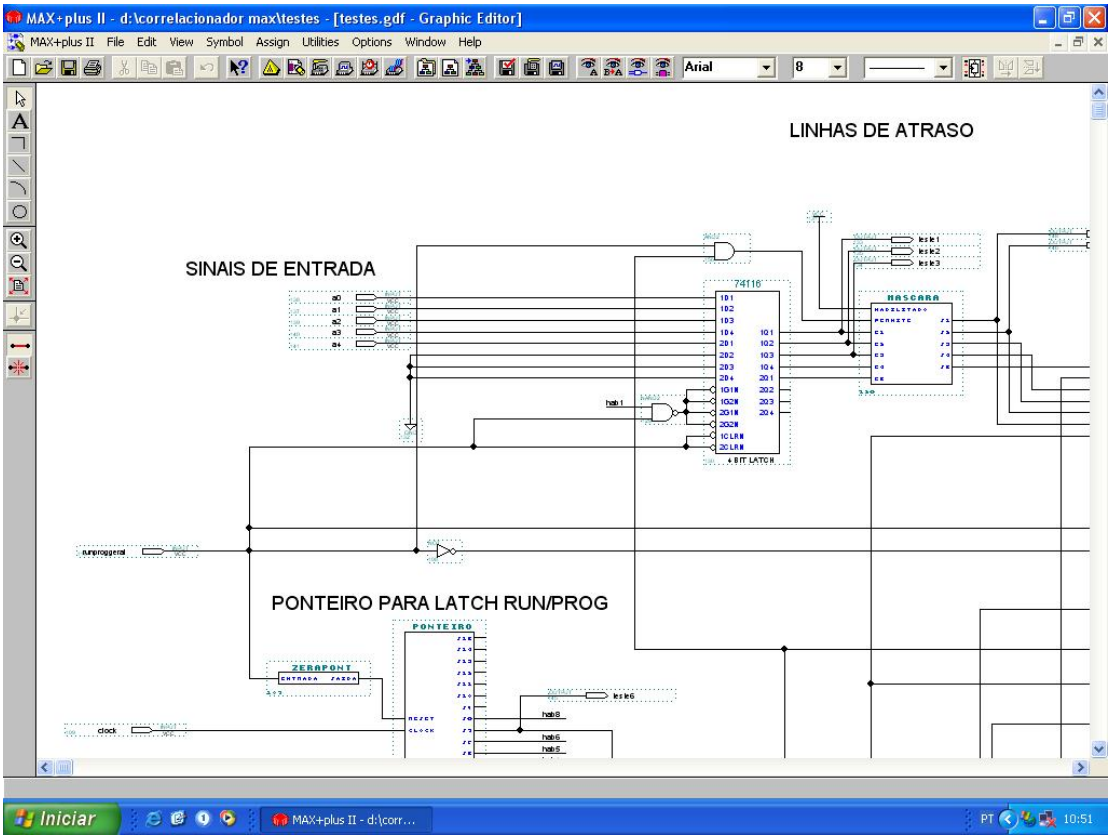

FIGURA 4.2 – Imagem do Ambiente de Desenvolvimento MAX PLUS II utilizado no projeto

O software possui também um poderoso ambiente de simulação (FIGURA 4.3) que foi utilizado nos testes do correlacionador digital, os recursos disponíveis no software são capazes de simular inclusive um possível atraso entre os blocos lógicos programáveis do chip FPGA.

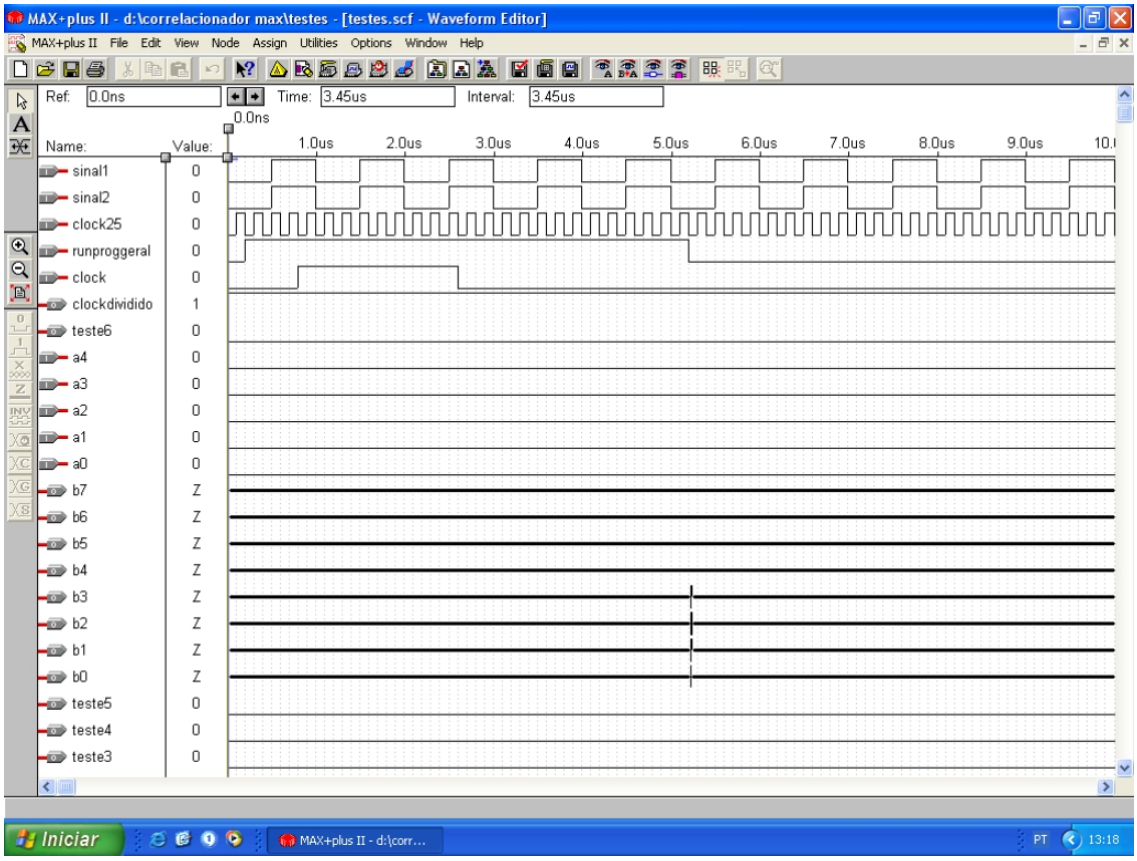

FIGURA 4.3 – Imagem do Ambiente de Simulação MAX PLUS II utilizado para simular os sinais para o circuito correlacionador.

4.2 Diagrama de Blocos do Correlacionador

Para o sistema correlacionador foi construído um digitalizador, utilizando o amplificador operacional AD741 devido ao baixo custo, para fornecer ao amostrador, localizado na FIGURA 4.4, um sinal TTL correspondente ao sinal analógico a ser correlacionado. Este sinal é amostrado por flip-flop tipo D com a freqüência de clock igual a 5 Mhz.

Após esta amostragem, os sinais devem entrar em um processo de sincronização para posteriormente serem multiplicados, esta etapa do processo de correlação é feita por um multiplicador de 1 bit, que consiste em uma porta XNOR.

Durante um intervalo de tempo predeterminado chamado de tempo de integração os valores da multiplicação devem ser acumulados para posteriormente serem enviados ao computador por um circuito de comunicação com a interface paralela.

A seqüência destes fatos é garantida por um circuito chamado de unidade de controle, FIGURA 4.5, este é responsável por configurar o correlacionador com o atraso entre os sinais

e garantir o funcionamento correto do circuito, sem que haja conflito entre os componentes do correlacionador.

Nos próximos capítulos iremos abordar o funcionamento de cada componente do correlacionador, desenvolvido no Kit FPGA, de maneira individual.

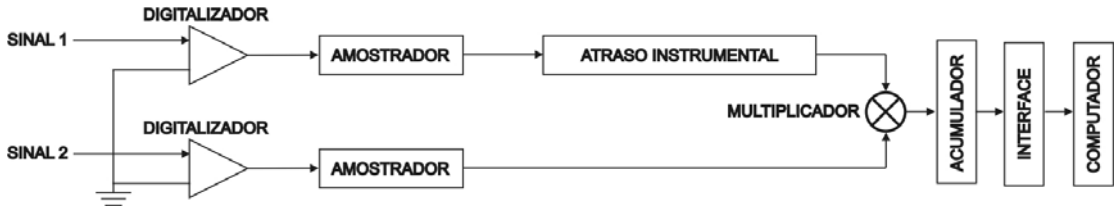

FIGURA 4.4 – Diagrama de blocos do circuito correlacionador

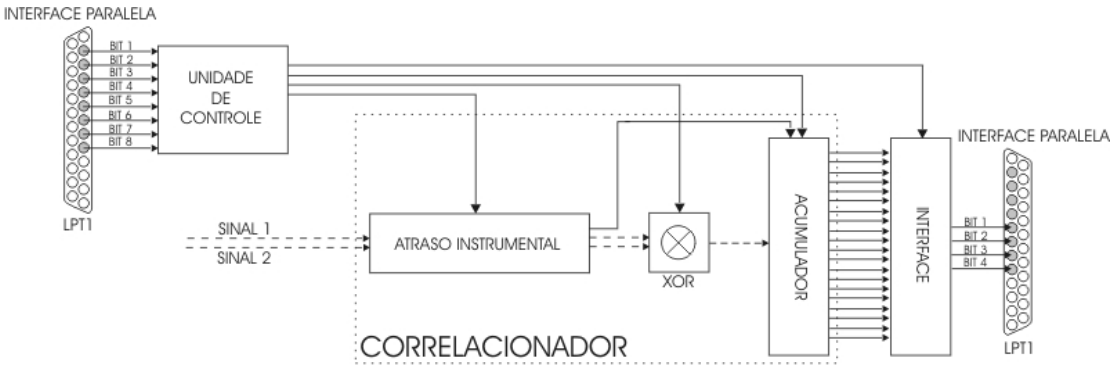

FIGURA 4.5 – Diagrama de blocos do correlacionador desenvolvido no FPGA

4.3 Correção do Atraso entre os Sinais

Para corrigir a diferença de percurso, foi implementado um circuito com a capacidade de sincronizar os sinais provenientes das antenas. O circuito é constituído por um registrador de deslocamento, onde os sinais digitalizados são inseridos com a velocidade do clock de amostragem e um circuito multiplexador por onde os sinais são retirados.

Seu funcionamento possui a seguinte característica o sinal que está dentro do registrador de deslocamento irá se deslocar conforme o sinal de clock (0,2μs), considerando que haja um atraso entre os sinais de 1 μs, o sinal que esta dentro do registrador de deslocamento após 5 pulsos do sinal de clock estará atrasado em exatamente em 1 μs estando então em fase com o outro sinal.

Para podermos retirar o sinal que agora está em fase e deve entrar em processo de multiplicação com o outro sinal, foi implementado um circuito demultiplexador que recebe o atraso da unidade de controle e retira o sinal que está no registrador de deslocamento, para posteriormente ser multiplicado.

A FIGURA 4.6 mostra como foi implementado o circuito que realiza o atraso instrumental máximo entre os sinais de 16 ciclos de clock entre os sinais.

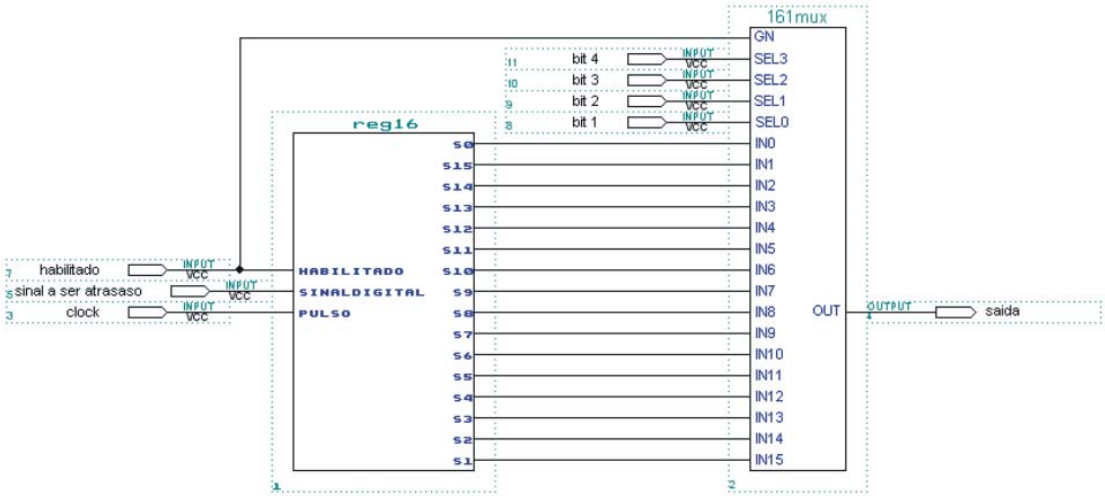

FIGURA 4.6 – Atraso instrumental realizado pelo correlacionador.

4.4 Sincronismo

Como foi citado no capitulo anterior para os sinais possam ser multiplicados os mesmos devem estar sincronizados, mas o circuito acima descrito não garante ainda o sincronismo entre os sinais.

Considerando que a multiplicação é realizada por uma porta XNOR ocorrerá o inconveniente de podermos estar multiplicando o sinal que esta na saída do registrador de deslocamento, que está em nível lógico 0, com o sinal que não possui atraso e esta ligado diretamente ao multiplicador, caso este esteja em nível lógico 0 também.

Para corrigirmos este inconveniente foi implementado um timer para inicio da integração, este timer consiste em um contador decrescente que é inicializado com o valor do atraso existente entre os sinais. O timer tem inicio quando os atrasos foram passados e o circuito começa a funcionar (runprog = 0). Quando o timer zera o valor ele habilita os componentes (habilitado

= 1) responsáveis pelo atraso, multiplicação, acumulador e o timer de 100ms. A FIGURA 4.7 mostra a implementação do Timer de Integração.

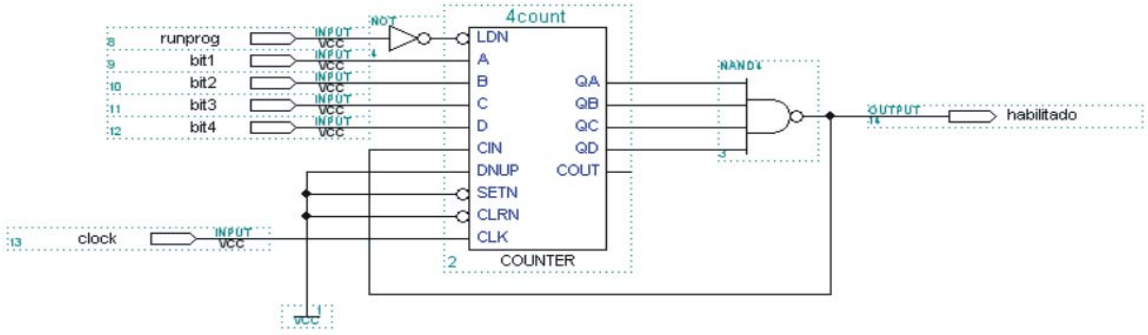

FIGURA 4.7 – Timer para inicio da integração de 100 ms

4.5 Circuito Multiplicador

Um multiplicador de um bit é responsável por comparar a similaridade entre os sinais do correlacionador. Este multiplicador consiste em uma porta XNOR que possui os seguintes resultados de acordo com a Tabela 4.1:

Tabela 4.1 – Tabela verdade de uma porta XNOR

Sinal 1	Sinal 2	Resultado
0	**0**	**1**
0	**1**	**0**
1	**0**	**0**
1	**1**	**1**

Esta porta lógica recebe em uma de suas entradas o sinal que passou pela correção de percurso (atraso instrumental) e em sua entrada restante o sinal que não tem percurso a ser corrigido.

Após a inserção do multiplicador o protótipo do correlacionador fica com a seguinte composição (FIGURA 4.8):

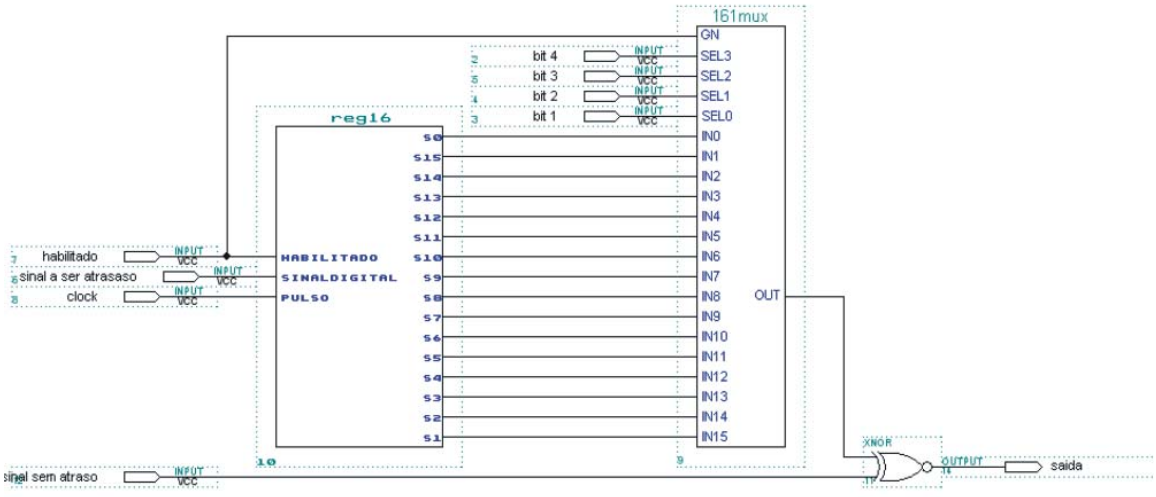

FIGURA 4.8 – Atraso instrumental acrescido de uma porta XNOR responsável pela multiplicação dos sinais

4.6 Circuito Acumulador

Durante o período de tempo de 100ms (t), tempo de integração sugerido no projeto, deve-se acumular os resultados das multiplicações dos sinais de acordo com os pontos de amostragens. A saída da porta XNOR esta ligada ao clock de um contador se os sinais são iguais a saída da multiplicação (da porta) é igual a 1, quando isto ocorre o contador é incrementado e acumulando assim os resultados das multiplicações durante o tempo de integração. O tempo de integração do correlacionador é próximo de 105ms. Para determinarmos o número máximo obtido pelo acumulador responsável pela integração, devemos dividir o tempo de integração do correlacionador pelo tempo do ciclo de amostragem.

Coeficiente máximo de correlação $\cong 0{,}105s/0{,}0000002s \cong$ 525000.

Durante a implementação do circuito acumulador foi utilizado um contador de 22 bits, criamos um acumulador com esta capacidade, para caso seja necessário, podermos utilizar um maior clock de amostragem. Se utilizarmos um clock de amostragem igual a 25MHz teremos que suportar um coeficiente máximo de correlação igual a 2600000 necessitando então de um acumulador de 22bits.

O acumulador foi sintetizado com linguagem VHDL que é uma linguagem capaz de programar um hardware para que funcione de diferentes formas, o contador foi feito desta maneira devido à facilidade do software em construir componentes com esta linguagem.

O circuito acumulador foi implementado de acordo com a FIGURA 4.9.

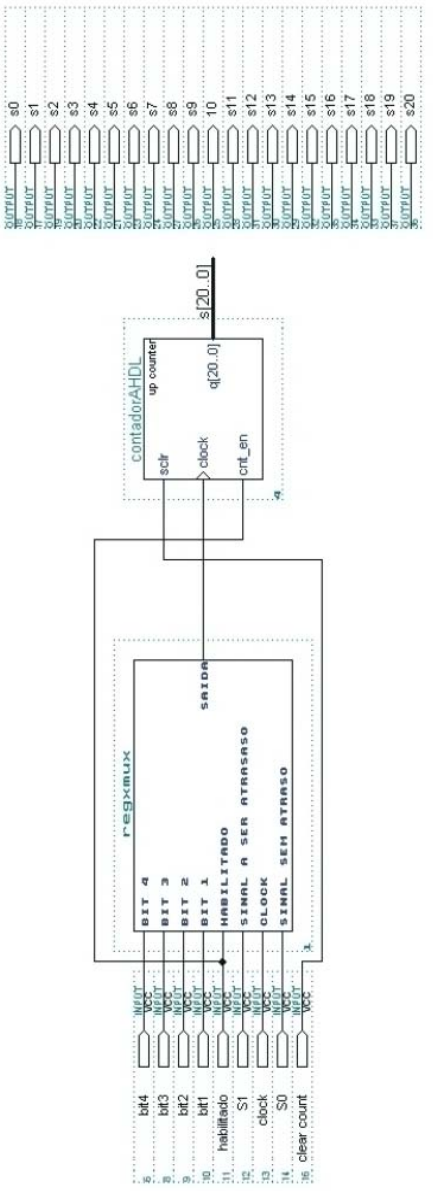

FIGURA 4.9 – Circuito responsável pela acumulação da integração.

4.7 Comunicação com a Interface Paralela

Toda a comunicação do circuito correlacionador com o computador é feita através de latchs (FIGURA 4.10) que são um conjunto de Flip-Flops que ao receber um pulso de clock armazena o valor de sua entrada até que seja dado outro pulso de clock.

O funcionamento de um latch está descrito na TABELA 4.2 onde L corresponde ao nível lógico 0, H ao nível lógico 1, X pode ser tanto 0 quanto 1 que as saídas não serão afetadas e por último Qo que corresponde a todas saídas em 0 (L)

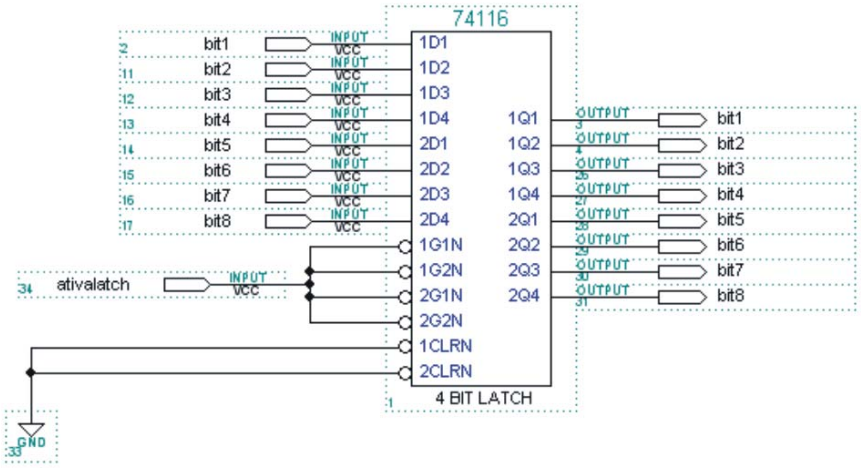

FIGURA 4.10 – Latch componente utilizado para armazenar os valores da correlação.

Tabela 4.2 – Tabela verdade de um latch

Inputs				Outputs
	Enable			
CLRN	G1N	G2N	D	Q
L	X	X	X	L
H	L	L	L	L
H	L	L	H	H
H	X	H	X	Qo
H	H	X	X	Qo

FONTE : Software Max – Plus II

Se considerarmos um número grande de antenas para um interferômetro correlacionador a quantidade de valores obtidos será enorme porque para cada par de sinais correlacionados existe um número de 20 bits a ser lido, logo, se tivermos que obter os valores correspondentes a dez correlações teremos que ler 200 bits.

Para o computador poder ler todos estes valores foi criada uma interface com a porta paralela capaz de enviar qualquer quantidade de informação.

Seu funcionamento é simples porque é baseado em um conjunto de latchs (Elemento 74116 na figura 4.11) que ao receberem os valores das multiplicações, contidos nos acumuladores do correlacionador, enviam esta informação, através de um barramento para porta paralela, de acordo com o latch acionado pelo PC.

Foi incluído também um componente tri-state na saída dos latchs, esta inserção corrigiu o problema ocasionado por um latch que ao enviar uma informação ao barramento poderia curto-circuitar a saída dos outros latchs ligados ao barramento.

O funcionamento de um tri-state é descrito através da TABELA 4.3

Tabela 4.3 – Valores para funcionamento de um Tri-state

Conectado	**Entrada**	**Saída**
0	**0**	**X**
0	**1**	**X**
1	**0**	**0**
1	**1**	**1**

FONTE : Software Max – Plus II

Como é demonstrado na tabela acima quando o pino *conectado* recebe o valor 1 sua saída recebe o mesmo valor da entrada, quando o pino *conectado* é igual a 0 sua saída é desconectada do barramento, impedindo assim o curto circuito nas porta lógica que esta conectada na entrada do tri-state.

O circuito de interface funciona da seguinte maneira: Após o término da integração, o circuito PC quando recebe um pulso no pino de clock ativa o latch que contém os bits menos significativos. Ao receber outro pulso de clock do PC o circuito ativa o latch que contém os bits mais significativos, este processo tem continuidade até que todos os bits, referentes a correlação de todos os pares de sinais contidos no correlacionador, tenham sido enviados para o computador.

A FIGURA 4.11 mostra parte do circuito de inteface.

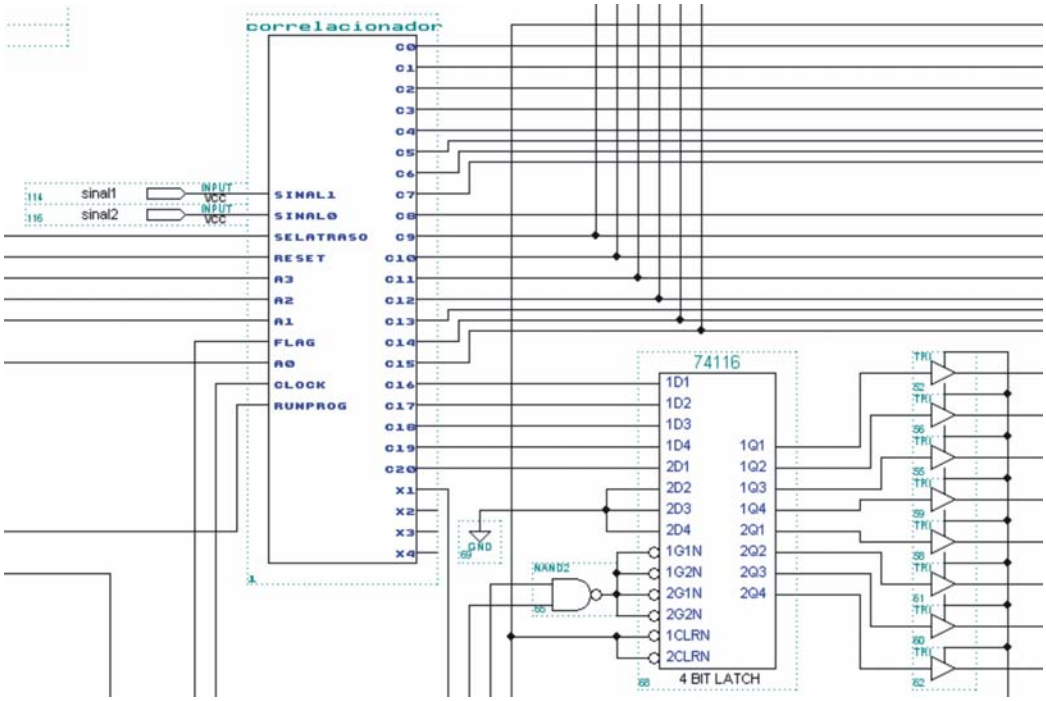

FIGURA 4.11 – Trecho do circuito responsável pela interface com o computador

4.8 Unidade de Controle

Para garantirmos a execução correta do correlacionador foi construída uma unidade de controle capaz de suprir a necessidade que encontramos para programar e enviar as informações ao correlacionador digital.

Foi definido que a unidade de controle funcionará com 7 bits (FIGURA 4.12) para programação do circuito correlacionador onde:

- O bit run/prog define se o correlacionador está em modo de programação (prog) ou está em modo de funcionamento (run).

- O bit clock incrementa o circuito acionador de latchs

- O bits 3 até o bit 7 informam o atraso entre os sinais. O bit 3 informa qual sinal está atrasado.

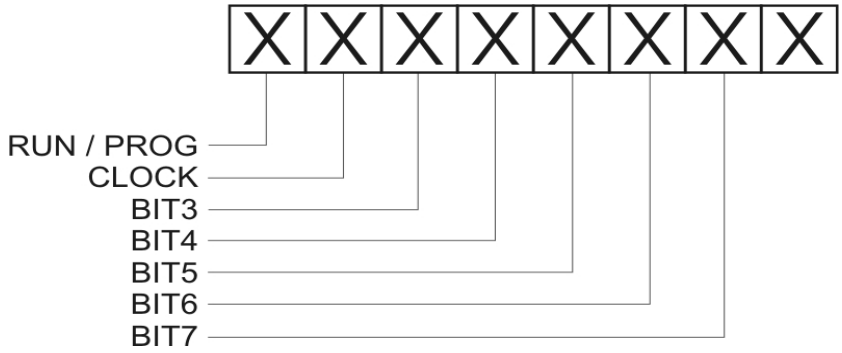

FIGURA 4.12 – Bits para programação do correlacionador totalizando 1 Byte

Para leitura os bits utilizados são os mesmos, porém o bit 3 informará que a correlação está pronta, ou seja já se passaram os 100ms do tempo de integração e o software já está pronto para obter os resultados, e os bits 4 ao 7 informarão os valores obtidos pelo acumulador durante o tempo de integração.

Tendo definido os bits de programação e envio da unidade de controle foram construídos os circuitos responsáveis pelo controle dos outros componentes do correlacionador.

Entre os diversos circuitos que compõem o correlacionador dois componentes distintos se destacam, são eles:

1. Timer de integração de 100ms

2. Acionador de Latchs

4.8.1 Timer de Integração

Após os sinais estarem em sincronismo o timer de integração inicia sua contagem de tempo, quando o timer chega ao limite de tempo proposto pelo projeto de 100ms ele emite um sinal para que os valores do acumulador passem ao circuito de interface e fiquem armazenados até que o circuito de controle os envie ao computador.

Este componente FIGURA 4.13 foi construído em linguagem vhdl devido à facilidade de criação do software que esta sendo usado.

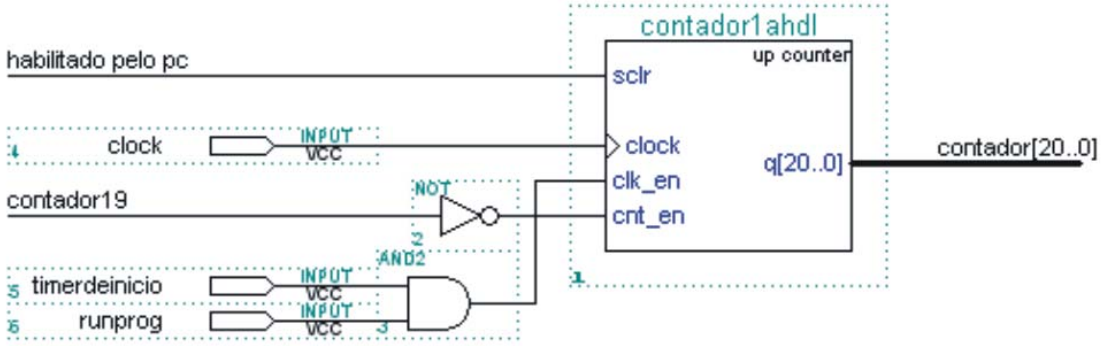

FIGURA 4.13 – Timer responsável pelo tempo de integração de 100 ms

4.8.2 Acionador dos Latchs

Para o circuito de acionador foi implementado um contador, responsável por passar o endereço do latch a ser acessado, e um demultiplexador que recebe o valor fornecido pelo computador para acessar suas saídas.

A FIGURA 4.14 mostra a implementação do circuito de controle, a FIGURA 4.15 acrescenta um circuito auxiliar (zerapont) que zera o valor do circuito de controle a cada subida ou decida do sinal do pino runprog.

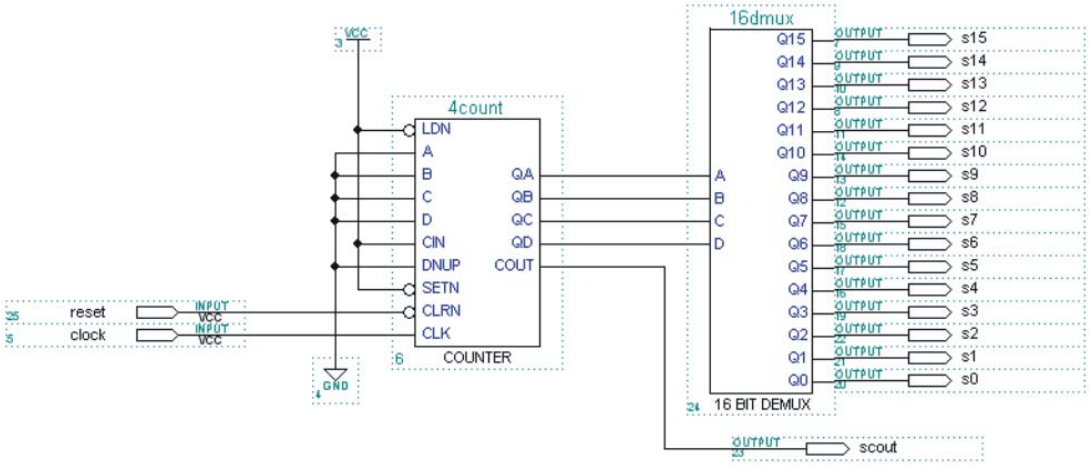

FIGURA 4.14 – Circuito responsável por ativar os latchs de programação e acumulação

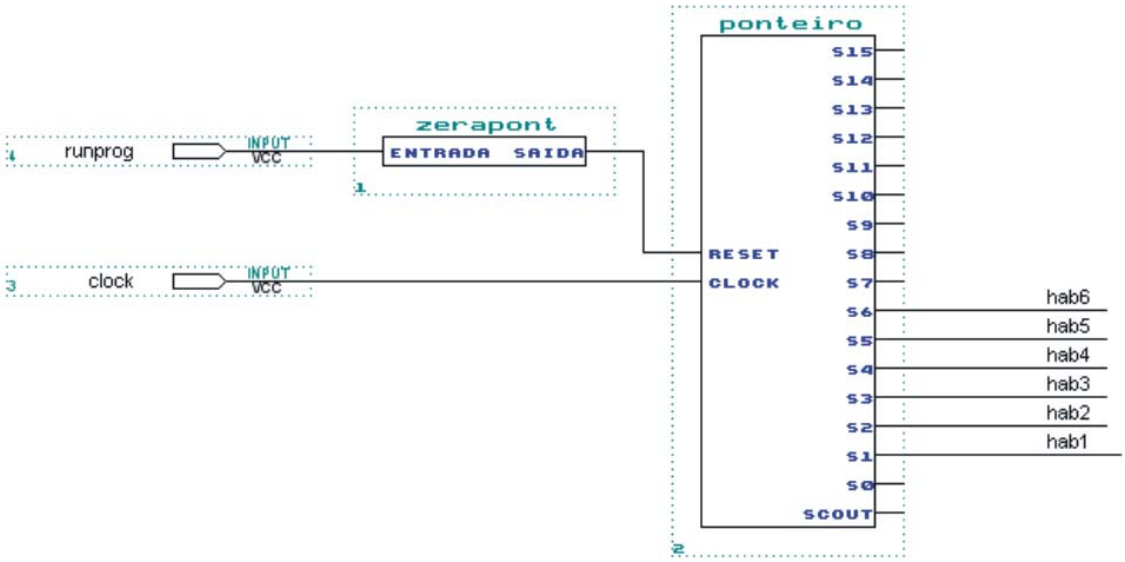

FIGURA 4.15 – Circuito responsável por zerar a saída do circuito ponteiro.

As saídas mostradas acima (hab1 até hab6) são responsáveis por acionarem os latchs responsáveis pelo armazenamento do atraso entre os sinais e pelo acionamento dos latchs que possuem o valor obtido pela integração que estão localizados dentro do circuito de interface.

5 SIMULAÇÕES E TESTES NO CIRCUITO CORRELACIONADOR

Uma vez que todos os sub-circuitos do correlacionador foram desenvolvidos realizou-se, utilizando o Max – Plus II, simulações para validação e testes dos mesmos. Para isto foi realizado 3 tipos de simulações:

1.Teste da Unidade de Atraso Instrumental.

2.Taxa de amostragem fixa de 5 Mhz com um sinal variando de 0.5 Mhz até 5 Mhz.

3.Taxa de amostragem variando de 10 Mhz, 20 Mhz e sinal fixo em 5 Mhz

O tempo de integração da simulação, com a taxa de amostragem fixa foi igual a 105 ms, enquanto para a simulação com a taxa de amostragem variante o tempo de integração variou de 52 ms para 10 Mhz e 25 ms para 20 Mhz, estas duas ultimas simulações mostra que o circuito suporta velocidades de clock superiores a qual ele foi projetado de 5 Mhz.

5.1 Exemplo de uma Simulação

A freqüência do clock de amostragem possui um ciclo de 0.2 µs, os sinais a serem correlacionados possuem um ciclo de 2.0 µs logo: Nº de amostras $= 2{,}0\mu s / 0{,}2\mu s = 10$ amostras.

A FIGURA 5.1 abaixo mostra os pontos de amostragem dos sinais durante um ciclo dos sinais.

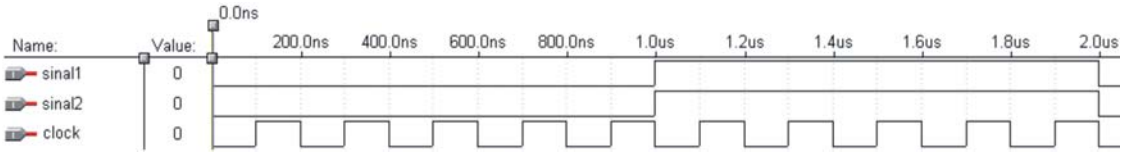

FIGURA 5.1 – Pontos de amostragem utilizados nas simulações.
FONTE: Ambiente de Simulação do Max-Plus II

A simulação do circuito foi feita da seguinte maneira: Foram inseridos dois sinais simulados onde um deles sofreu um atraso de 0 a 2.0µs com passos de 0.2µs. Nota-se que com uma diferença de atraso de 1.0µs ocorre uma defasagem de 180º produzindo assim uma interferência destrutiva entre os sinais.

5.2 Teste da Unidade de Atraso Instrumental

5.2.1 Atraso Inserido com Taxa de Amostragem Fixa

Caso os sinais que estamos utilizando na simulação são idênticos e possuem o mesmo comprimento de onda citado no capitulo anterior os valores obtidos devem ser máximos, porém se configurarmos o correlacionador para corrigir um atraso, que no caso é inexistente, os valores que iremos obter da correlação entre os sinais deverá ser idênticos aos valores obtidos da simulação com o atraso simulado (FIGURA 5.2).

As simulações executadas possuíam dois sinais idênticos e sem atraso entre eles, para cada simulação o correlacionador foi configurado para corrigir um atraso que foi de 0.0µs até 2.0µs de maneira gradativa de 0.2µs em 0.2µs.

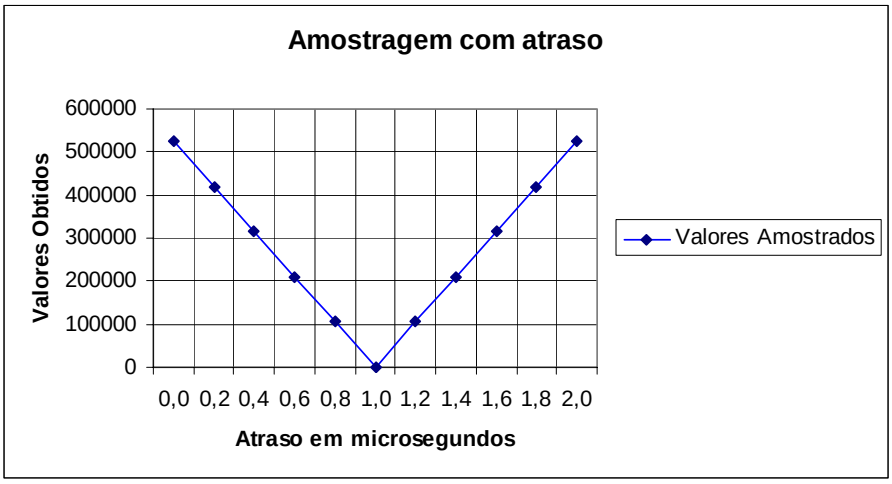

FIGURA 5.2 – Gráfico dos valores obtidos das simulações com atrasos inseridos

5.2.2 Atraso Simulado e Corrigido com Taxa de Amostragem Fixa

Se os sinais inseridos são iguais o valor obtido da integração deve ser máximo, quando foi inserido um sinal com um atraso em relação ao outro este valor caiu gradativamente até o instante em que começou a ocorrer uma sobreposição construtiva entre os sinais. Então, se o atraso for corrigido de maneira correta os sinais estarão sempre em sincronismo, obtendo então um valor máximo na integração FIGURA 5.3.

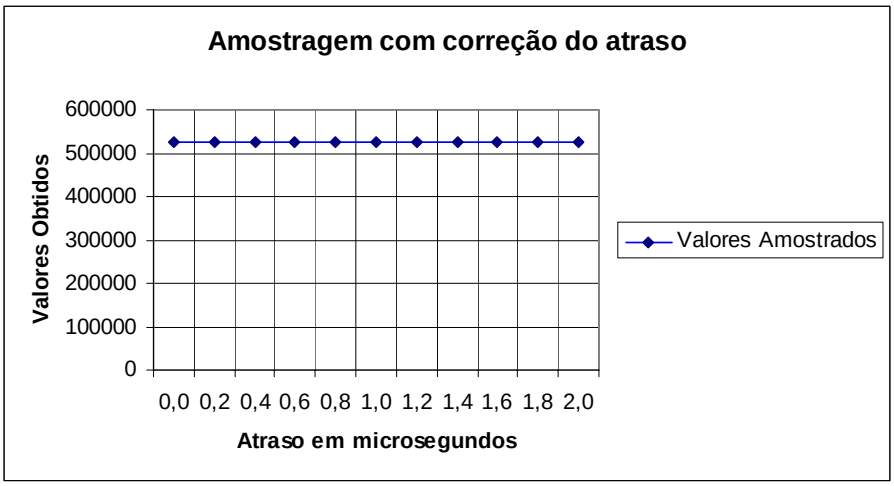

FIGURA 5.3 – Gráfico dos valores obtidos da simulação do atraso simulado e corrigido

5.3 Atraso Simulado com Taxa de Amostragem Fixa

As figuras FIGURA 5.4 a FIGURA 5.14 seguir mostram os atrasos, de 0.0µs até 2µs, de um sinal em relação ao outro, durante um intervalo de tempo de 8.0µs. Após o tempo 105ms (tempo de integração) o circuito correlacionador produziu um valor correspondente ao coeficiente de correlação dos sinais.

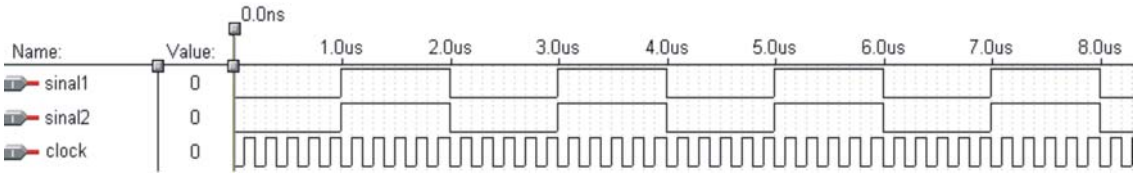

FIGURA 5.4 – Atraso de 0,0 µs com coeficiente de correlação igual a 524288

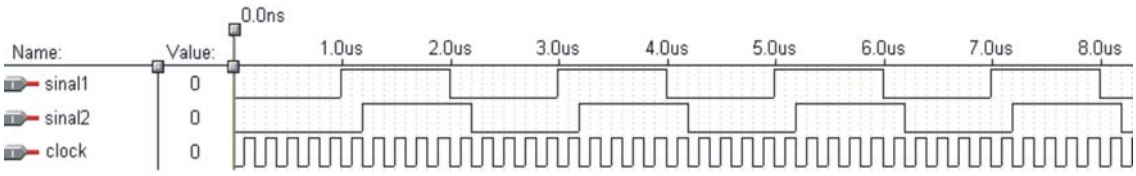

FIGURA 5.5 – Atraso de 0,2 µs com coeficiente de correlação igual a 419431

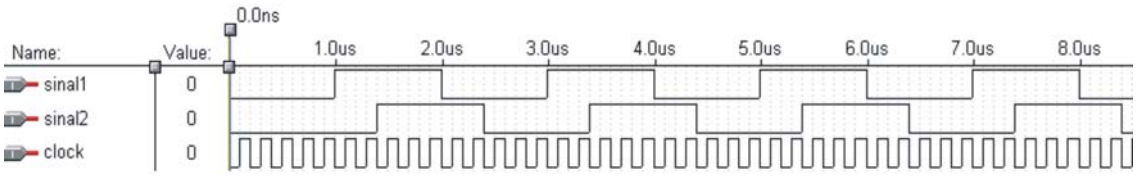

FIGURA 5.6 – Atraso de 0,4 µs com coeficiente de correlação igual a 314573

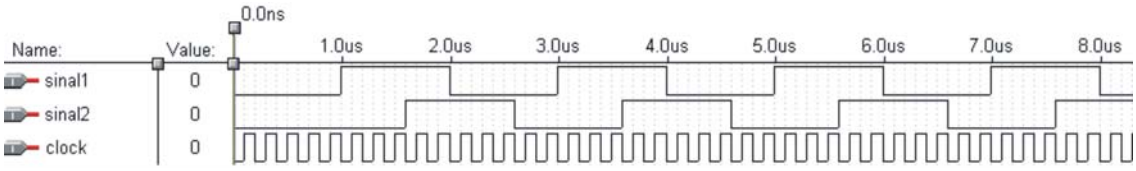

FIGURA 5.7 – Atraso de 0,6 µs com coeficiente de correlação igual a 209715

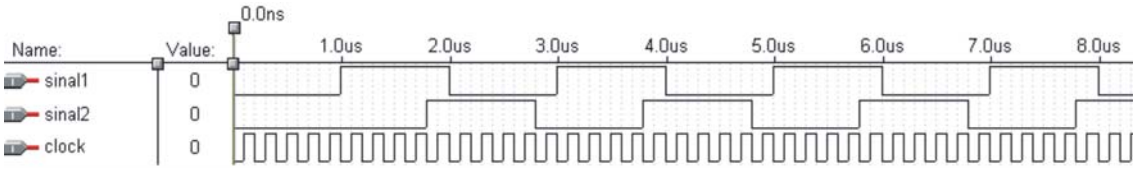

FIGURA 5.8 – Atraso de 0,8 µs com coeficiente de correlação igual a 104857

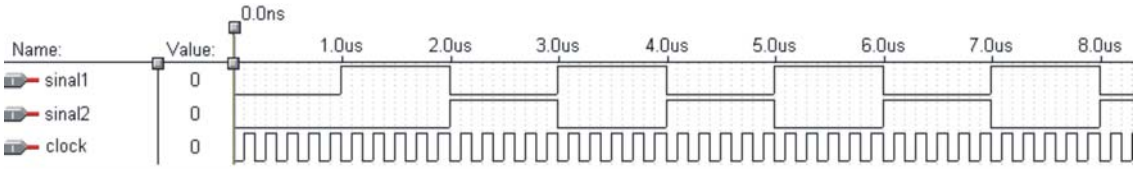

FIGURA 5.9 – Atraso de 1 µs com coeficiente de correlação igual a 0

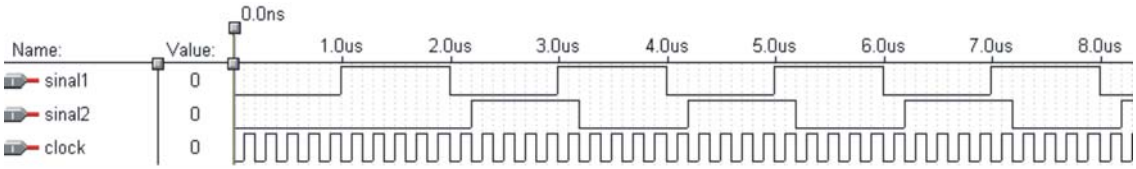

FIGURA 5.10– Atraso de 1,2 µs com coeficiente de correlação igual a 104857

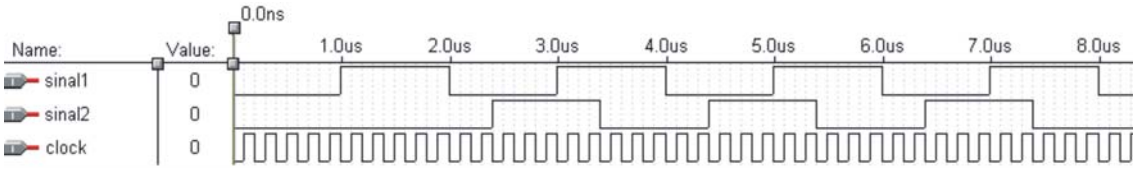

FIGURA 5.11– Atraso de 1,4 µs com coeficiente de correlação igual a 209715

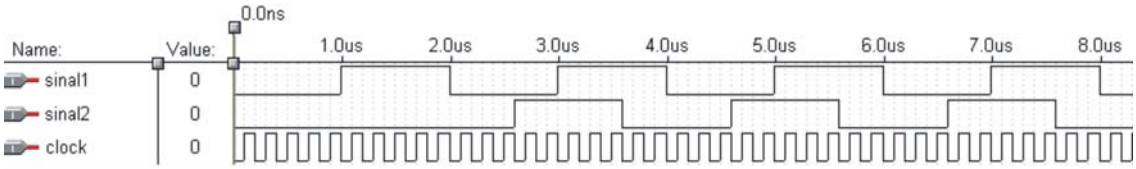

FIGURA 5.12– Atraso de 1,6 µs com coeficiente de correlação igual a 314573

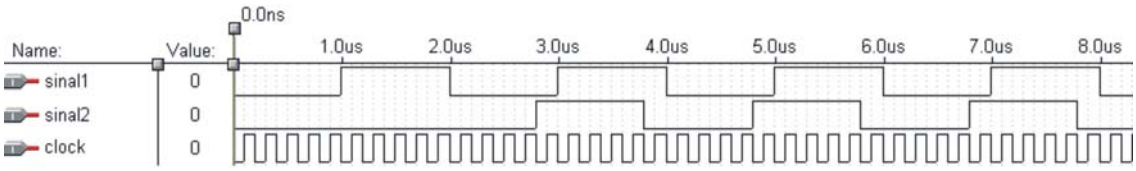

FIGURA 5.13– Atraso de 1,8 µs com coeficiente de correlação igual a 419431

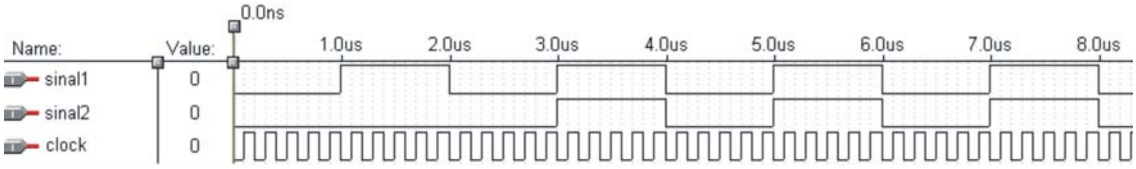

FIGURA 5.14– Atraso de 2 µs com coeficiente de correlação igual a 524288

A FIGURA 5.15 mostra o gráfico obtido dos valores correspondentes às simulações executadas

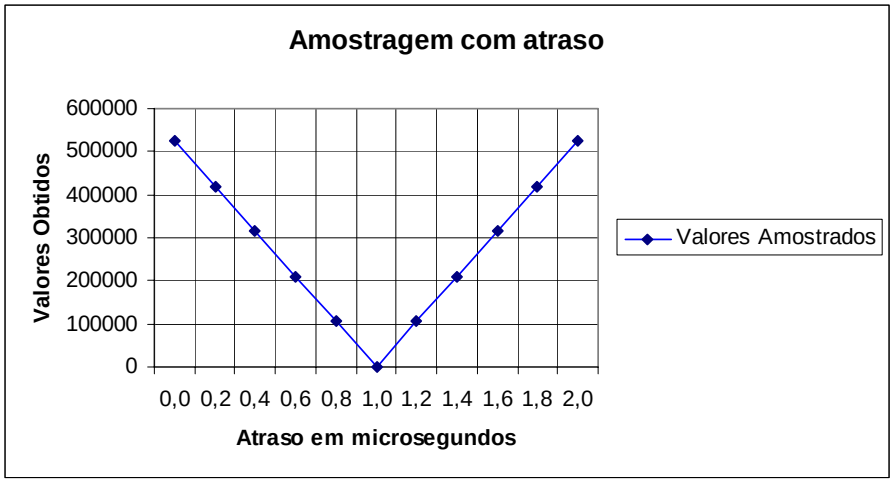

FIGURA 5.15– Gráfico dos valores obtidos das simulações anteriores.

6 SISTEMA DE AQUISIÇÃO

O presente projeto consistiu no desenvolvimento de uma interface física e lógica entre o circuito correlacionador desenvolvido e o computador responsável pela programação e aquisição dos dados, referentes às correlações entre os dois canais do circuito correlacionador.

Para possibilitar a comunicação entre os dispositivos foi realizado um estudo sobre a interface paralela para construção de um circuito eletrônico, que juntamente ao circuito correlacionador, fosse capaz de trocar informações com o computador por meio desta interface.

Com o circuito de comunicação desenvolvido foi criado um software, para o sistema operacional Linux, capaz de programar e manipular o circuito correlacionador por meio da interface paralela. Para esta comunicação ser realizada foi necessária a construção de um cabo capaz de transmitir os dados obtidos pelo circuito para a porta paralela.

Como citado anteriormente o sistema operacional escolhido é o Linux que consiste em um sistema operacional, derivado do UNIX e compatível com o standard POSIX. Não existe um Linux, mas sim uma grande variedade, disponível para todos os objetivos e limitações, cada distribuição é composta por um kernel, aplicações e ferramentas de instalação. Outro ponto relevante á o fato do sistema operacional poder utilizado sob licensa *GPL,* ou seja, o sistema operacional é um software livre.

O preço real de uma escolha de plataforma não é somente o preço do hardware e do sistema operacional, mas o preço do software adicional, serviços e suportes requeridos para criar e operar esta estrutura.

Em relação aos softwares adicionais tais como tcp/ip, servidores httpd, software de administração de rede, bancos de dados SQL, software de backup, e-mail, NFS, firewalls, estes podem ser encontrados em versões de livre distribuição, shareware, freeware ou comerciais.

Tudo em ferramentas básicas de segurança, como SSL, encriptação RSA, firewalls, enpacotadores tcp, sistemas de arquivos encriptados, túneis IP, S/Key aparecem disponíveis para Linux, além de compatibilidade com os padrões Unix de mais de duas décadas de utilização e aperfeiçoamento.

Linux oferece um claro caminho de escalabilidade: PCs Intel, DEC Alpha, Sparcs, multiprocessados. Aplicações desenvolvidas para Linux podem ser portadas com facilidade de/para diversas plataformas: AIX, UX, Sun-OS, Solaris.

O Linux oferece uma extensa lista de hardwares suportados, abrangendo interfaces SCSI, placas multiseriais, motherboards, notebooks, PCMCIA.

Para aumentar a lista de vantagens às distribuições oferecem ainda diversas opções de interfaces gráficas para o administrador, a partir das quais pode ser feita a administração de usuários, discos, arquivos, rede, aplicações. Pode-se optar por interfaces padrão X Windows, Windows 95 ou Windows 98, já disponibilizando o conceito de dados distribuídos: de qualquer ponto do sistema pode-se acessar dados que esteja localizado local ou remotamente.

Ao optar-se pelo Linux abre-se um leque de opções de distribuições, desenvolvedores e aplicações, todos sob o sistema de livre distribuição, que garantem ao usuário a continuidade, o fácil acesso, baixo custo e o suporte ao produto. Linux traz liberdade e independência.

Com a plataforma escolhida e tendo a comunicação entre o computador e o correlacionador pronta iniciou-se o processo de criação das interfaces gráficas do software de controle que foi viabilizada com a utilização do GLADE, um front-end para a biblioteca GTK. Esta biblioteca é um toolkit multi-plataforma e oferece um conjunto completo de widgets, ou seja, objetos gráficos. A utilização do GLADE para construção de interfaces gráficas facilitou a manipulação, em tempo de desenvolvimento, dos objetos da biblioteca GTK.

Após o processo de criação das interfaces foi desenvolvido um modulo capaz de plotar os valores enviados pelo correlacionador ao software de controle. Para tal funcionalidade foi utilizada a ferramenta Gnuplot que é um software de plotagem de dados para varias plataformas como Windows, Unix, Macintosh e algumas outras.

As funcionalidades do Gnuplot foram adicionadas ao software através da inclusão de uma biblioteca, escrita na linguagem C, capaz de fornecer todas as funções do Gnuplot ao código c em que ela foi inserida.

6.1 Ferramentas de Desenvolvimento

Para o presente projeto foram utilizadas basicamente três ferramentas capazes de atender as necessidades criadas para a elaboração da interface gráfica, bem como a plotagem dos dados enviados pelo correlacionador.

As ferramentas utilizadas no projeto foram:

- **Gnuplot** esta ferramenta é parte do projeto GNU e foi criada com o intuito de permitir aos profissionais que desejam plotar valores científicos de maneira simples e eficaz.

- **GTK** é uma biblioteca que contém widgets que são os objetos gráficos como botões, caixa de textos dentre outros objetos.

- **GLADE** esta ferramenta na verdade é um front-end para o GTK, ous seja um software que torna a manipulação da biblioteca de maneira visual e mais simplificada.

6.1.1 GNUPLOT

O Gnuplot é um modulo de plotagem de dados para varias plataformas como Windows, Unix, Macintosh e algumas outras, inicialmente sua finalidade foi permitir a cientistas ou profissionais que desejavam visualizar dados matemáticos de maneira simples. Por esta finalidade o *gnuplot* é um programa voltado para a área exata, possuindo várias ferramentas para a manipulação de dados, estas ferramentas possibilitam de maneira simples a plotagem de dados 2D ou até mesmo 3D.

6.1.1.1 Instalando Gnuplot

Para realizar a instalação no gnuplot em seu sistema operacional deve-se realizar o download do código fonte no site http://www.gnuplot.info/ e proceder os passos descritos para um instalação correta no sistema operacional desejado. Como a distro utilizada no projeto é o Debian a instalação é realizada pela rede através de um terminal digitando o comando 1 para instalar o pacote gnuplot e o comando 2 para instalar o pacote que utiliza os recursos do X11

```
# apt-get install gnuplot                    (1)

# apt-get install gnuplot-x11                (2)
```

6.1.1.2 Executando Gnuplot

Para executar o Gnuplot basta digitar no console o comando gnuplot que o software entrará em execução (FIGURA 6.1 **– Prompt de Comando após executar comando gnuplot**), para executar os comandos de plotagens de gráficos basta digitá-los no prompt respeitando os parâmetros necessários de cada função. Juntamente com a instalação do Gnuplot existe um arquivo de help completo e bem descritivo sobre cada parâmetro das funções plotagem por ele realizada. Além das funcionalidades de plotagem o software também oferece a possibilidade de salvar as imagens geradas em um arquivo do tipo png.

Todas as funções de plotagem do gnuplot abrem obrigatoriamente um janela para mostrar os dados da função executada como mostra a FIGURA 6.2 **– Gráfico gerado pela função plot sin(x).** .

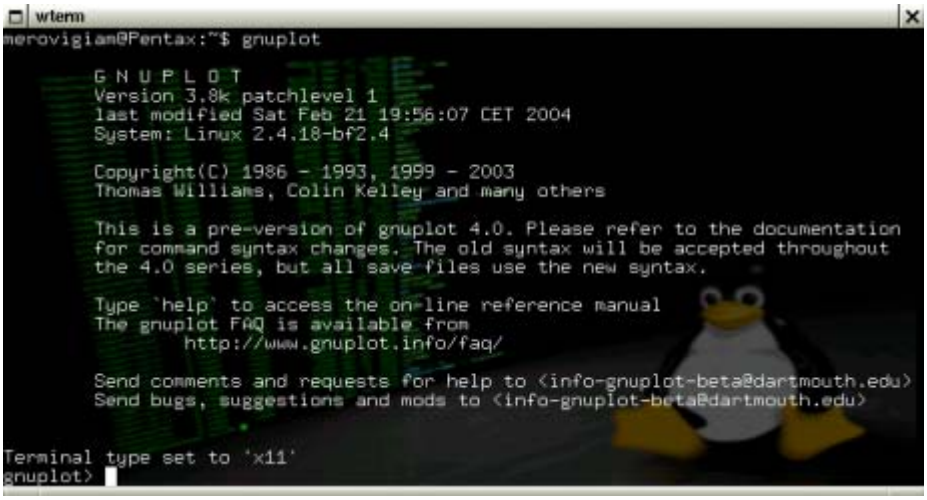

FIGURA 6.1 – Prompt de Comando após executar comando gnuplot

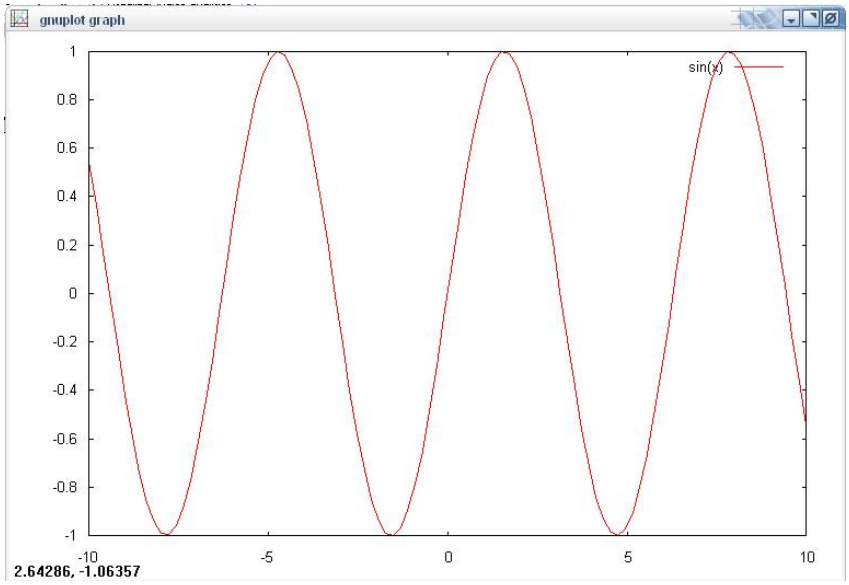

FIGURA 6.2 – Gráfico gerado pela função plot sin(x).

6.1.1.3 Integrando o Gnuplot ao Código em C

Para integrarmos o Gnuplot a um programa na linguagem C é necessária à inclusão de uma biblioteca capaz de fornecer acesso às funcionalidades do Gnuplot, para tal deve-se incluir a biblioteca gnuplot_i.c nas bibliotecas do programa, esta biblioteca permite executar os comandos do Gnuplot de maneira transparente dentro do código fonte do modulo de aquisição de dados do software do correlacionador.

Os passos para utilizar as funções da biblioteca dentro do código fonte são poucos, porém de extrema importância para o funcionamento correto do software. Primeiramente deve-se inicializar uma variável do tipo *gnuplot_ctr* (3), esta variável tem o funcionamento semelhante ao de um handle e deve ser passado como parâmetro em todas as funções da biblioteca.

```
gnuplot_ctrl *screenplot;                    (3)
```

Com esta inicialização feita todas as outras funções como *gnuplot_setstyle(screenplot, "lines");* ou *gnuplot_cmd(screenplot,"set grid");* podem ser utilizadas. O código fonte a seguir mostra um exemplo de como deve ser realizado a plotagem de um vetor double com valores aleatórios.

As linhas numeradas no ANEXO 2 CODIGO 3 nos mostram que primeiramente devemos inicializar o handle através da função gnuplot_init(); ,posteriormente seta-se o tipo da linha do gráfico a ser plotado, no caso linhas podendo ainda ser pontos (dots), a linha de código a seguir possui a função mais interessante da biblioteca gnuplot_i.c ela é capaz de executar qualquer função do Gnuplot que não tenha sido definida dentro da biblioteca.

A função gnuplot_plot_xy é responsável por plotar os pontos com os valores do vetor xplot e yplot juntamente com as propriedades setadas anteriormente na variável screenplot.

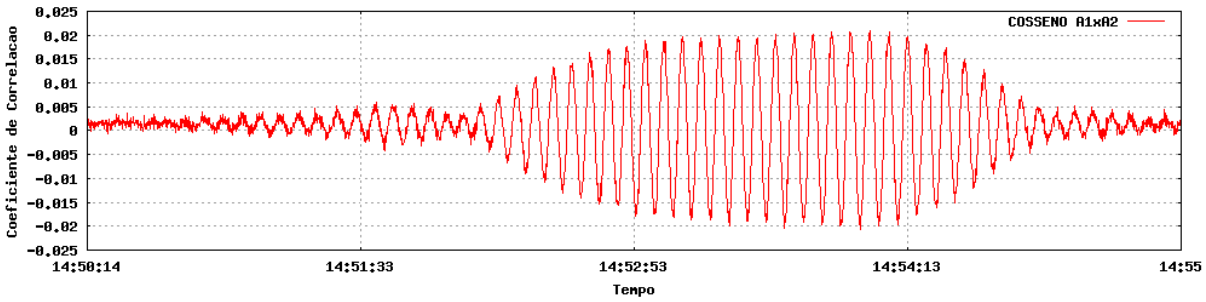

FIGURA 6.3 – Franja produzida pela integração dos sinais e gerada pelo software de aquisição

6.1.2 GTK

Para manipular todas as funções do software foi desenvolvida uma interface gráfica com a ajuda da biblioteca GTK+ (GIMP Tool Kit), é um toolkit multi-plataforma que oferece um conjunto completo de widgets, ou seja, objetos gráficos.

A criação inicial do GTK+ foi para utilização dos programas gráficos do GIMP, é o primeiro das mais populares ferramentas de widgets para Sistemas de janelas X, pretendido para criar interfaces gráficas para usuários.

Licenciado pela GNU GPL, GTK+ é um software livre e é parte integrante do projeto GNU. GTK+ é escrita em C e seu design é orientado a objeto. Existem bindings para construção de programas GTK+ em C++, Perl, Java, Python e PHP.

O toolkit oferecido pelo GTK+ é configurável pelo usuário e oferece diferentes opções de engines. Há engines emulando a aparência de outros populares toolkits ou plataformas como Windows 95, MAC OS entre outros.

O ambiente GNOME usa GTK+ como base do seu sistema, vários programas para GNOME usam GTK como seus toolkits. Aplicações GNOME não são as únicas a usá-la, alguns programas GTK+ podem rodar em outros sistemas de desktop, como KDE ou XFce. Em dispositvos móveis, o GTK+ é usado como base de ambientes como o GPE Palmtop e o Maemo. GTK+ também pode ser usado no Microsoft Windows.

GTK+ é baseado em três bibliotecas desenvolvidas pela própria equipe do GTK+:

- **GLib** é a biblioteca de baixo nível, é o núcleo que dá forma à base do GTK+, esta fornece as estruturas de dados em C.
- **Pango** é uma biblioteca para o layout e a renderização do texto.
- **A biblioteca** ATK fornece um conjunto de funções de interfaces para acesso ao do objeto. Suportando as relações do ATK, uma aplicação ou um toolkit pode ser usado com ferramentas para construção de softwares.

GTK+ foi projetado para ser utilizado por várias linguagens de programação, não somente C/C++, especialmente em combinação com o construtor de interfaces Glade que fornece um método eficaz de desenvolvimento da aplicação.

6.1.2.1 Modelo de programa utilizando GTK

Os programas que possuem interfaces simples possuem o código fonte relativamente simples também como podemos notar no código fonte abaixo. A execução código fonte ANEXO 2 CODIGO 1 produz a interface gráfica (FIGURA 6.3) com apenas uma janela e com o titulo da mesma com o valor “Alo Mundo”.

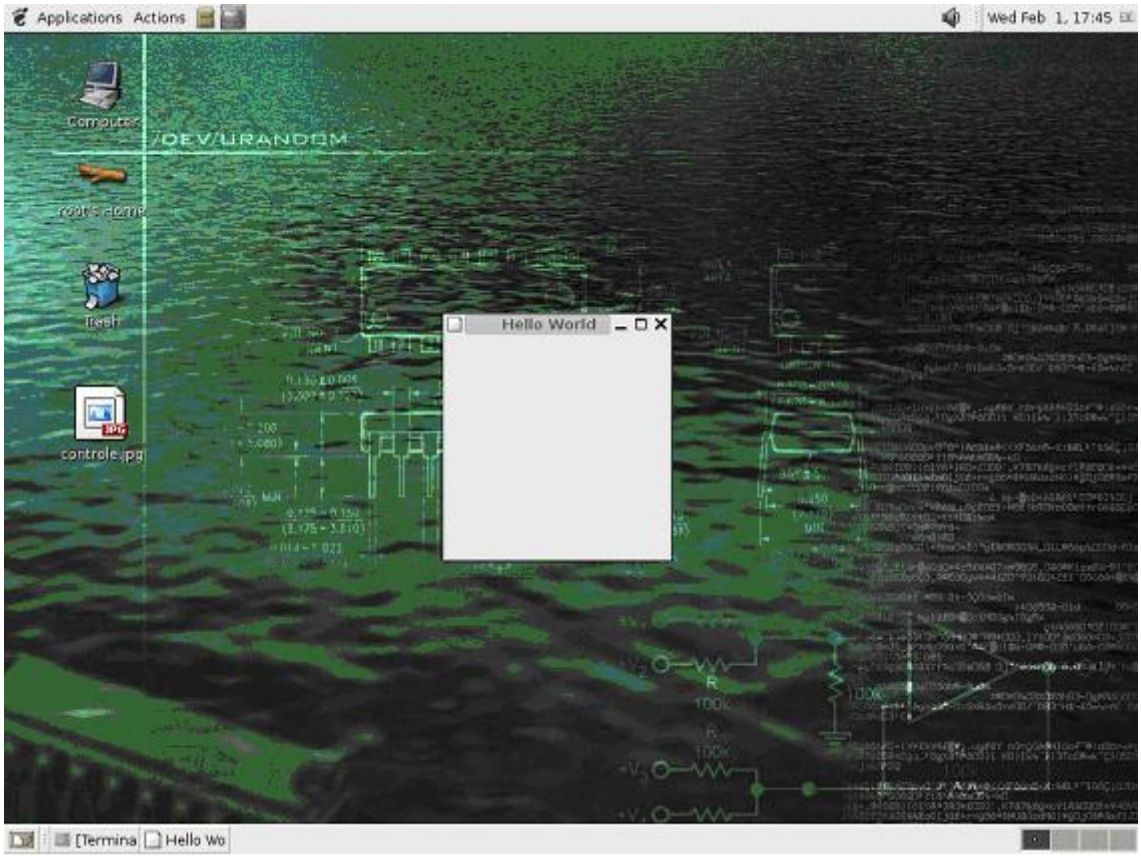

FIGURA 6.4 – Interface gráfica produzida pelo programa Alo Mundo em execução.

O código fonte deste programa é relativamente simples onde a variável do tipo GtkWidget consiste em um ponteiro que armazenará as propriedades do objeto que no caso incide em um janela, posteriormente as variáveis internas e necessárias ao funcionamento da interface criada pelo GTK é inicializada através da função gtk_init(&argc, &argv). Após esta inicialização o ponteiro recebe os valores referentes a janela padrão criada pela função anterior, este padrão é alterado pela função seguinte gtk_window_set_title(GTK_WINDOW (janela), "Alo Mundo"), esta linha no código é responsável por setar o título da janela em “Alo Mundo”.

Posteriormente a configuração da janela é feita uma chamada a função gtk_widget_show(janela), responsável por mostrar a janela no vídeo, e em seguida a função gtk_main(), esta ultima função é a mais importante pois é esta que mantém o programa em funcionamento. Esta função de execução só é interrompida caso ocorra a chamada da função gtk_main_quit() ou o Sistema Operacional encerre o processo.

6.1.3 GLADE como construtor de interfaces

Como demonstrado no capítulo anterior podemos digitar todo o código da interface desejada, porém caso seja uma interface com muitos recursos o código fonte tende a tornar-se complexo e de difícil organização por parte de quem esta gerenciando o projeto.

Para resolver este problema deve-se utilizar o GLADE que consiste em um construtor visual para os usuários GTK+ e GNOME e liberado sob a licença de GNU GPL.

As interfaces desenvolvidas no Glade são mantidas em arquivo XML, e usando a biblioteca do libglade os objetos criados na interface podem ser carregados por aplicações dinâmicamente quando haja necessidade.

Usando o libglade, os arquivos do Glade XML podem ser utilizados em diversas linguagens de programação como C, C++, Java, Perl, Python, C #.

O GLADE possui uma interface gráfica de acordo com a FIGURA 6.5 com as janelas principais destacadas em vermelho.

Ao se que ao trabalhar em um ambiente de desenvolvimento visual como o mostrado a construção de interfaces complexas se torna mais simples e de fácil visualização, já que no modo texto era necessário compilar o código para visualizar a interface que estava sendo construída.

Para criar-se uma interface no neste ambiente basta adicionar o componente window na paleta de componentes e adicionar os objetos a serem utilizados na interface.

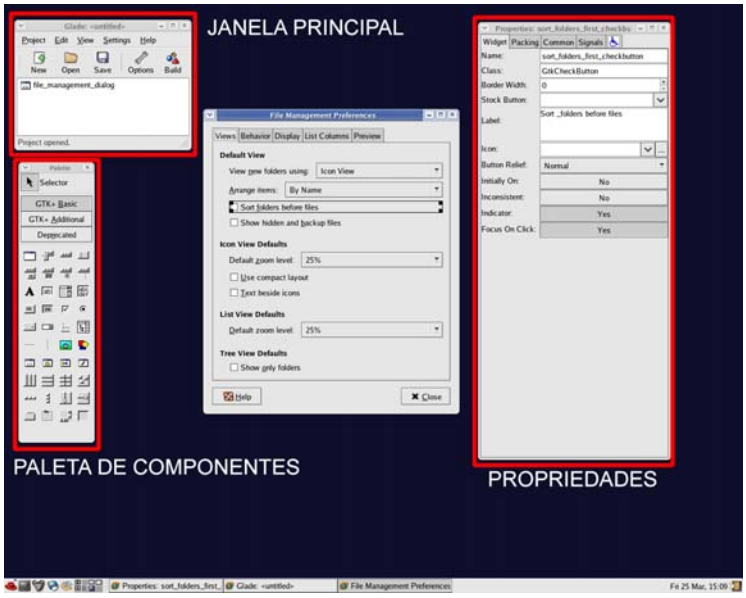

FIGURA 6.5 – Screenshot do ambiente GLADE.

FONTE: http://glade.gnome.org/screenshots.html

6.2 Interface de Comunicação kit-computador

Uma vez que o circuito de comunicação já esta definido (CAPITULO 1), foi realizado um estudo prévio do funcionamento da interface paralela para tornar possível a comunicação entre o computador e o kit FPGA utilizado no projeto.

O desenvolvimento da interface de comunicação consistiu no desenvolvimento de rotinas, capazes de enviar e receber informações por meio da porta paralela, e um link físico entre o chip FLEX10K, utilizado para síntese do circuito correlacionador, e a porta paralela.

6.2.1 Funcionamento da Interface Paralela

A porta paralela é uma interface de comunicação entre o computador e um periférico, lançada inicialmente pela IBM o objetivo principal era fornecer uma conexão, através desta interface, a um periférico como uma impressora, mas posteriormente outros periféricos utilizaram-se desta interface para enviar e receber dados.

Antes de descrever as configurações da interface deve-se lembrar que ela consiste em um conector do tipo DB25 e possui a pinagem descrita na FIGURA 6.6

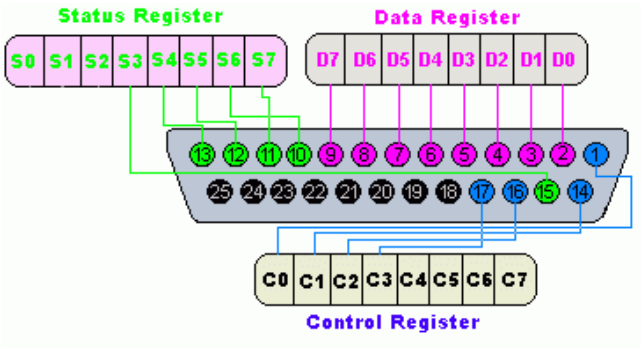

FIGURA 6.6 – Descrição dos pinos da interface paralela.

A porta paralela SPP (Standard Parallel Port) pode chegar a uma taxa de transmissão de dados a 150KB/s. Comunica-se com a CPU utilizando um BUS de dados de 8 bits. Para a transmissão de dados entre periféricos são usado 4 bits por vez.

A porta avançada EPP (Enhanced Parallel Port) chega a atingir uma taxa de transferência de 2 MB/s. Para atingir essa velocidade, é necessária a utilização de um cabo especial. Comunica-se com a CPU utilizando um BUS de dados de 32 bits e para a transmissão de dados entre periféricos são usado 8 bits por vez.

A porta avançada ECP (Enhanced Capabilities Port) tem as mesmas características que a EPP, porém, utiliza DMA (acesso direto à memória), sem a necessidade do uso do processador, para a transferência de dados. Utiliza também um buffer FIFO de 16 bytes.

Para uma maior facilidade no manuseio do kit FPGA durante a execução do projeto a porta paralela foi mantida na configuração EPP, caso a configuração da porta fosse modificada o software de programação MAX PLUS II não é capaz de reconhecer o kit FPGA, pois o mesmo é programado por meio desta interface também.

O computador nomeia a Porta Paralela, chamando-a de LPT1, porém seu endereço físico para acesso de qualquer software que deseje utilizar a interface paralela é o 378h para escrita e 379h para leitura.

Basicamente para manipular a interface paralela, em um sistema operacional Linux, por meio de um software deve-se abrir o seu endereço físico e manipular a informação, tanto para escrita quanto para leitura.

O código fonte abaixo exemplifica o que deve ter um programa em C para acesso a interface paralela. ANEXO 2 CODIGO2

Como se pode perceber a função ioperm juntamente com o endereço da porta abre o dispositivo para a escrita e leitura realizadas posteriormente pelas funções outb e inb.

6.2.2 Conexão da Interface Paralela ao FPGA

O kit FPGA adquirido pelo projeto não possuía nenhuma interface de acesso ao chip Flex10K (FIGURA 6.7), para solucionar o problema foi retirado um conector IDE macho de uma placa mãe e soldado na furação oferecida ao lado direito do chip (FIGURA 6.8).

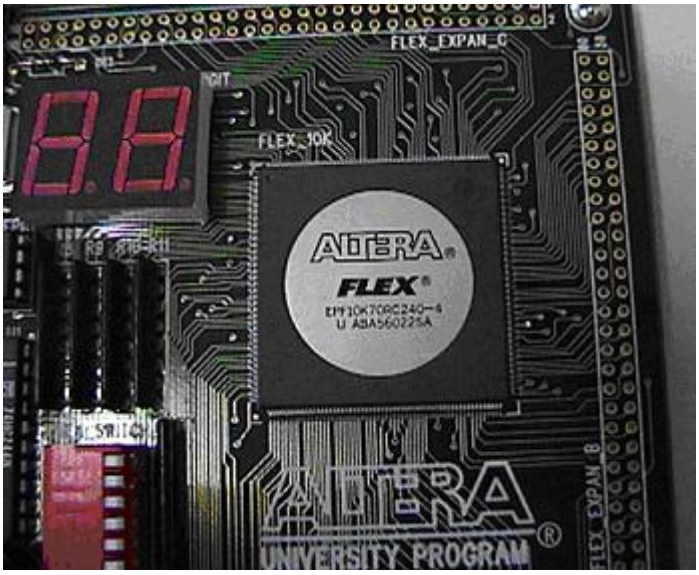

FIGURA 6.7 – Chip Flex 10k utilizado na gravação e execução do circuito correlacionador.

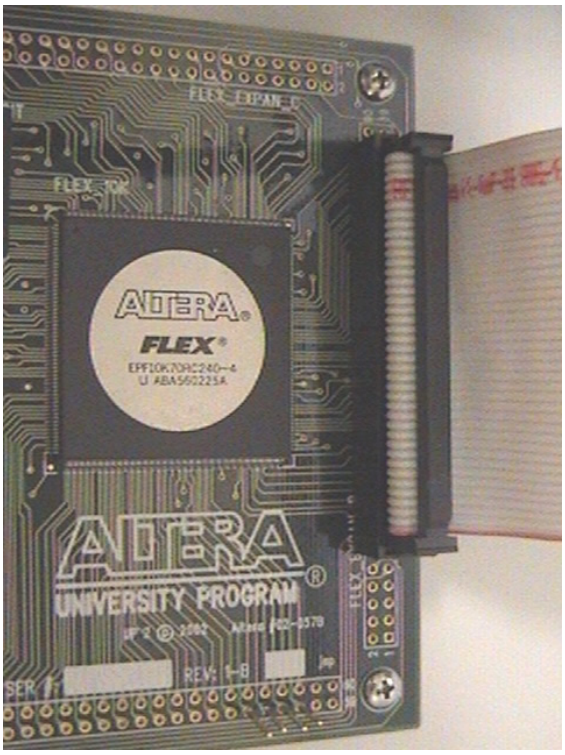

FIGURA 6.8 – Foto do conector soldado e o cabo IDE conectado.

Posteriormente foi utilizado um cabo IDE de 40 vias, retirando de uma das pontas a conexão IDE e substituindo, através de um processo de solda dos fios, por um conector DB25 macho como mostra a FIGURA 6.9 e FIGURA 6.10

FIGURA 6.9 – Cabo IDE adaptado para uso com o kit FPGA

Desta maneira a conexão entre o kit FPGA e o computador pode ser realizada com sucesso, para isto deve-se levar em consideração o fato do cabo flat ser de baixa perda e suportar uma taxa de transmissão de dados superior a da taxa de transmissão da interface paralela.

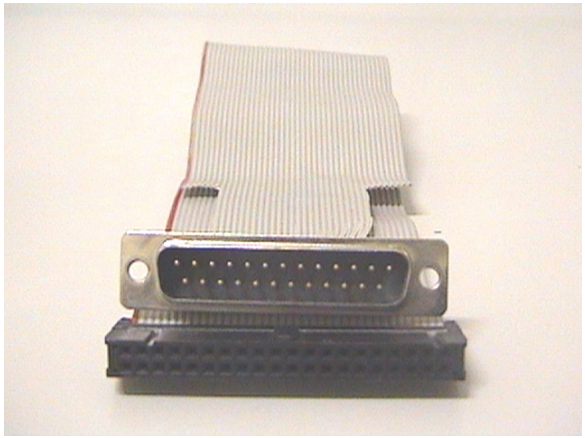

FIGURA 6.10 – Cabo Flat com os dois conectores utilizados.

Para termos certeza do funcionamento correto do cabo criado foi gravado dois circuitos simples, primeiramente uma porta e onde sua saída foi direcionada para o pino IDE ligado ao pino DB25 10 e suas entradas nos pinos 2 e 3.

O primeiro teste foi realizado da seguinte maneira: foi enviado para porta 0378h um determinado valor, de acordo com as entradas da porta E (Tabela 6.1), e verificado se o valor de saída era correto. Todas possíveis entradas foram testadas e produziram uma saída correta.

Tabela 6.1 – Tabela verdade de uma porta E

Entrada	**Entrada**	**Saída**
0	**0**	**0**
0	**1**	**0**
1	**0**	**0**
1	**1**	**1**

Para o segundo teste foi inserido um contador binário de 4 bits e verificado o valor de saída a cada pulso de clock, setando um pino em 1 posteriormente em 0, realizado pelo software demonstrado no capitulo anterior.

6.3 Software de Controle e Aquisição de Dados

O sistema de controle e aquisição do correlacionador desenvolvido é capaz de programar o circuito com o atraso previamente calculado e adquirir as medidas de similaridade, obtidas pelo processo de integração durante 100ms, entre os canais do circuito.

Para o controle do circuito uma interface gráfica foi desenvolvida com a finalidade de setar o atraso entre os sinais e de aquisição dos valores obtidos. Após o termino do processo de aquisição as integrações obtidas são disponibilizadas para visualização ao usuário e, caso necessário, plotagem das franjas obtidas durante o período de observação escolhido.

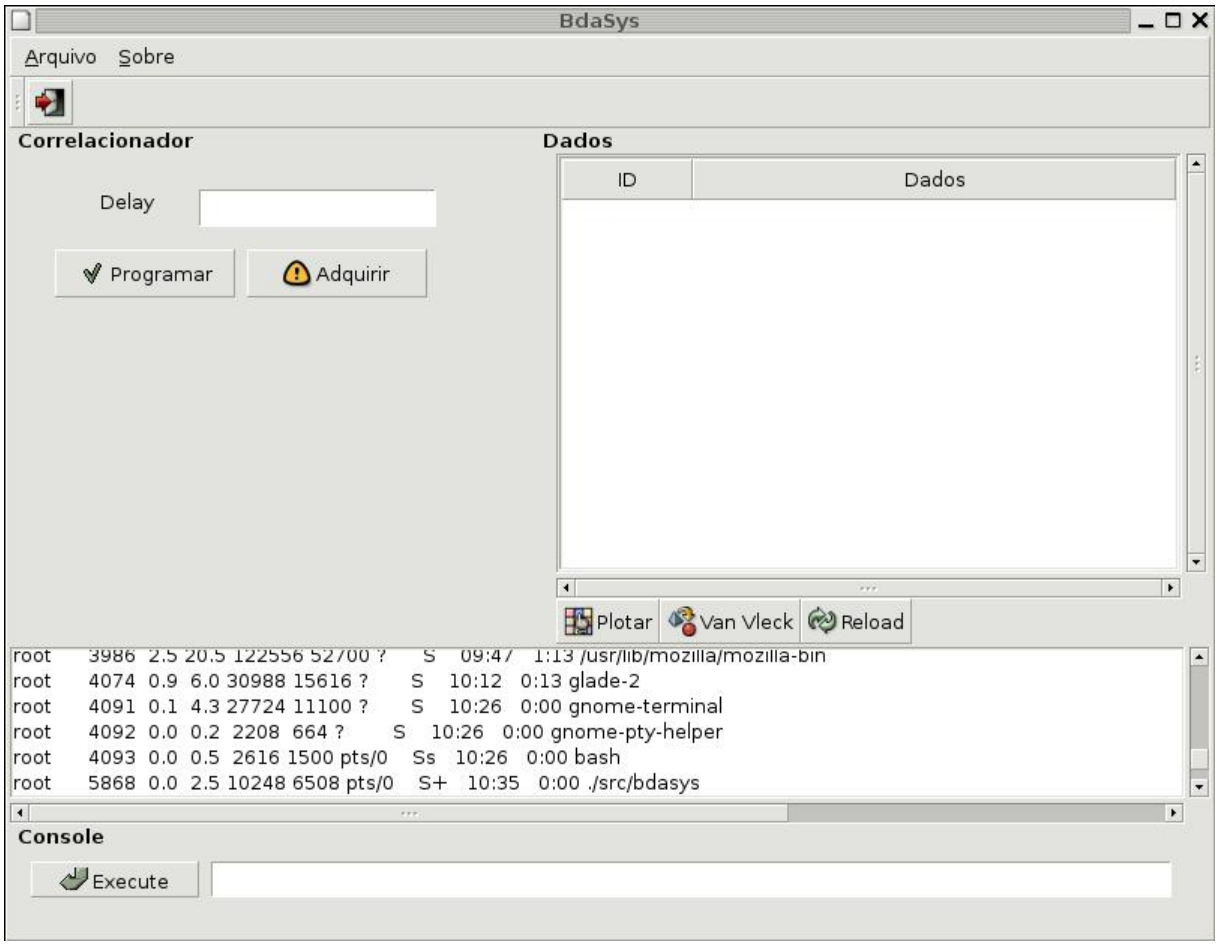

FIGURA 6.11 – Screenshot do Software de controle e aquisição de dados.

O como ilustrado pela FIGURA 6.10 a manipulação dos atributos referentes aos dados do correlacionador são realizados de maneira clara e objetiva, o usuário deve informar a diferença de fase entre os sinais e clicar no botão Programar FIGURA 6.12.

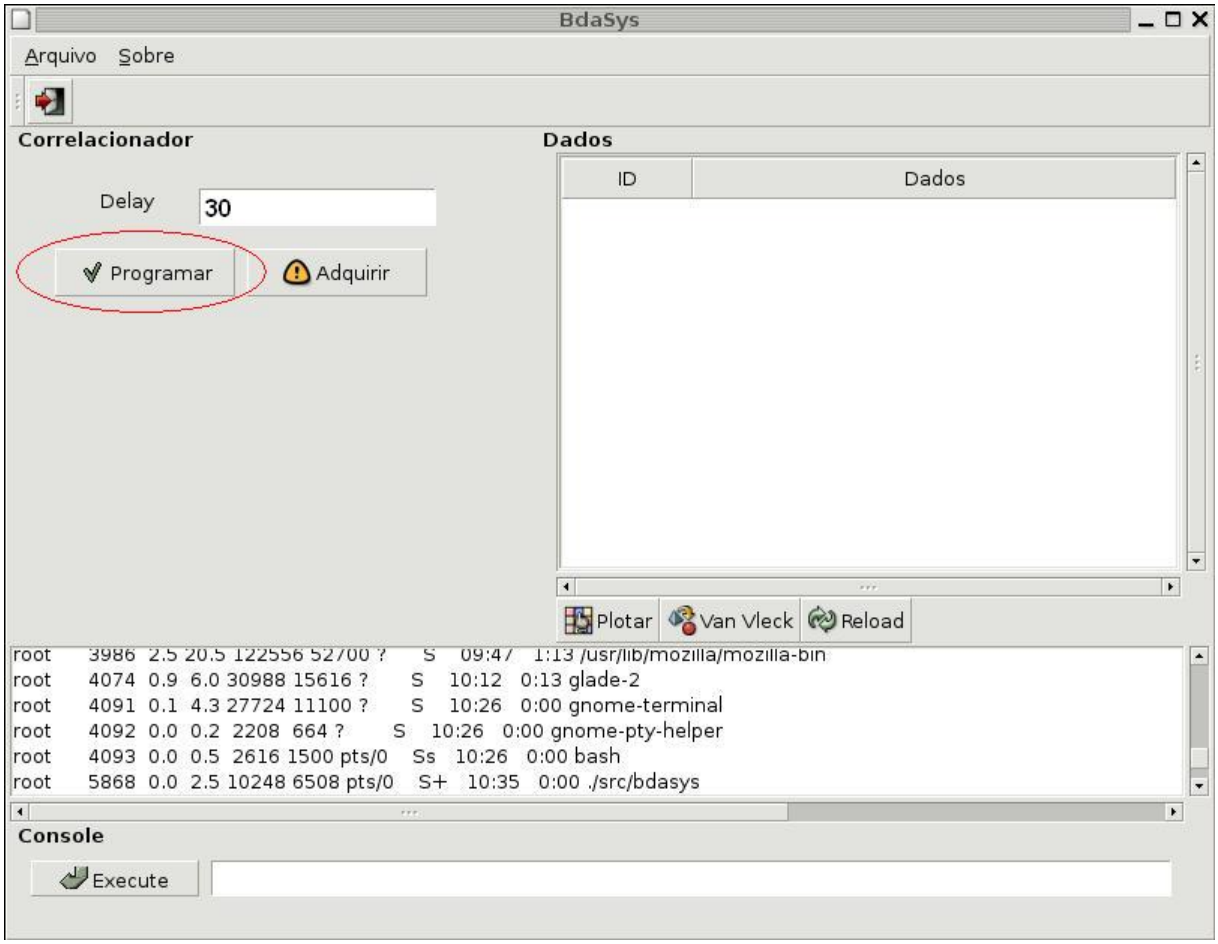

FIGURA 6.12 – Programando o correlacionador.

Enquanto o correlacionador esta integrando os sinais o software armazena os coeficientes obtidos no arquivo first.dat, no diretório corrente, este arquivo é atualizado a cada período de integração e pode ser ter seus dados visualizados clicando no botão Reload, tendo os valores carregados na interface para plotarmos os dados obtidos basta clicar no botão plotar que uma nova janela irá mostrar a franja obtida pelo correlacionador durante o tempo de observação. Porém os dados obtidos pelo correlacionador são dados brutos, ou seja, porduzem uma onda triangular com um intervalo de dados que pode ir de 0 a 525000, para corrigir os valores e visualizar o sinal de maneira correta deve-se aplicar a correção de Van Vleck.

Enquanto o usuário não programar novamente o correlacionador, um arquivo, de nome first.dat salvo no diretório corrente, está sendo atualizado a cada integração com os dados obtidos. Estes dados, caso necessário, são disponibilizados para plotagem, porém os valores

obtidos são dados brutos, ou seja, representam apenas a quantização da similaridade entre os canais do correlacionador. Estes valores quando plotados formam uma onda de formato triangular com valores entre 0 e 525000.

Esta correção é descrita pela formula (6.1), porém o valor a ser obtido, em relação as aquisições é justamente o valor de ρ e isolando a variável temos a seguinte equação (6.2). O resultado desta correção faz produz uma onda senoidal em função da onda triangular produzida pelo correlacionador.

$$\rho^2 = \frac{2}{\pi}\sin^{-1}\rho \tag{6.1}$$

$$\rho = \sin\rho^2\frac{\pi}{2} \tag{6.2}$$

Para exemplificar o funcionamento da correção a FIGURA 6.13 mostra a franja triangular similar a onda produzida pelo correlacionador e a FIGURA 6.14 mostra como fica os valores após a correção e normalização, nota-se que os valores de saída estão entre o intervalo de -1 até 1.

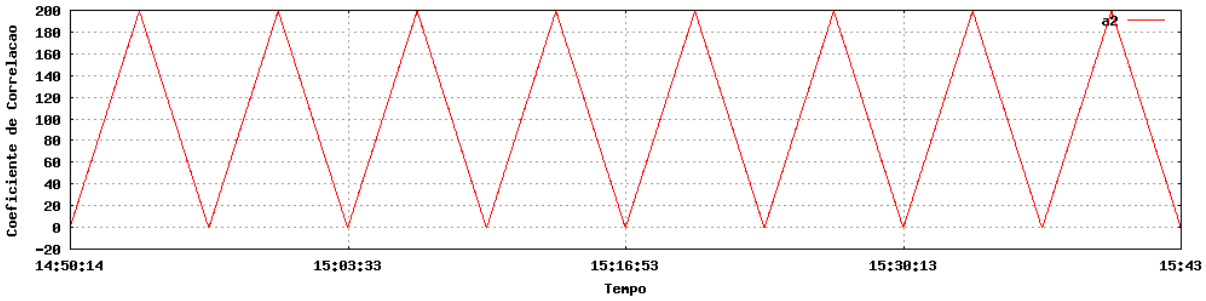

FIGURA 6.13 – Franja de teste simulando os valores produzidos pelo correlacionador.

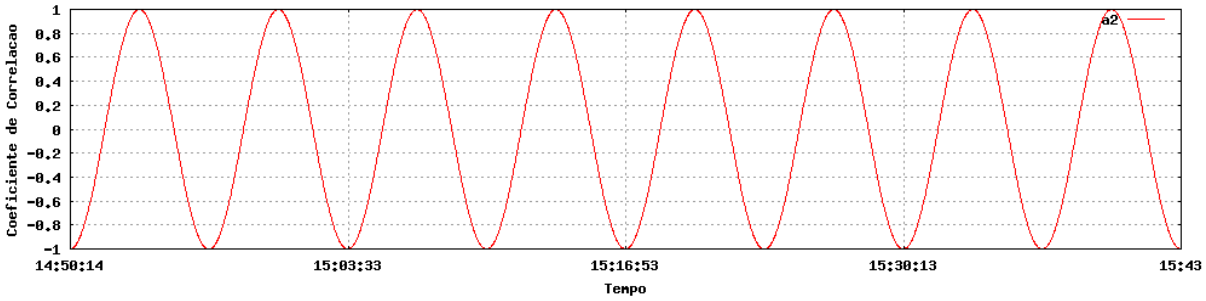

FIGURA 6.14 – Franja obtida após correção de Van Vleck.

7 CONSIDERAÇÕES FINAIS

O trabalho apresentado consistiu no esforço da construção de um correlacionador digital de dois canais, com um bit de quantização, que possui a capacidade de medir o coeficiente de correlação entre os sinais em um tempo pré-determinado de 100 ms. Esta etapa do trabalho obteve sucesso devido aos resultados obtidos, nas simulações executadas, que foram coerentes e precisos.

Uma das vantagens deste circuito correlacionador desenvolvido é a capacidade do circuito ser independente da taxa de amostragem, pois esta depende apenas do kit FPGA onde o circuito seria sintetizado. Esta independência nos da a capacidade de escolhermos a velocidade com que desejamos medir o coeficiente de correlação entre os sinais provenientes de uma fonte emissora. Esta velocidade provém do kit FPGA que estamos utilizando para sintetizar o circuito correlacionador, podendo chegar à ordem de 10^9. Outra vantagem do circuito correlacionador vem da estabilidade do circuito quando este está sintetizado em um Kit FPGA.

Para o tornar-se possível à comunicação entre o kit FPGA e o computador foi projetado e desenvolvido um link físico capaz de suportar a taxa de transferência entre o chip, utilizado na síntese do circuito, e a interface paralela do computador. A comunicação entre os hardwares é realizada através de um software gráfico desenvolvido com a capacidade de informar ao correlacionador o atraso entre os sinais e de aquisição das integrações obtidas, outra característica importante é fornecer ao usuário a opção de visualizar o conjunto de dados obtidos em um gráfico.

Para atestarmos o funcionamento correto do projeto deve-se lembrar que não foram realizados testes com sinais reais, ou seja, sinais provenientes de um par de radiotelescópios, após os testes necessários e validações este circuito correlacionador poderá ser aplicado em futuras aplicações em Radio-Interferometria.

Sobre as otimizações possíveis no circuito deve-se levar em consideração que o mesmo pode ter o número de canais aumentados, dependendo da capacidade do kit FPGA utilizado, podendo assim, suportar um número maior de radiotelescópios.

8 REFERÊNCIAS BIBLIOGRÁFICAS

Crutcher, R.M. Imaging the Universe at Radio Wavelengths. **IEEE Computational Science & Engineering**, vol. 1, n. 2, pp. 39-49, 1994.

Faria, C. **Reconstrução de Imagens Solares a Partir de Observações Radiointerferométricas com Alta Resolução Temporal e Espacial**. São José dos Campos. Proposta de tese de doutorado em Computação Aplicada - Instituto Nacional de Pesquisas Espaciais, 2002.

Hinderks, James. **The Theory, Design and Implementation of a High-Speed Digital Correlator**. Princeton University, 1999.

Kellermann, K. I.; Moran, J. M**.** The development of high-resolution imaging in radio astronomy. **Annual Review of Astronomy and Astrophysics**, Vol. 39, p. 457-509. 2001.

Nakajima, et al. The Nobeyama radioheliograph. **Proceedings of IEEE**, Vol. 82, no. 5, p. 705-713, 1994.

Pertence Junior, Antonio, **Eletrônica analógica: amplificadores operacionais e filtros ativos: teoria, projetos, aplicações e laboratório** / Antonio Pertence Júnior. Porto Alegre: Bookman,2003.

Ramesh, R.; Subramanian, K. R.; Sundara Rajan; M. S.; Sastry, CH. V. The Gauribidanur Radioheliograph. **Solar Physics,** v. 181, n. 2, p. 439-453, 1998.Berlin, 1986

Sawant, H. S. AND BDA TEAM. First light from Prototype Brazilian Decimetric Array, **Advances in Space research**, 2004.

Sawant, H. S. et al. **Brazilian Decimetric Array: O Primeiro Interferômetro Decimétrico da América Latina**. (INPE-13051-RPQ/252), 71 p. 2005.

Thompson, A. R., Moran, J. M. AND Swenson, G.W. **Interferometry and Synthesis in Radio Astronomy**. Wiley, New York, 1994.

Van Vleck, J. H.; MIDDLETON, D. The Spectrum of Clipped Noise. **Proc. IEEE**, v. 54, n. 1, p. 2, 1966.

Wohlonben, R., Mattes, H. e Krichbaum, Th. **Interferometry in Radioastronomy and Radar Techniques** Dordrecht, Netherlands: Kluwer, 1991, et al.

9 ANEXO 1 – SIMULAÇÕES

9.1 Simulações com sinais variantes e Taxa de Amostragem Fixa

As simulações a seguir mostram as correlações com vários comprimentos de onda, estas simulações foram executadas com o clock de amostragem de 0,2µs. Os sinais que foram simulados possuem comprimentos de onda que vão desde o próprio comprimento do sinal de clock utilizado até um comprimento de onda 8 vezes maior que o sinal de clock.

As simulações a seguir foram feitas de maneira semelhante à simulação anterior (capitulo 2.4.4) onde um dos sinais foi atrasado de maneira gradativa de 0.2µs em 0.2µs de -2.0µs até 2.0µs, porém desta vez o ponto onde ocorre uma defasagem de 180º em relação ao outro sinal que gera uma interferência destrutiva entre os sinais não foi respeitado, podendo ocasionar assim várias sobreposições que devem ser medidas pelo correlacionador com os mesmos valores obtidos antes que ocorra a sobreposição.

9.1.1 Simulação 1

Esta simulação foi executada da seguinte maneira: Foi inserido um sinal com freqüência 8 vezes menor que o clock de amostragem de 5 Mhz, após 105ms foi lido o coeficiente de correlação calculado pelo correlacionador e foi realizada nova simulação porém desta vez com um dos sinais deslocados em 0.2µs. Este processo foi realizado com os sinais variando em um intervalo de tempo de -2.0µs a 2.0µs.

Os valores obtidos pelas simulações estão na TABELA 9.1 e estão plotados na FIGURA 9.1

Tabela 9.1 – Sinais com comprimento de onda 8 vezes maior que o sinal de clock.

Atraso em µs	Valor Obtido
-2	262145
-1,8	393217
-1,6	524288
-1,4	393217
-1,2	262145
-1	131073
-0,8	0
-0,6	131073
-0,4	262145
-0,2	393217
0	524288
0,2	393217
0,4	262145
0,6	131073
0,8	0
1	131073
1,2	262145
1,4	393217
1,6	524288
1,8	393217
2	262145

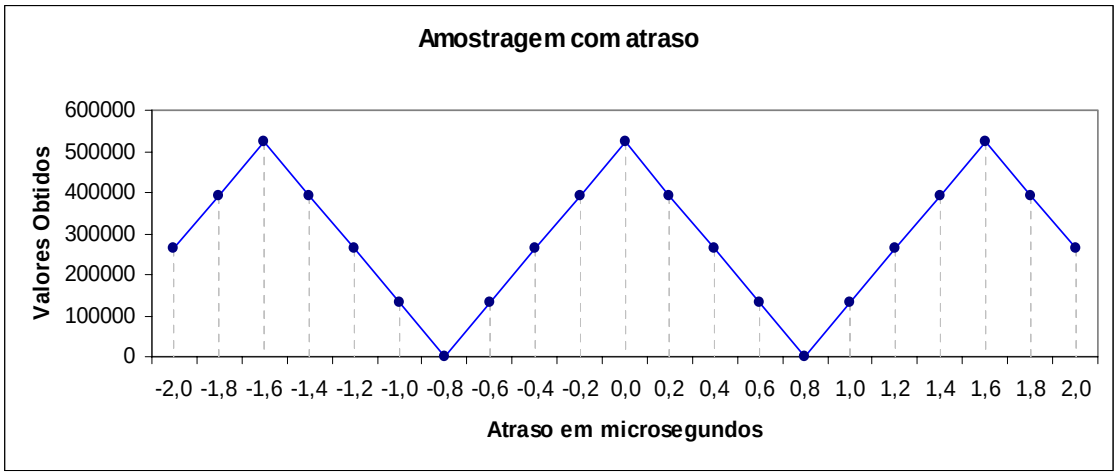

FIGURA 9.1– Gráfico obtido dos valores da tabela 9.1

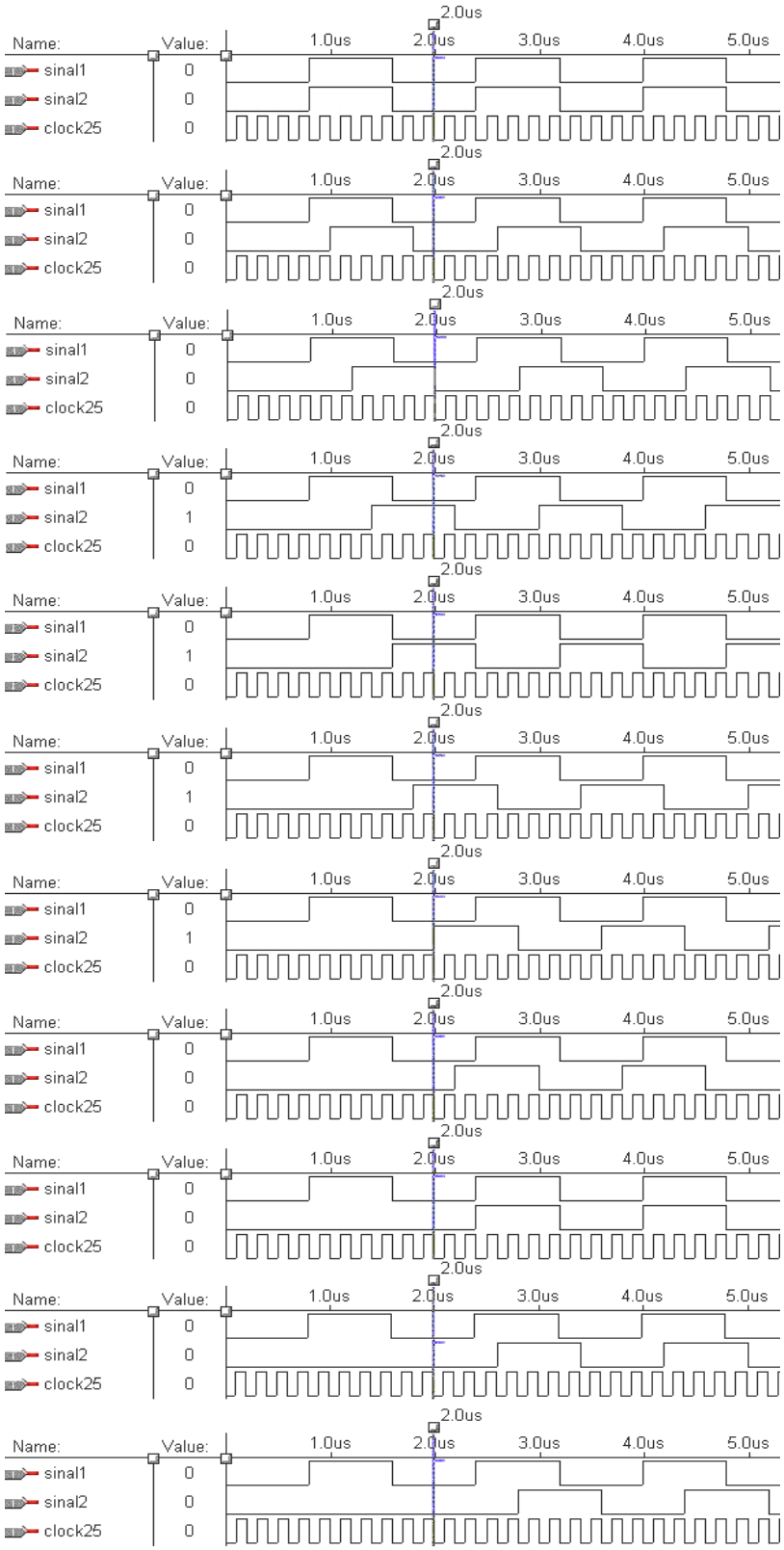

FIGURA 9.2– Seqüência de imagens com o comprimento de onda 8 vezes maior que o clock junto com os atrasos inseridos

9.1.2 Simulação 2

Esta simulação foi executada da seguinte maneira: Foi inserido um sinal com freqüência 6 vezes menor que o clock de amostragem de 5 Mhz, após 105ms foi lido o coeficiente de correlação calculado pelo correlacionador e foi realizada nova simulação porém desta vez com um dos sinais deslocados em 0.2µs. Este processo foi realizado com os sinais variando em um intervalo de tempo de -2.0µs a 2.0µs.

Os valores obtidos pelas simulações estão na TABELA 9.2 e estão plotados na FIGURA 9.3

Tabela 9.2 – Valores obtidos com sinal 6 vezes maior que o sinal de clock

Atraso em µs	Valor Obtido
-2	174763
-1,8	0
-1,6	174763
-1,4	349526
-1,2	524288
-1	349526
-0,8	174763
-0,6	0
-0,4	174763
-0,2	349526
0	524288
0,2	349526
0,4	174763
0,6	0
0,8	174763
1	349526
1,2	524288
1,4	349526
1,6	174763
1,8	0
2	174763

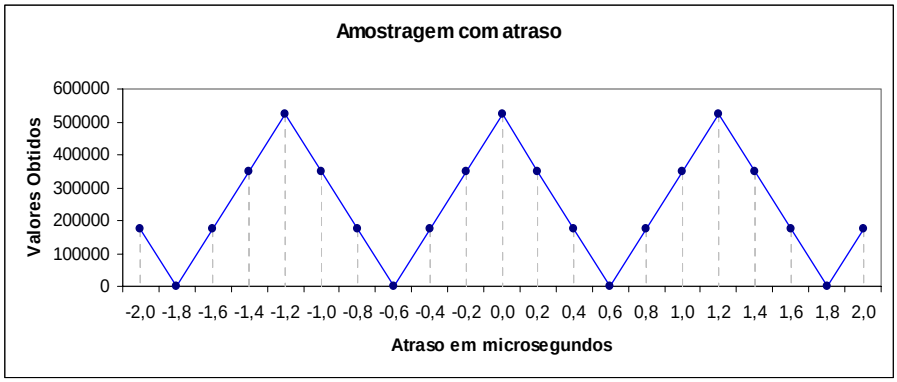

FIGURA 9.3– Gráfico obtido dos valores da tabela 9.2

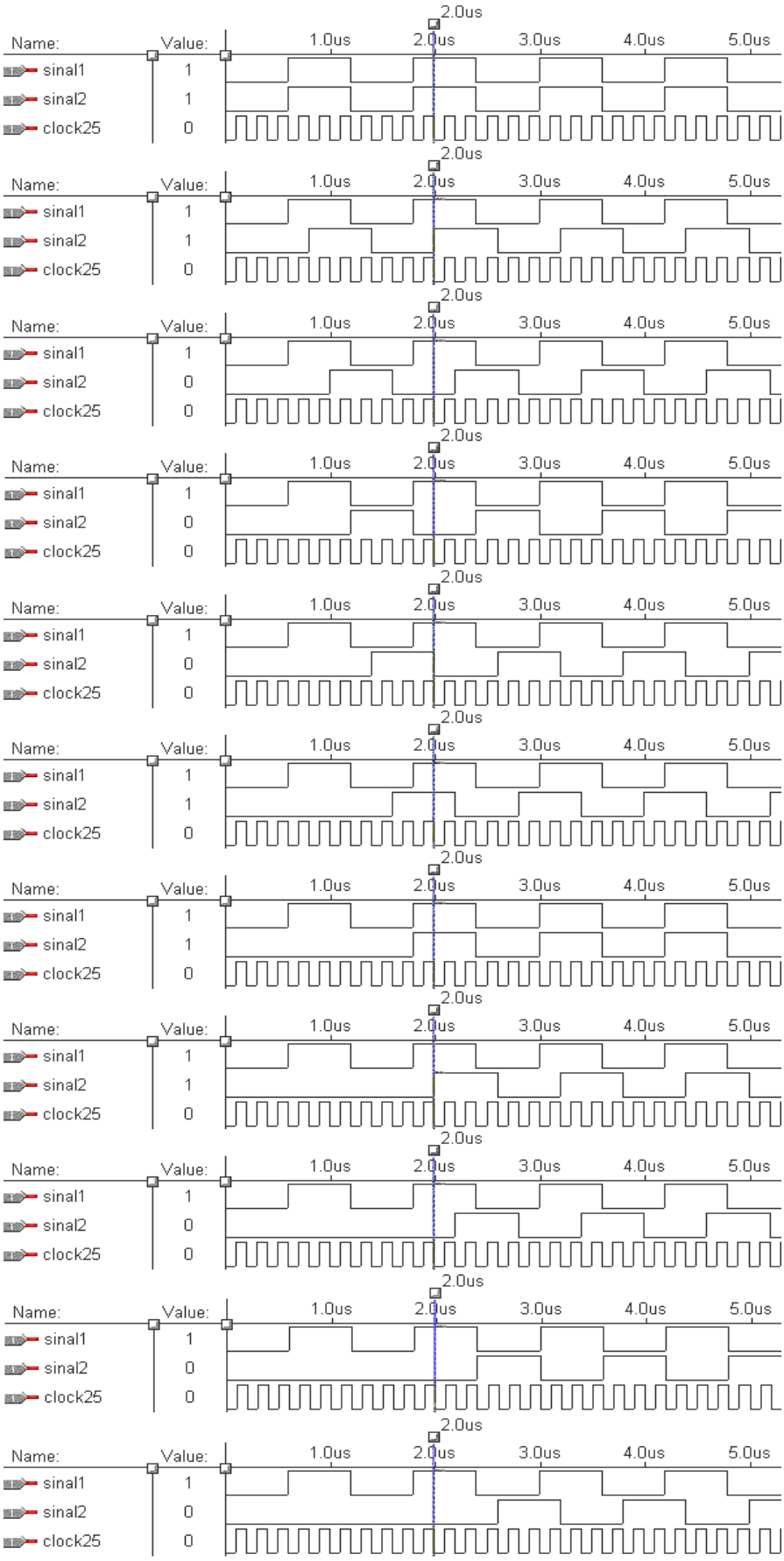

FIGURA 9.4– Seqüência de imagens com o comprimento de onda 6 vezes maior que o clock junto com os atrasos inseridos

9.1.3 Simulação 3

Esta simulação foi executada da seguinte maneira: Foi inserido um sinal com freqüência 4 vezes menor que o clock de amostragem de 5 Mhz, após 105ms foi lido o coeficiente de correlação calculado pelo correlacionador e foi realizada nova simulação porém desta vez com um dos sinais deslocados em 0.2µs. Este processo foi realizado com os sinais variando em um intervalo de tempo de -2.0µs a 2.0µs.

Os valores obtidos pelas simulações estão na TABELA 9.3 e estão plotados na FIGURA 9.5

Tabela 9.3 – Valores obtidos com sinal 4 vezes maior que o sinal de clock

Atraso em µs	Valor Obtido
-2	0
-1,8	262144
-1,6	524288
-1,4	262144
-1,2	0
-1	262144
-0,8	524288
-0,6	262144
-0,4	0
-0,2	262144
0	524288
0,2	262144
0,4	0
0,6	262144
0,8	524288
1	262144
1,2	0
1,4	262144
1,6	524288
1,8	262144
2	0

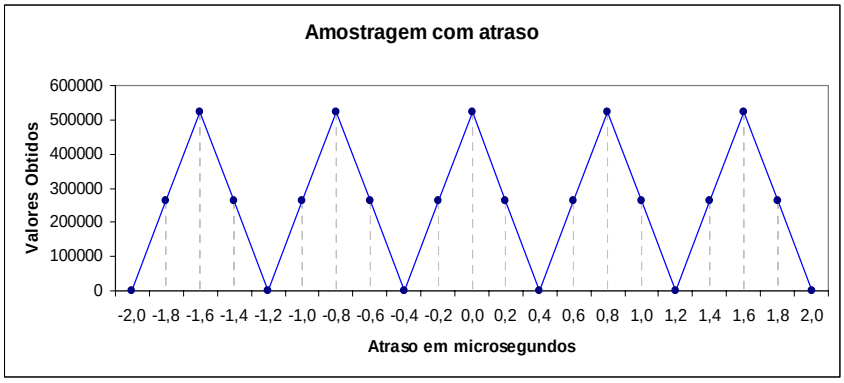

FIGURA 9.5– Gráfico obtido dos valores da tabela 9.3

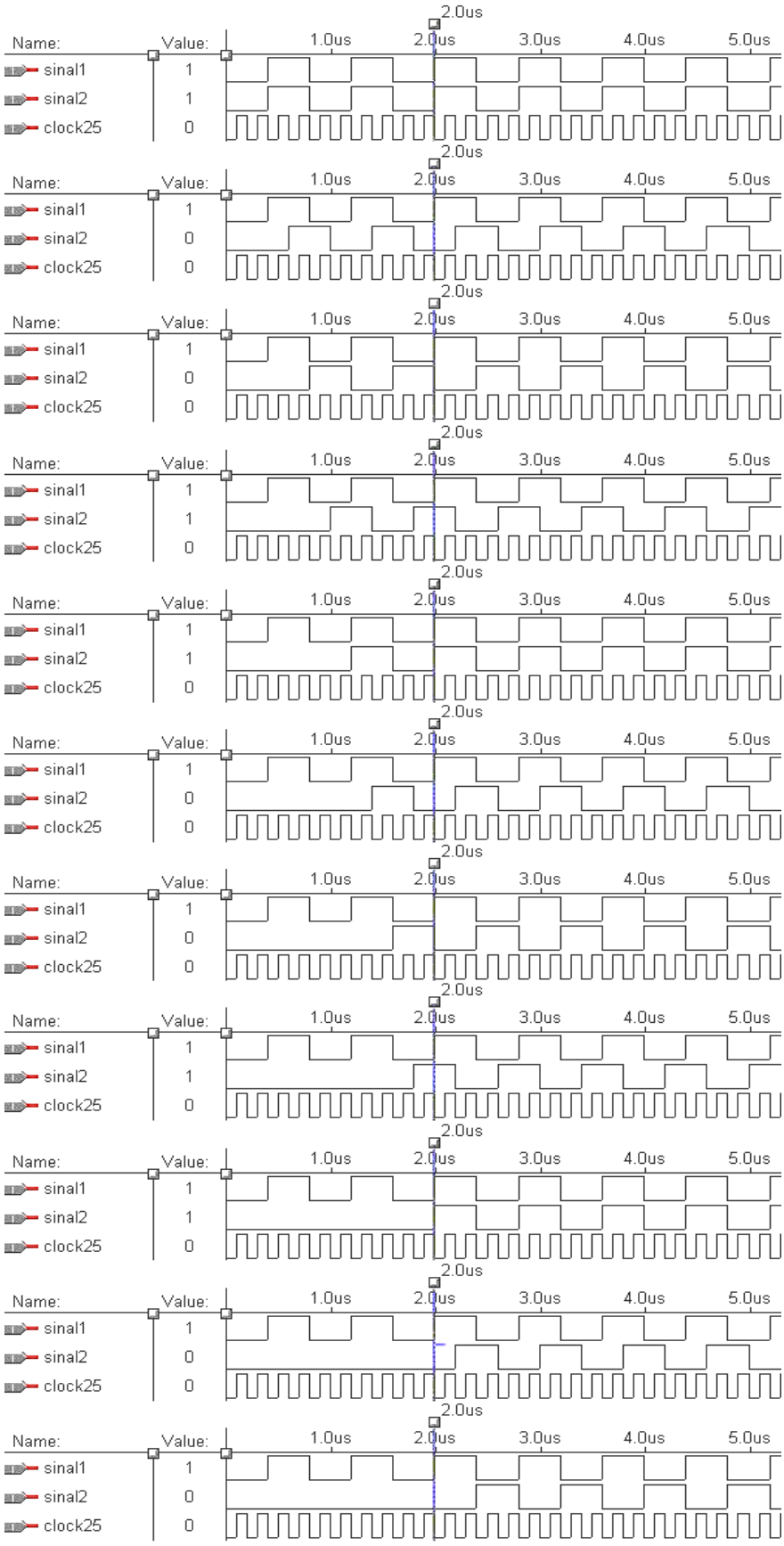

FIGURA 9.6– Seqüência de imagens com o comprimento de onda 4 vezes maior que o clock junto com os atrasos inseridos

9.1.4 Simulação 4

Esta simulação foi executada da seguinte maneira: Foi inserido um sinal com freqüência 2 vezes menor que o clock de amostragem de 5 Mhz, após 105ms foi lido o coeficiente de correlação calculado pelo correlacionador e foi realizada nova simulação porém desta vez com um dos sinais deslocados em 0.2µs. Este processo foi realizado com os sinais variando em um intervalo de tempo de -2.0µs a 2.0µs.

Os valores obtidos pelas simulações estão na TABELA 9.4 e estão plotados na FIGURA 9.7

Tabela 9.4 – Valores obtidos com sinal 2 vezes maior que o sinal de clock

Atraso em µs	Valor Obtido
-2	524288
-1,8	0
-1,6	524288
-1,4	0
-1,2	524288
-1	0
-0,8	524288
-0,6	0
-0,4	524288
-0,2	0
0	524288
0,2	0
0,4	524288
0,6	0
0,8	524288
1	0
1,2	524288
1,4	0
1,6	524288
1,8	0
2	524288

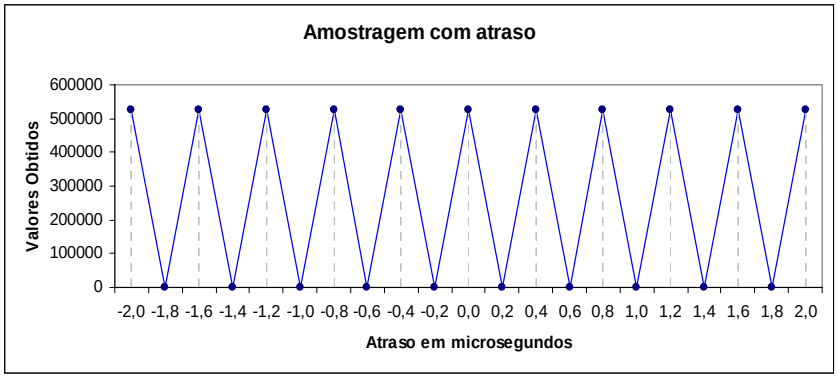

FIGURA 9.7 – Gráfico obtido dos valores da tabela 9.4

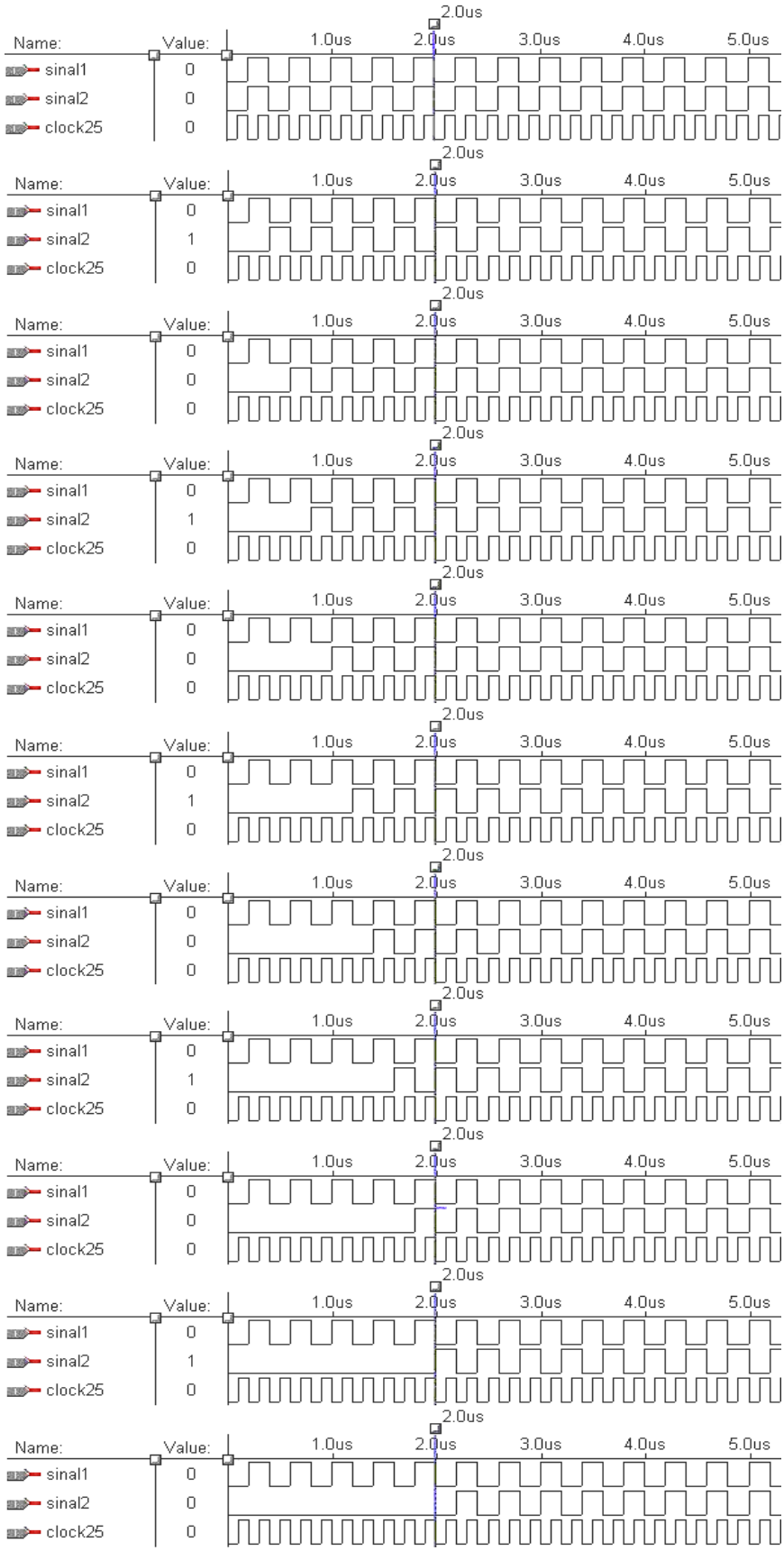

FIGURA 9.8 – Seqüência de imagens com o comprimento de onda 2 vezes maior que o clock junto com os atrasos inseridos

9.2 Simulações com sinal fixo e Taxa de Amostragem Variando.

Foram executadas também simulações com um clock de amostragem maior que 5 Mhz, este aumento da velocidade possibilita corrigir, de maneira mais eficiente, um possível atraso existente entre os sinais a serem correlacionados. Este aumento na velocidade da taxa de amostragem reduz também o tempo de integração fazendo com que o correlacionador retorne mais rapidamente o coeficiente de correlação entre os sinais no período de tempo determinado.

As simulações a seguir mostram as correlações com taxa de amostragem de 10 Mhz e 20 Mhz.

9.2.1 Simulação 1

Esta simulação foi executada da seguinte maneira: Foi inserido um sinal com freqüência 2 vezes menor que o clock de amostragem de 10 Mhz, após 52ms foi lido o coeficiente de correlação calculado pelo correlacionador e foi realizada nova simulação porém desta vez com um dos sinais deslocados em 0.2µs. Este processo foi realizado com os sinais variando em um intervalo de tempo de -2.0µs a 2.0µs.

Os valores obtidos pelas simulações estão na TABELA 8.5 e estão plotados na FIGURA 9.9

Tabela 9.5 – Valores obtidos com sinal 2 vezes menor que o sinal de clock

Atraso em µs	Valor Obtido
-2	524288
-1,8	0
-1,6	524288
-1,4	0
-1,2	524288
-1	0
-0,8	524288
-0,6	0
-0,4	524288
-0,2	0
0	524288
0,2	0
0,4	524288
0,6	0
0,8	524288
1	0
1,2	524288
1,4	0
1,6	524288
1,8	0
2	524288

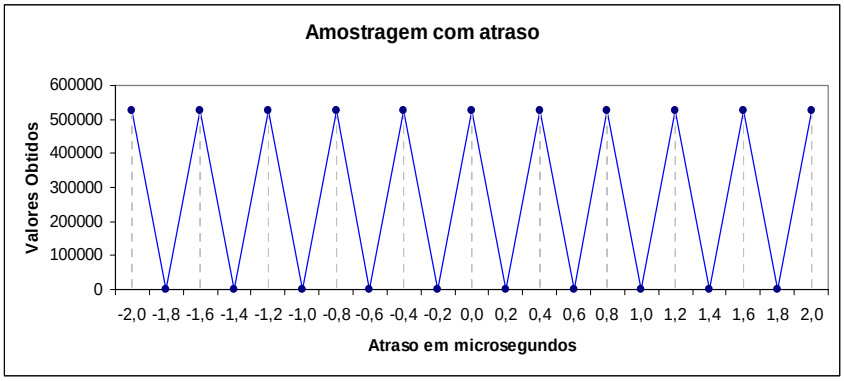

FIGURA 9.9 – Gráfico obtido dos valores da tabela 9.5

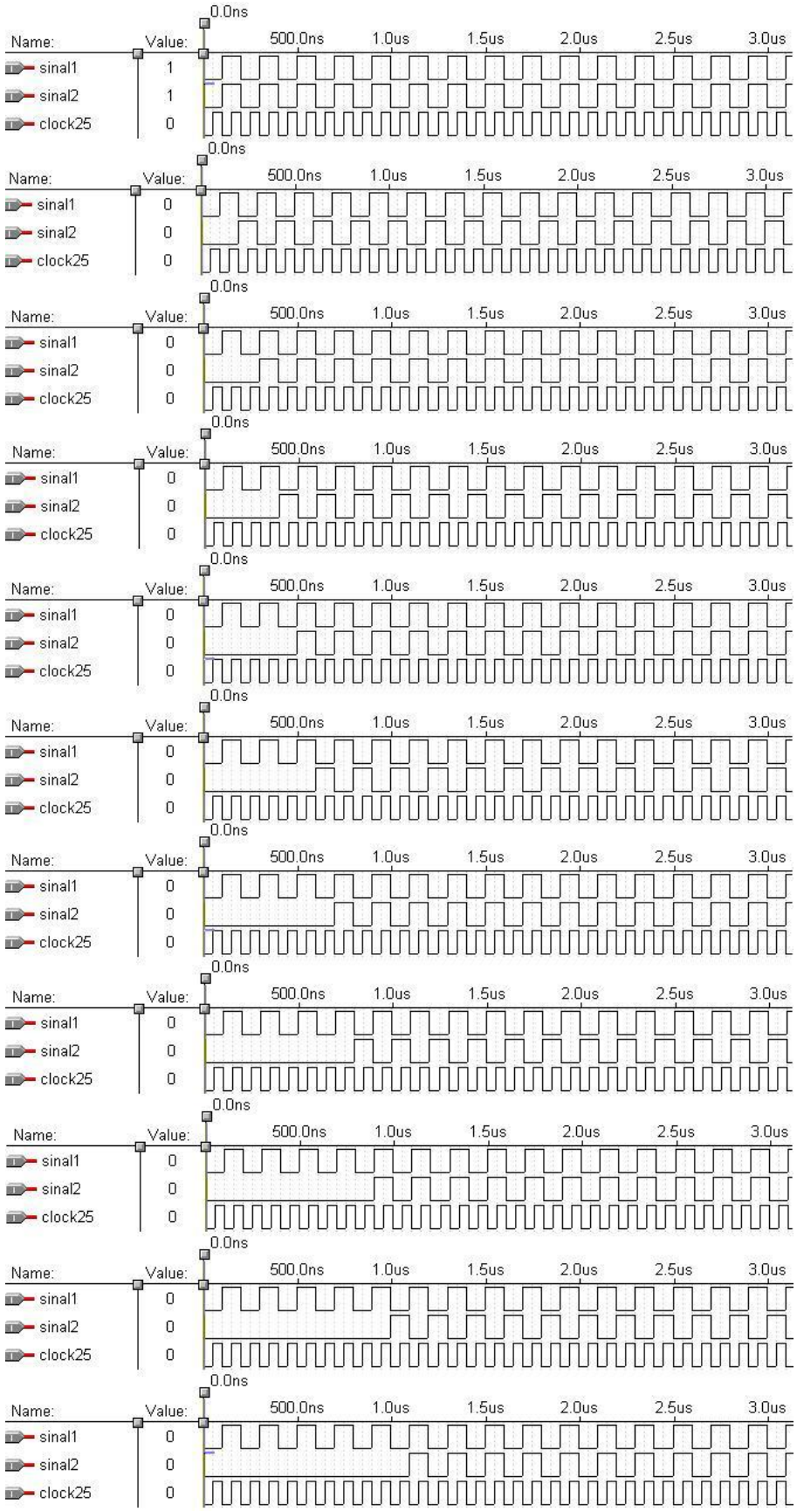

FIGURA 9.10 – Seqüência de imagens com o comprimento de onda 2 vezes menor que o clock junto com os atrasos inseridos

9.2.2 Simulação 2

Esta simulação foi executada da seguinte maneira: Foi inserido um sinal com freqüência 2 vezes menor que o clock de amostragem de 20 Mhz, após 25ms foi lido o coeficiente de correlação calculado pelo correlacionador e foi realizada nova simulação porém desta vez com um dos sinais deslocados em 0.2µs. Este processo foi realizado com os sinais variando em um intervalo de tempo de -2.0µs a 2.0µs.

Os valores obtidos pelas simulações estão na TABELA 9.6 e estão plotados na FIGURA 9.11

Tabela 9.6 – Valores obtidos com sinal 4 vezes maior que o sinal de clock

Atraso em µs	Valor Obtido
-2	0
-1,8	262144
-1,6	524288
-1,4	262144
-1,2	0
-1	262144
-0,8	524288
-0,6	262144
-0,4	0
-0,2	262144
0	524288
0,2	262144
0,4	0
0,6	262144
0,8	524288
1	262144
1,2	0
1,4	262144
1,6	524288
1,8	262144
2	0

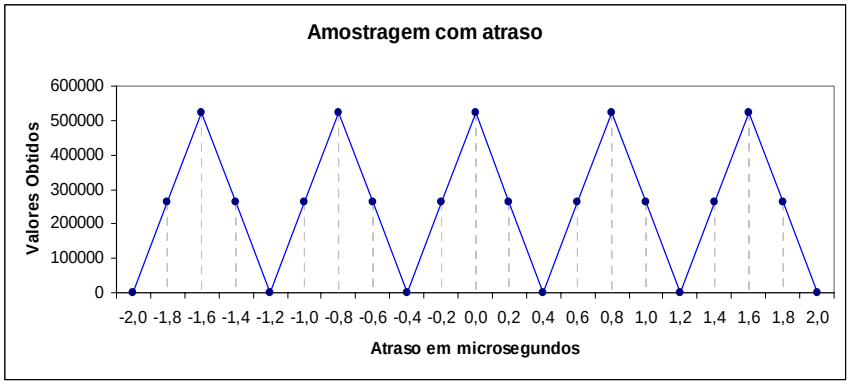

FIGURA 9.11 – Gráfico obtido dos valores da tabela 9.6

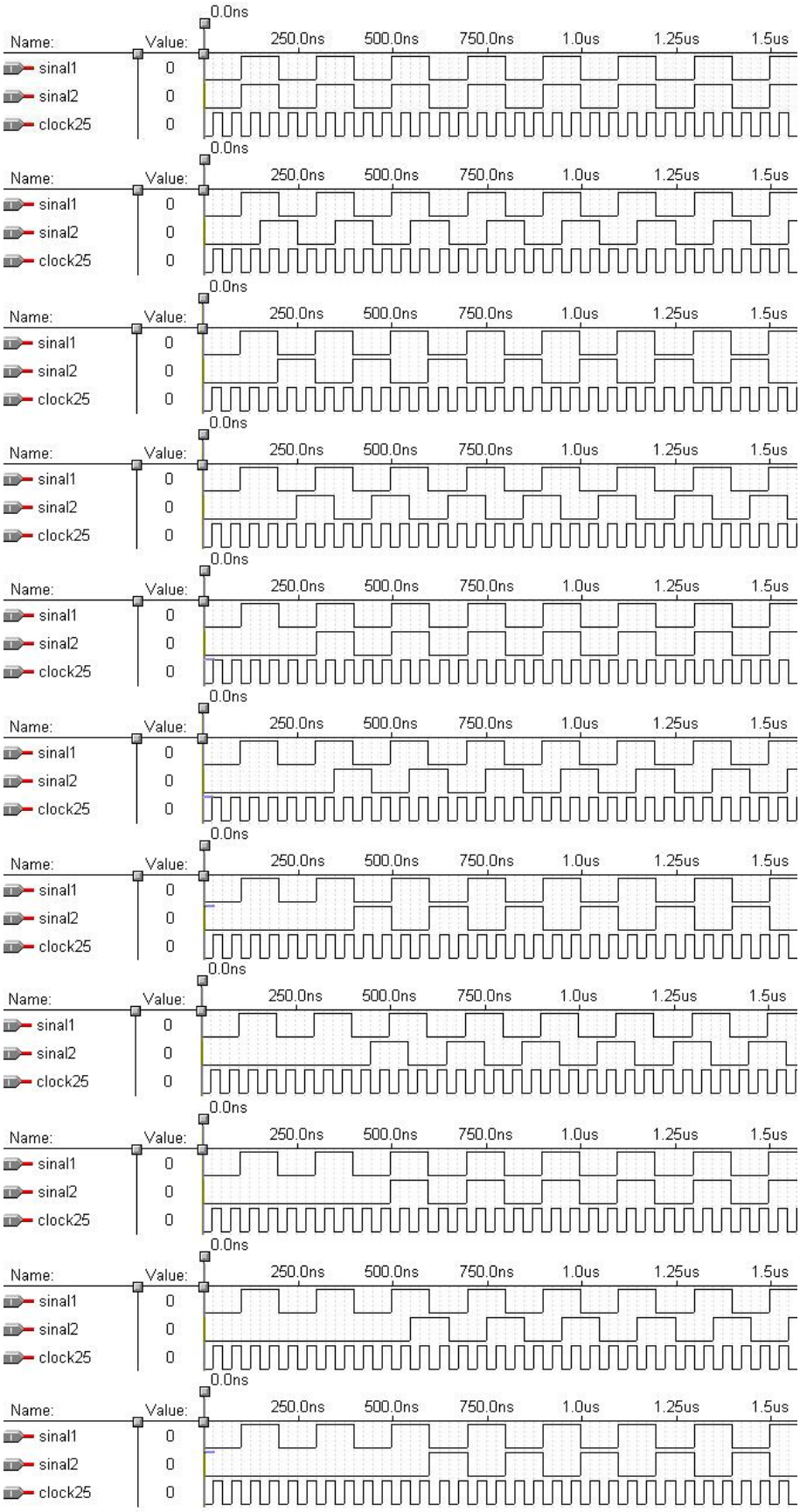

FIGURA 9.12– SEQÜÊNCIA DE IMAGENS COM O COMPRIMENTO DE ONDA 4 VEZES MENOR QUE O CLOCK JUNTO COM OS ATRASOS INSERIDOS

10 ANEXO 2 – CODIGOS FONTE

10.1 Codigo 1

```
#include <gtk/gtk.h>
int main(int argc, char **argv)
{
		GtkWidget *janela;
		gtk_init(&argc, &argv);
		janela = gtk_window_new(GTK_WINDOW_TOPLEVEL);
		gtk_window_set_title(GTK_WINDOW (janela), "Alo Mundo");
		gtk_widget_show(janela);
		gtk_main();
		return 0;
}
```

10.2 Codigo 2

```
#include <stdio.h>
#include <unistd.h>
#include <asm/io.h>
#define BASEPORT 0x378

int main()
{
	if (ioperm(BASEPORT, 3, 1)) {perror("ioperm"); exit(1);}
	outb(0, BASEPORT);
	printf("status: %d\n", inb(BASEPORT + 1));
	exit(0);
}
```

10.3 Codigo 3

```
#include <stdio.h>
#include <stdlib.h>
#include "./include/gnuplot_i.c"
int main()
{
	  gnuplot_ctrl *screenplot;
	  int contador;
	  double xplot[10],yplot[10];
	  for (contador=0;contador < 10;contador++)
	  {
		  xplot[contador]=contador;
		  yplot[contador]=contador;
```

```
        }
        screenplot = gnuplot_init();                                    (4)
        gnuplot_setstyle(screenplot, "lines");                          (5)
        gnuplot_cmd(screenplot, "set xlabel 'Tempo'");                  (6)
        gnuplot_cmd(screenplot, "set ylabel 'Coeficiente de Correlacao'");
        gnuplot_plot_xy(screenplot,yplot,xplot,10,”Primeiro Grafico”);  (7)

        gnuplot_close(screenplot);                                      (8)

        return 0;
}
```