Ministério da
Ciência e Tecnologia

GOVERNO FEDERAL
BRASIL
PAÍS RICO É PAÍS SEM POBREZA

DESENVOLVIMENTO DE TRANSPONDER DE COLETA DE DADOS DIGITAL PARA O SISTEMA DE COLETA DE DADOS AMBIENTAIS (SBCDA)

José Lenival Gomes de França

Relatório final do PIBIC, orientado por Mst. João Carlos Pécala Rae

INPE
Natal
2011

LISTA DE FIGURAS

Pág.

LISTA DE TABELAS

Pág.

SUMÁRIO

Pág.

1 INTRODUÇÃO

Esse relatório tem como objetivo apresentar o que foi feito na pesquisa sobre o desenvolvimento de transponder de coleta de dados digital para o sistema de coleta de dados ambientais (SBCDA) que está sendo desenvolvido no INPE-CRN. De forma mais específica, serão apresentados aqui a especificação, a implementação e os testes de uma unidade SPI (*Serial Periferal Interface*) em FPGA (*Field Programmable Gate Array*) para configuração do CI (Circuito Integrado) AD9874.

O AD9874 é responsável pela digitalização de FI (Freqência Intermediária) e fará parte do *Front-End*. A unidade SPI será parte do processador digital do transponder. Uma discursão um pouco mais aprofundada sobre o CI, a SPI e seu desenvolvimento serão temas dos próximos capítulos. O esquema do transponder pode ser visto na figura 1.1. Mais especificamente temos o esquema do *Front-End* na figura 1.2.

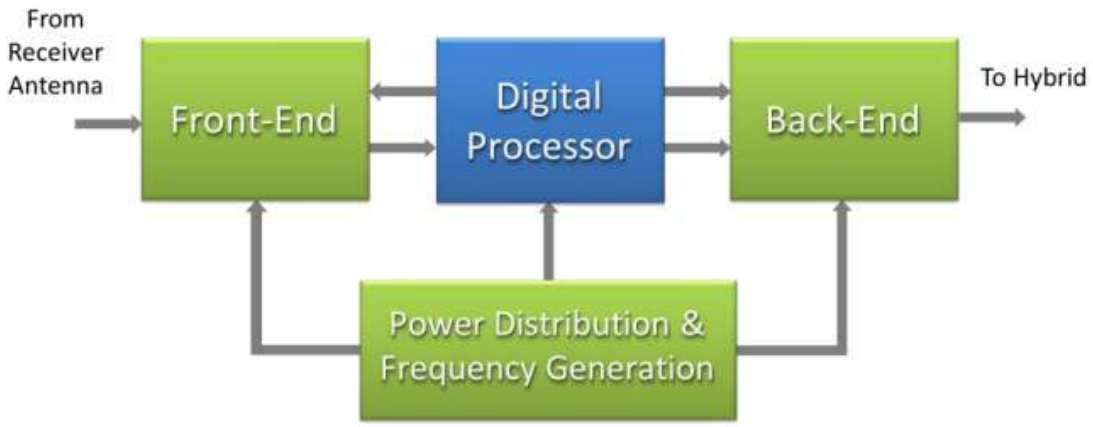

Figura 1.1 - Diagrama simplificado do transponder digital

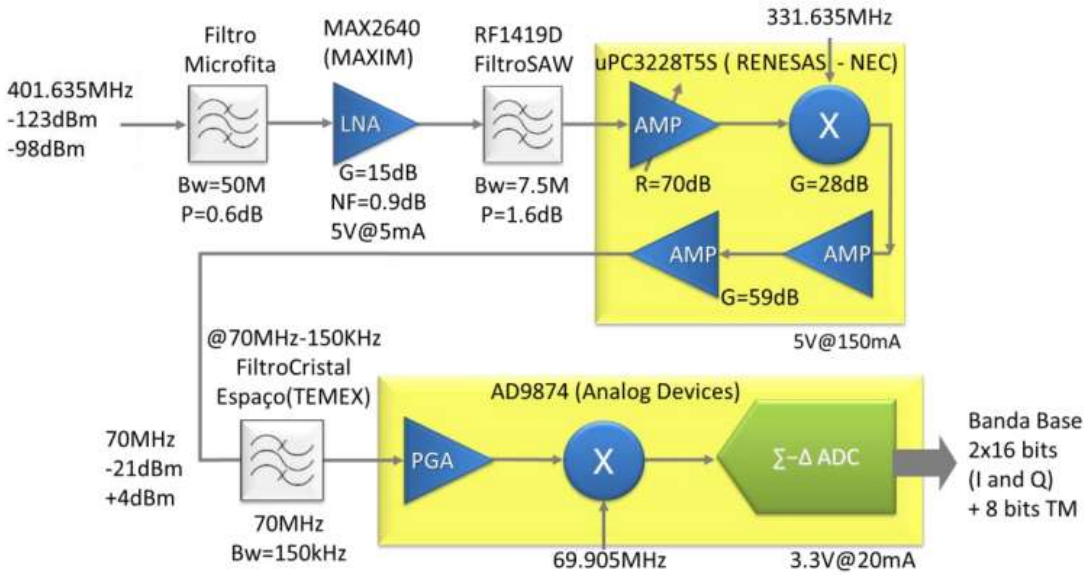

Figura 1.2 - Esquema do *Front-End*

2 O DIGITALIZADOR DE FREQUÊNCIA INTERMEDIÁRIA AD9874

O AD9874 é um subsistema de propósito geral que digitaliza sinais de 10 MHz a 300 MHz em FI com largura de banda entre 6.8 KHz a 270 KHz. Este subsistema é composto por um LNA (Amplificador de Baixo Ruído), um *mixer*, um ADC (Conversor Digital Analógico) sigma-delta e um filtro decimador com fator de decimação programável. Um circuito de AGC (Controle Automático de Ganho) interno ao AD9874 permite um ajuste de 12 dB no ganho do sinal de entrada.

Os parâmetros do AD9874 são configurados via porta SPI de acordo com a aplicação. Dentre os parâmetros que podem ser ajustados estão: tempo de decaimento/subida e atenuação do AGC, potência do sinal recebido, fator de decimação, formato do dado na saída e corrente de polarização. Esta interface será mais bem detalhada numa próxima seção.

O sinal digitalizado é entregue por meio de uma porta SSI (Interface Serial Síncrona) que pode funcionar com 2 ou 3 vias. No modo de três vias, um pino é destinado a sincronização do *frame* (FS), um para o *clock* (CLKOUT) de saída e o último (DOUTA) para o *stream* de dados. No caso de duas vias, o FS é suprimido e a sincronização do frame é feita por meio do *stream* de dados e apenas os pinos de *clock* e de dados (CLKOUT e DOUTA) são usados. Uma descrição mais detalhada de cada parâmetro, bem como características físicas do AD9874 pode ser conseguida em seu *datasheet* no site da *Analog Devices*.

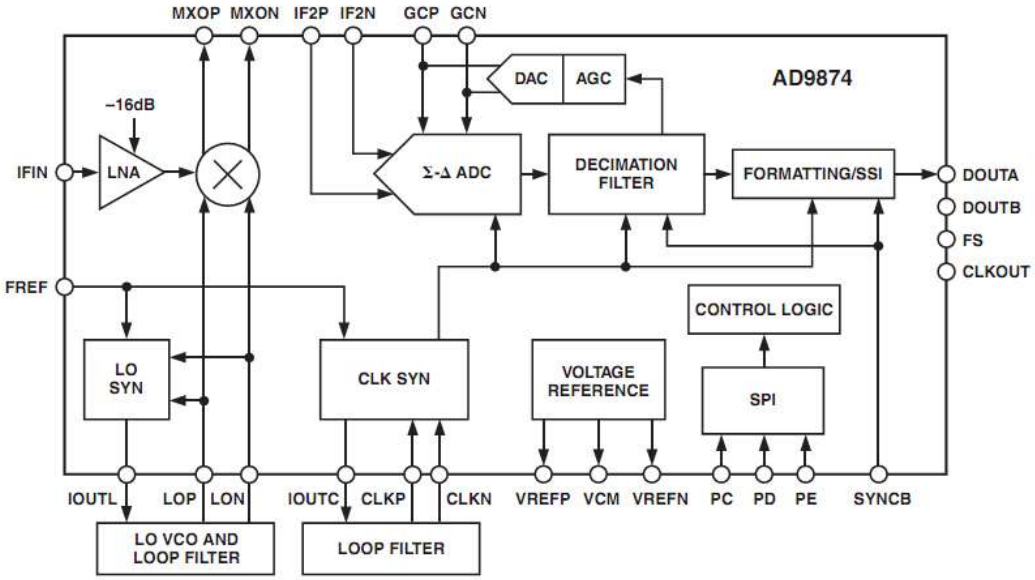

Figura 2.1 - Diagrama de blocos do AD9874

3 SERIAL PERIPHERAL INTERFACE (SPI)

A SPI, também chamada de comunicação *four-wires*, é uma interface de comunicação serial síncrona. Ela opera em modo *half-duplex* com quatro ou três vias de comunicação. No modo de quatro vias (4-*wires*) temos: uma via para o sinal de *clock*, uma para o *stream* de bits que será enviado, uma para o *stream* de bits que será recebido e uma última para habilitar a o processo de comunicação, este último tem uma importância maior quando temos o *interfaceamento* de múltiplos dispositivos com SPI. No modo de três vias (3-*wires*), as vias responsáveis pelos *streams* de dados tornam-se uma e esta é multiplexada no tempo de acordo com a necessidade. Embora muitas vezes esquecido, é importante que exista mais uma conexão referente aos *grounds* de ambos os dispositivos que estejam se comunicando via SPI.

Não existe um protocolo padrão para o SPI. Isto às vezes pode dificultar um pouco o processo de comunicação como será relatado mais a frente. Em linhas gerais, numa comunicação SPI entre dois dispositivos, um deles, chamado *Master*, dará início ao processo de comunicação e fornecerá o sinal de *clock* e o dado inicial. O outro dispositivo, o *Slave*, receberá o dado em sua via de entrada e, caso seja necessário o envio de resposta, faz-se este envio por meio de uma quarta via ou pela mesma via de entrada.

3.1. Processo de comunicação SPI detalhado

Para exemplificação a comunicação SPI, consideremos a figura 3.1.

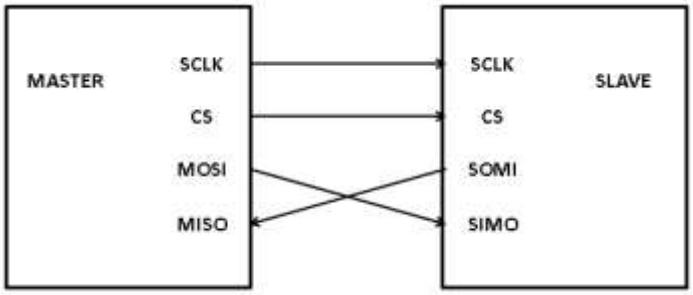

Figura 3.1 - Exemplo de dispositivos interligados via SPI

A figura 3.1 representa um dispositivo *Master,* um *Slave*, e as conexões entre eles. O processo de comunicação é descrito como a seguir:

a) O *Master* ativa o sinal CS (*Chip Select*), geralmente CS é ativo em nível baixo;

b) Após a ativação de CS, o primeiro bit a ser enviado é disponibilizado na via MOSI (*Master Out Slave In*);

c) Após disponibilizar o bit, o *Master* inicia o envio de um sinal de *clock*. De acordo com o dispositivo *slave*, este sinal de *clock* inicia com uma transição de subida ou de decida, pois esta transição que indicará ao *Slave* que um bit está sendo enviado e deve ser capturado. De acordo com os requisitos de ordem física do dispositivo *Slave*, entre a disponibilização de um bit e a transição de *clock* deve transcorrer um determinado tempo. Caso isto não seja obedecido, o dado que será enviado será uma versão deslocada de um bit do dado válido ou algo aleatório.

d) Na recepção do dado, o *Slave* capturará o dado recebido no pino SIMO (*Slave In Master Out*). Caso seja solicitada uma resposta do *Slave,* este fará por meio do SOMI (*Slave Out Master In*), no caso do modo 4-*wires.* Caso a comunicação seja 3-*wires* e o *Master* necessite de resposta do *Slave* após o dado ser enviado, o CS permanece ativo, a via MOSI torna-se entrada e a resposta fica sendo esperada por esta via.

3.2. Características da comunicação SPI do AD9874

Para a configuração do AD9874 foi escolhida a comunicação SPI com três vias, optou-se por esta alternativa devida não haver necessidade de comunicação bidirecional por esta porta. Estas vias de comunicação são referenciadas por: PC para via de *clock*, PD como de dados e PE como CS. O dado é capturado na subida de *clock* enquanto o pino CS mantem-se em nível lógico baixo.

A palavra enviada é de 16 bits, sendo o MSB (*Most Significant Bit*) enviado primeiramente. O MSB indica se a operação é de leitura (estado alto) ou escrita (estado baixo). Os seis bits posteriores indicam o endereço associado à operação. O próximo bit é um *don't care*, este serve para que seja possível deixar PD em alta impedância antes de uma operação de leitura onde o dado será recebido por esta via. O byte restante conterá a informação que será enviada, no caso de uma operação de escrita, ou recebida no caso de uma leitura.

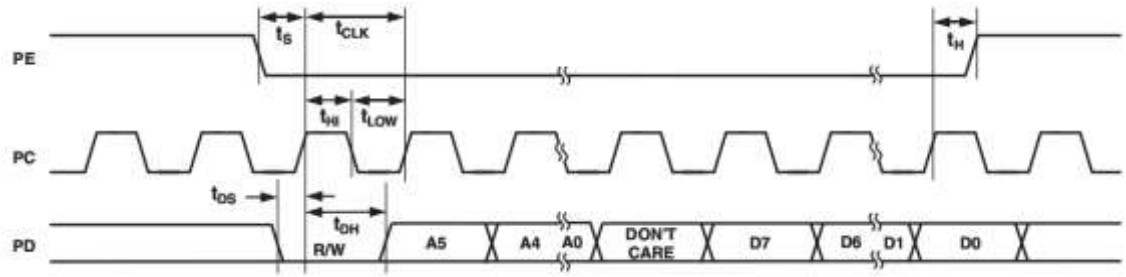

Figura 3.2 - Exemplo de tranferência de dados via SPI

As entradas de PC, PD e PE têm um *Schmitt Trigger* com uma histerese nominal de 0.4 V centrada na tensão de alimentação (VDDH/2). Assim, para o bit enviado ser considerado em estado alto, ele deve ter ultrapassado este limiar antes da transição do *clock*, o processo análogo ocorre para ser considerado em estado baixo. Este tempo até a transição é referenciado na figura 3.2, assim como outros tempos importantes ao projeto da interface de configuração para o CI.

3.3. Faixas de tempo a serem obedecidas

Na figura 3.2 foram feitas referências a alguns intervalos de tempo, estes intervalos são de imensa importância ao correto funcionamento da interface de comunicação. O descumprimento dos requisitos mínimos destes tempos pode gerar resultados indesejados e errôneos. As informações contidas na tabela a seguir foram retiradas do *datasheet* do CI e, segundo o próprio, são fixas para qualquer faixa de temperatura dentro daquelas nas quais o dispositivo funciona.

Tabela 3.1 - Requisitos de tempo da comunicação SPI

t_{CLK}	t_{HI}	t_{LOW}	t_{DS}	t_{DH}	t_S	t_H	
100	45	45	2	2	5	5	ns

Fonte: Datasheet do AD9874

4 PROJETO DA SPI EM VHDL A SER UTILIZADA NO PROJETO

A interface projetada deve atender aos requisitos de tempo impostos pelo AD9874, a figura 3.2 e a tabela 3.1 contém estas informações. Outros requisitos importantes são sua estabilidade após iniciado, o sistema deve convergir para um estado que não permita que operações indesejadas sejam executadas sem ordem específica para tal, e sua robustez, apenas um comando explícito para cancelamento ou reenvio perturbará o estado de transmissão ou recepção. Por último, o sistema deve ser descrito em VHDL (*Very High Speed Integrated Circuits Hardware Description Language*) de forma a ser compatível com o processador digital do transponder.

4.1. Primeira proposta – Descrição estrutural com unidades comportamentais

A primeira proposta foi implementar uma descrição comportamental do sistema. Tal proposta é uma forma direta e rápida de implementação, porém, sua natureza comportamental implica em desconhecimento, durante a fase de implementação, de quantas unidades lógicas serão utilizadas e como o software sintetizará o sistema em nível de *gates* e *flip-flops*. O resultado pode ser visto na figura 4.1. Os blocos apresentados em verde foram descritos de maneira comportamental.

Figura 4.1 - Representação RTL da primeira proposta

Esta proposta mostrou-se não confiável, pois alguns resultados esperados não foram observados. Ainda foram implementadas modificações ao sistema onde foram adicionados um controle de envio e uma unidade de leitura. Mesmo após

inúmeras correções, esta proposta foi abandonada. Após contato com a *Analog Devices*, concluiu-se que o descumprimento aos requisitos de tempo fazia com que uma versão deslocada do *stream* de dados fosse entregue ao CI.

4.2. Segunda proposta - Descrição estrutural

Após contatos com a *Analog Devices*, os requisitos de tempo começaram a ser levados em consideração. Uma versão para testes foi feita utilizando *LabVIEW*. Esta versão funcionou plenamente e a partir de então começou-se o desenvolvimento de uma nova proposta em VHDL que levasse em consideração os requisitos de tempo.

A nova proposta tem suas principais unidades descritas de maneira estrutural, utilizando-se apenas de *gates* e *flip-flops*. A partir dos requisitos, foram elaborados modelos, feitas simulações, testes e correções para atendimento de todos os requisitos estipulados. O sistema pode ser visto em seu mais alto nível na figura 4.2. As próximas seções descreverão os principais blocos construtivos e como eles funcionam.

4.3. Funcionamento dos blocos

A figura 2.2 mostra o sistema em seu mais alto nível, chamado no projeto de gravador. As interfaces de entrada do sistema são:

a) *en*: Quando em nível alto, desabilita o sistema completamente.

b) *clk*: Entrada de *clock* do sistema;

c) *clr*: *Reseta* o sistema;

d) *send*: Envia uma instrução de configuração gravada na memória ROM (bloco *data*).

As interfaces de saídas têm os mesmos nomes e funções dos pinos do AD9874 aos quais estão associadas.

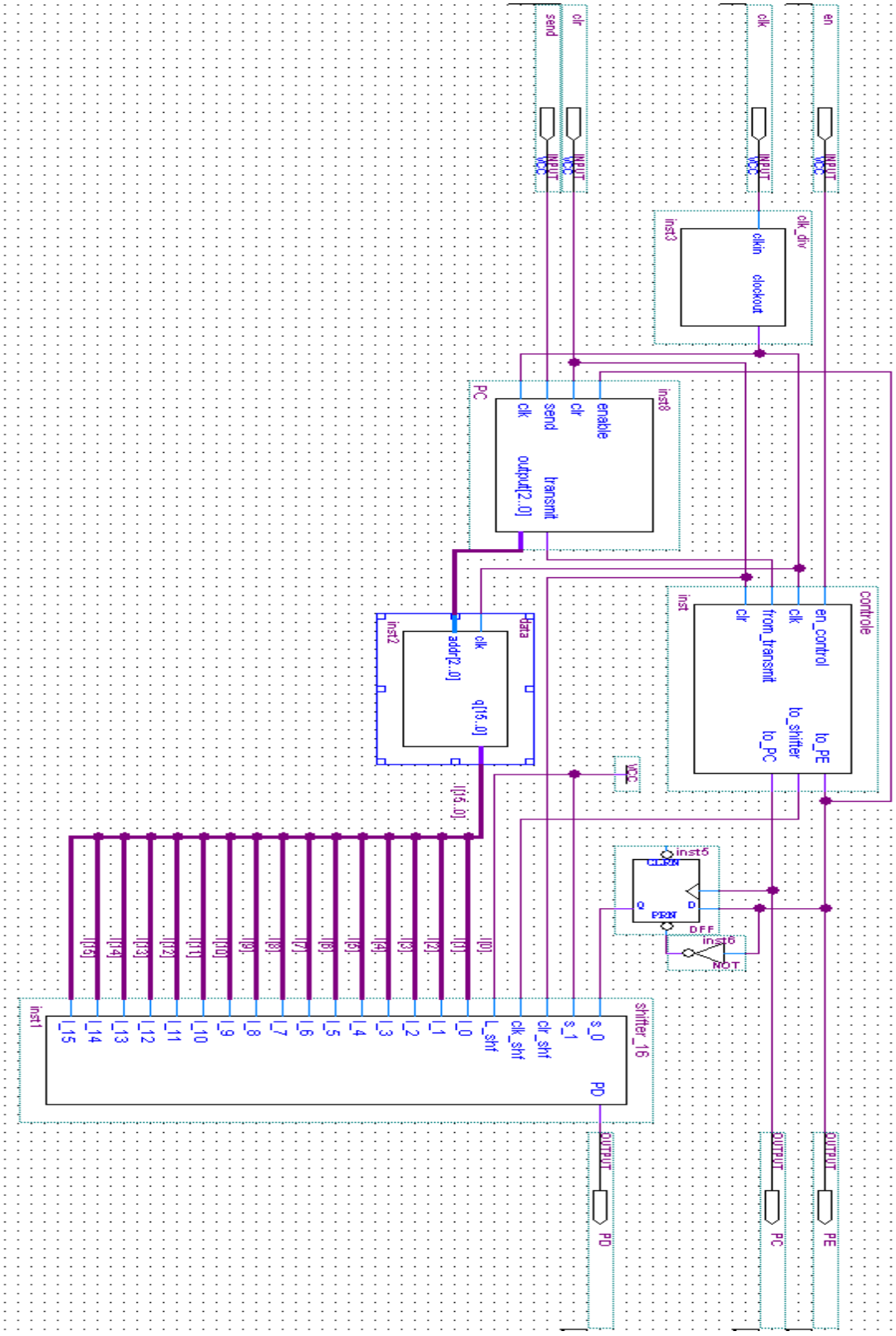

Figura 4.2 - Esquema em alto nível do sistema implementado

O sistema funciona enviando, pela via PD do AD9874, *streams* de 16 bits previamente gravados no bloco data e, durante o processo de envio, mantém o pino PE em nível-baixo e um sinal de *clock* no pino PC. O envio ocorre a cada transição de nível-baixo para nível-alto (subida) no pino *send*. Um evento de envio não pode iniciar enquanto outro estiver em execução, assim, múltiplas transições de *send* enquanto PE estiver em nível-baixo não surtirão efeito.

A análise do funcionamento inicia levando em consideração que o pino *en* está em nível-alto e que *send* e *clr* estão em nível-baixo. Para poder habilitar o funcionamento do sistema, fazemos com que o *en* vá para nível-baixo e, logo após, *clr* para nível-alto. Agora, quando *send* for para nível-alto, a unidade PC enviará um sinal *transmit* para a unidade controle e, quando ocorrer uma transição para nível-alto em *clk*, o dado na saída de PC será atualizado.

Agora, a unidade de controle, ao receber o sinal da unidade PC, fará com que o sinal PE vá para nível-baixo. Enquanto PE está em nível baixo, a unidade PC está desativada e a unidade shifter_16 opera em modo de deslocamento. Após pelo menos um ciclo do sinal *clockout* vindo da unidade *clk_div*, o controle começa a enviar transições pelo pino PC e pelo sinal *to_shifter*. Os sinais que saem por *to_PC* e *to_shifter* operam de forma a sempre ocorrer uma transição para nível-alto em *to_shifter* antes de uma em nível-alto em *to_PC*, assim, o bit a ser transmitido estará disponível antes da transição para nível-alto no pino PC.

A unidade *shifter_16* é um deslocador que opera de forma a quando o sinal *s_0* estiver alto, os bits presentes nas entradas *I_0* até *I_15* são armazenados a cada subida de *clk_shf*. Quando *s_0* está baixo o dado armazenado internamente será deslocado no sentido de *I_0* até *I_15*. Seu *buffer* interno é *resetado* quando *clr* está em nível baixo. O *flip-flop* D encontrado entre o sinal *to_PC* e a entrada *s_0* de *shifter_16* serve para que na primeira transição de *clk_shf*, ainda tenhamos *shifter_16* em modo de armazenamento.

Seguindo a análise, quando a unidade de controle executar 16 transições de subida para o sinal *to_PC*, o processo de transmissão termina e o sinal *to_PE* vai para nível alto permitindo que a unidade PC esteja preparada para receber um novo sinal *send.*

A unidade *data* funciona recebendo um valor de três bits vindos da unidade PC. Este valor corresponde ao endereço da instrução armazenada que será enviada pela via *q[15..0]*. Um ciclo de *clk* após um sinal *transmit* ser enviado pela unidade PC, a unidade data atualiza seu dado na via *q[15..0].*

As unidades controle e *shift_16* serão mais bem descritas nos apêndices A e B.

4.4. Simulações

Durante o projeto foram feitas inúmeras simulações para avaliar o pleno funcionamento do sistema. Concluída a implementação, foram feitas as simulações finais descritas aqui. Estas simulações têm por finalidade observar apenas o comportamento funcional do sistema, não levando em consideração interferência por fatores de ordem física.

4.4.1. Simulação da tentativa de múltiplas ordens de envio durante uma transmissão

A figura 4.3 mostra que o sistema é insensível a tentativas de reenvio durante a transmissão de uma instrução. O detalhe em vermelho indica um segundo sinal de envio antes de outro envio ser concluído.

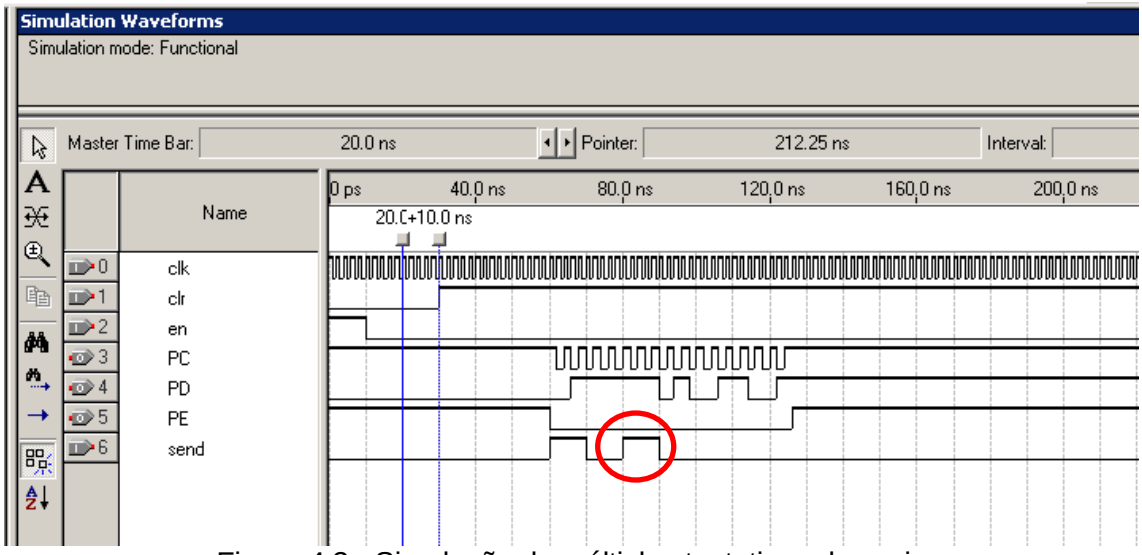

Figura 4.3 - Simulação de múltiplas tentativas de envio

4.4.2. Simulação de comando de reset

A figura 4.4 mostra o envio das configurações e, durante o envio da quarta *stream* de dados, a ocorrência de um evento de reset do sistema leva-o ao estado inicial do sistema. Como esperado, após um o pino *clr* passar por um nível baixo, o sistema retorna as suas condições iniciais de funcionamento. O detalhe em vermelho da figura 4.4 mostra o momento em que o sinal *clr* está em nível baixo.

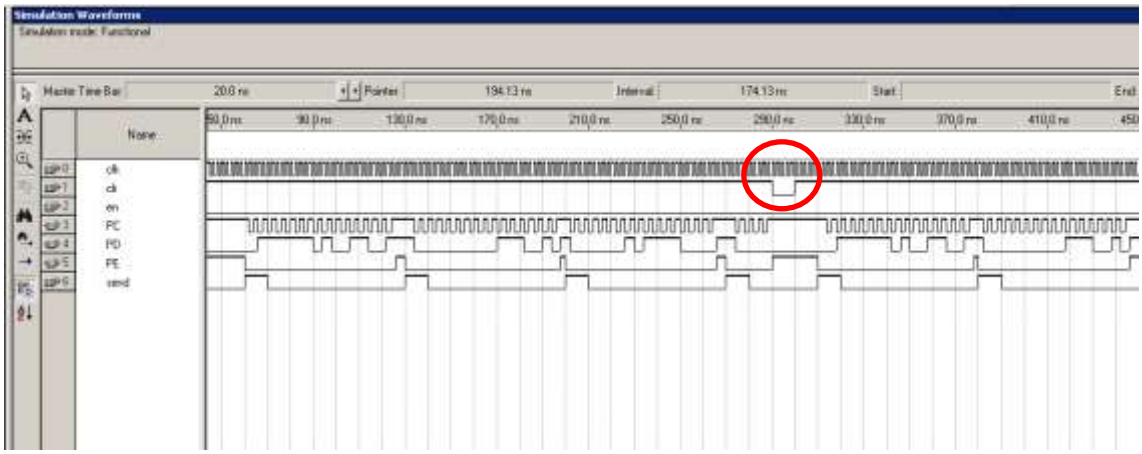

Figura 4.4 - Simulação do comando reset durante uma transmissão

4.4.3. Sensibilidade apenas a subida do sinal *send*

A figura 4.5 mostra que o sistema é sensível apenas a transição de subida do sinal *send*. O detalhe em vermelho mostra que mesmo quando o sinal *send* retorna ao nível baixo após um envio, o próximo envio só ocorre com uma subida.

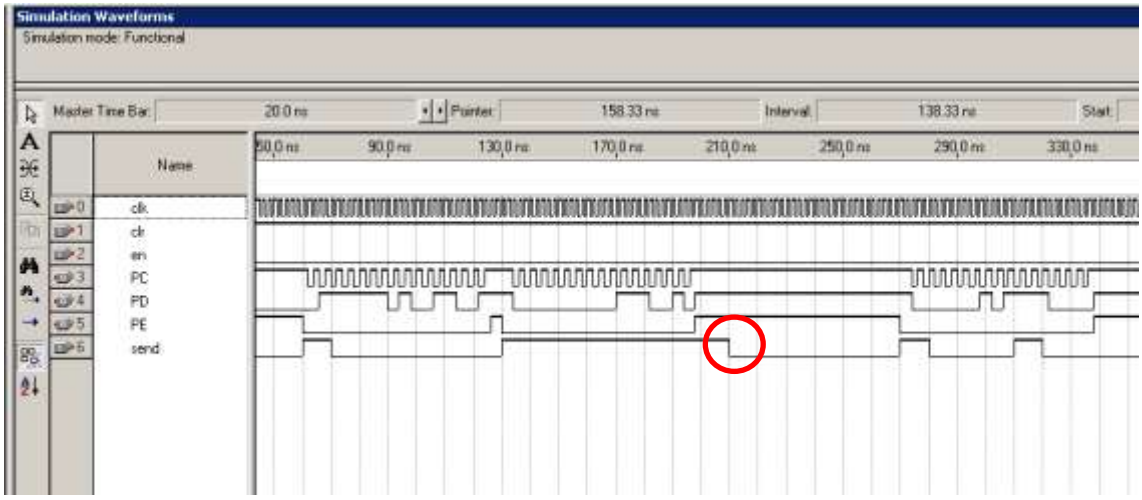

Figura 4.5 - Sensibilidade apenas a subida de *send*

4.4.4. Testes práticos

O objetivo dos testes práticos era utilizar esta interface em VHDL implementada no FPGA (*Field Programmable Gate Array*) *Altera Cyclone II* (EP2C35F672C6) da Placa de Desenvolvimento DE2, configurar seis registradores do AD9874 e obter os mesmos resultados conseguidos com a interface em *LabVIEW*. Os registradores configurados e suas funções são apresentados na tabela 4.1.

Tabela 4.1 - Configurações do AD9874 utilizadas nos testes da SPI

SPI Registradores							
Função	Endereço	End Bits	Bit Breakdown	Tamanho	Valor	Nome	Descrição
POWER CONTROL REGISTERS	0x00	00000000	(7:0)	8	01110010	STBY	Standby Control Bits (REF, LO, CKO, CK, GC, LNAMX, Unused and ADC). '0'=ON; '1'=OFF
	0x01	00000010	(7:6)	2	11110000	LNAB	LNA Bias Current ('0'= 0.5mA, '1'=1mA, '2'=2mA, '3'=3mA)
			(5:4)	2		MIXB	Mixer Bias Current ('0'= 0.5mA, '1'=1.5mA, '2'=2.7mA, '3'=4mA)
			(3:2)	2		CKOB	CK Oscilator Bias ('0'= 0.25mA, '1'=0.35mA, '2'=0.40mA, '3'=0.65mA)
			(1:0)	2		ADCB	Do not use.
TEST RESISTERS AND SPI PORT READ ENABLE	0x3A	01110100	(7:4, 2:0)	7	00001000	TEST	Factory Test Mode. Do not use.
			(3)	1		SPIREN	Enable read from SPI port.
	0x3B	01110110	(7:4, 2:0)	7	00000000	TEST	Factory Test Mode. Do not use.
			(3)	1		TRI	Three-state DOUTB
	0x3F	01111110	(7:0)	8	10011001	ID	Revision ID (Read-Only); A write of 0x99 to this resister is equivalent to a power-on reset.
DEC FACTOR	0x07	00001110	(7:5)	8	00000000	Unused	

Os testes foram bem sucedidos e os resultados podem ser vistos a seguir.

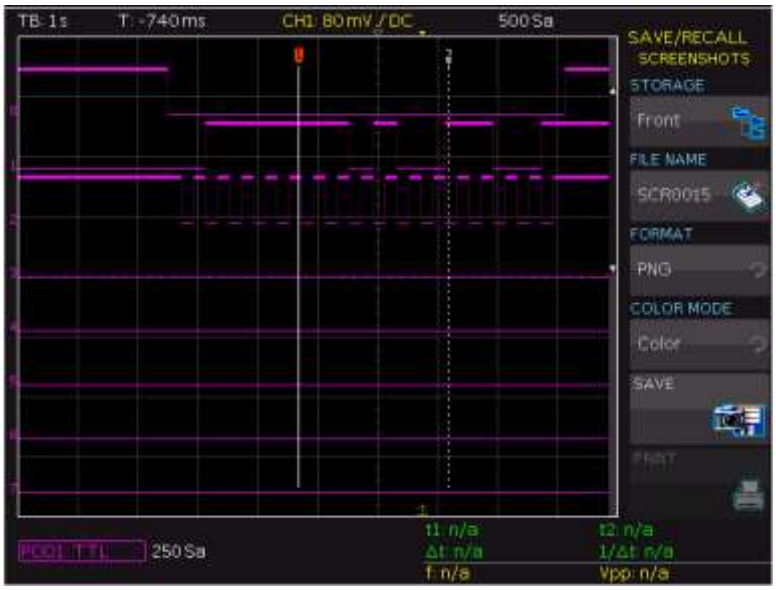

Figura 4.6 - Comando de write utilizando a SPI

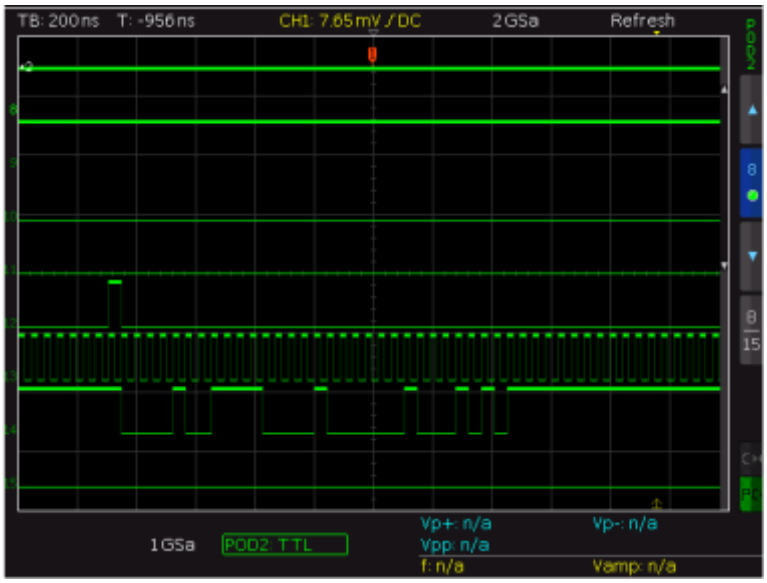

Figura 4.7 - Recepção do dado pela interface SSI

5 CONCLUSÃO

A implementação da SPI foi um sucesso e os testes confirmaram seu funcionamento. Para sua integração ao processador digital ainda é necessário que sejam definidos os parâmetros a serem configurados no AD9874.

O próximo passo é testar o conversor Digital-Analógico AD9767 que fará parte do *Back-End*. Depois de concluídos os testes, serão necessários a construção de uma PCB (*Printed Circuit Board*) na qual estarão apenas os componentes indispensáveis ao funcionamento do AD9767 e sua integração ao outros componentes do *Back-End*.

REFERÊNCIAS BIBLIOGRÁFICAS

livro

PEDRONI, V. A. **Circuit design with VHDL**. Edição 1. Massachusetts Institute of Technology. 2004. 376 páginas. ISBN 0-262-16224-5

manuais

Datasheet AD9874 Rev. A– ANALOG DEVICE, IF Digitizing Subsystem.

programa de computador

Quartus II Web Edition Versão 9.0 Build. Local: Altera, 2009, arquivo iso disponível em www.altera.com acessado em 29/04/2009.

APÊNDICE A - IMPLEMENTAÇÃO DO DESLOCADOR BIDIRECIONAL

Levando em consideração possíveis necessidades de leitura de um dado escrito previamente em um endereço do AD9874, o deslocador de 16 bits utilizado é bidirecional. Este deslocador é resultado do cascateamento de 16 deslocadores de 1 bit. Vejamos o esquema da unidade deslocadora de 1 bit na figura A.1.

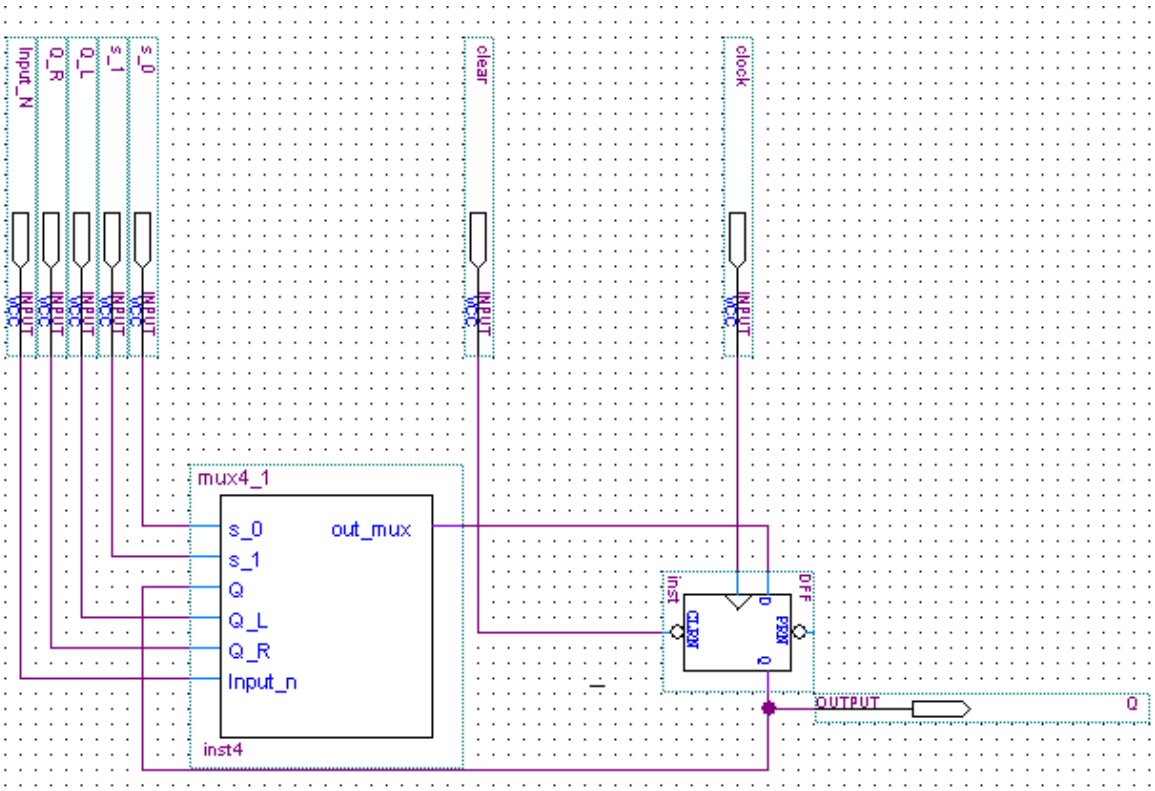

Figura A.1 - Implementação do deslocador bidirecional de 1 bit

A unidade *mux4_1* é um multiplexador com 4 entradas de 1 bit. Seu esquema pode ser visto na figura A.2. O deslocador segue a lógica de controle descrito pela tabela A.1.

Tabela A.1 - Tabela com descrição do funcionamento da unidade *mux4_1*

S_0	S_1	clk	Q
0	0	↑	Q
0	1	↑	Q_L
1	0	↑	Q_R
1	1	↑	Input_N

Caso venha a ser necessária a implementação da recepção, é necessário que a saída PD seja bidirecional e sua direção controlada. No procedimento de recepção, é necessário que, após o envio do sétimo bit, a via de envio se torne entrada durante o tempo do bit *don't care.*

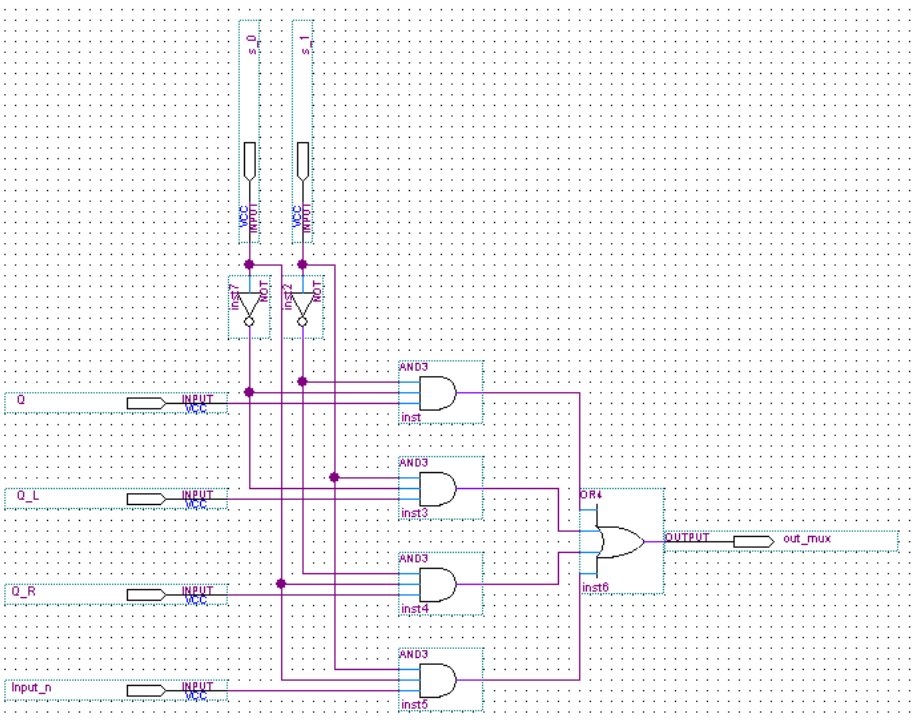

Figura A.2 - Implementação da unidade mux4_1

APÊNDICE B - PROJETO DO CONTROLE DE TEMPORIZAÇÃO E CLOCK DE ENVIO

O bloco de controle foi projetado de forma a atender aos requisitos de temporização e controlar o deslocamento do dado no registrador de deslocamento. A figura B.1 mostra o esquema do bloco de controle.

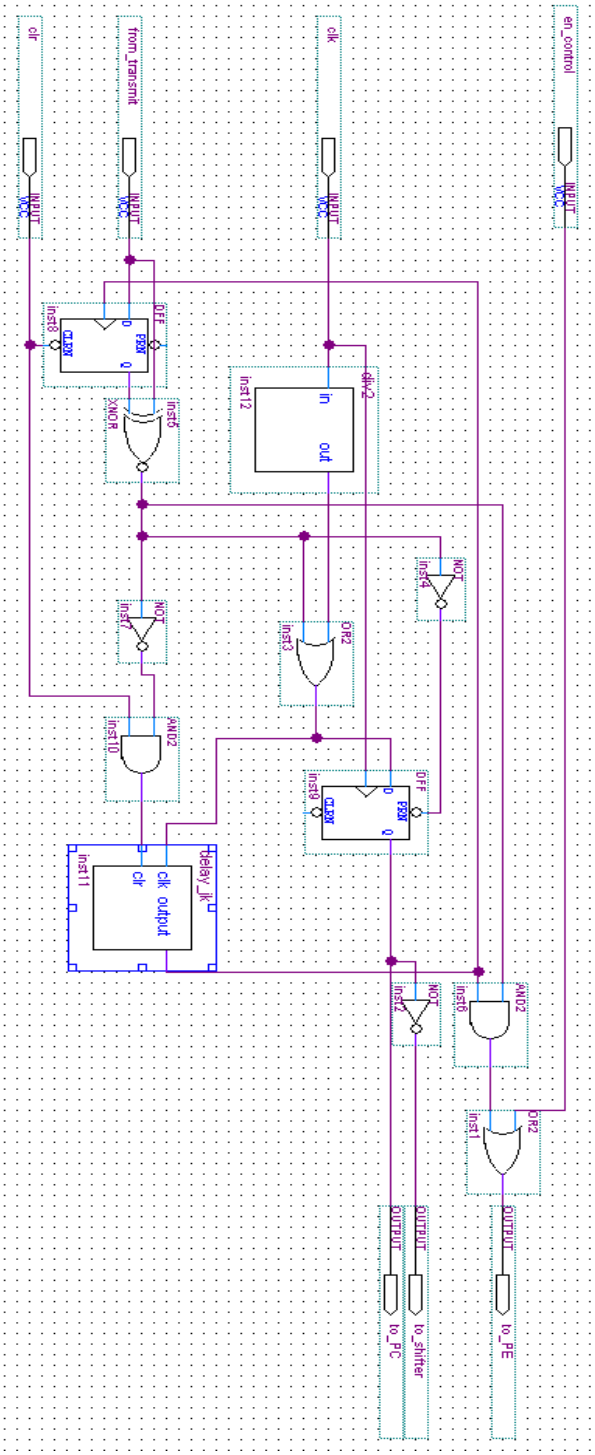

Figura B.1 - Implementação da unidade de controle

A unidade div2 é um simples divisor de *clock* que divide a frequência do sinal em sua entrada por 2 e a unidade *delay_jk* é uma unidade feita com *flip-flops jk* com o intuito de contar 16 transições de subida que entrem do sinal de entrada clk. Veja o esquema destes blocos nas figuras B.2 e B.3.

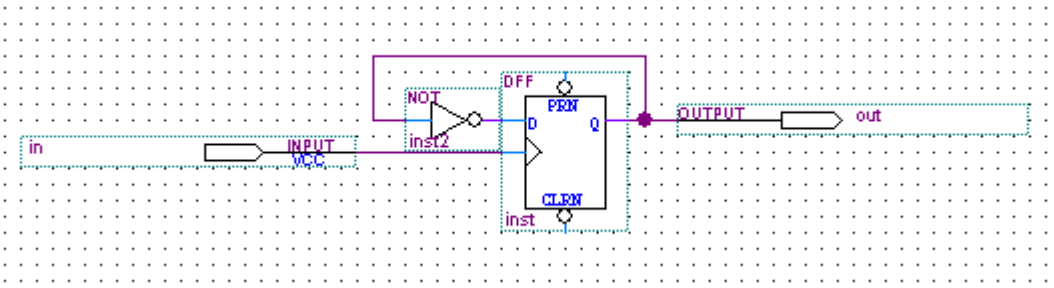

Figura B.2 - Implementação da unidade div2

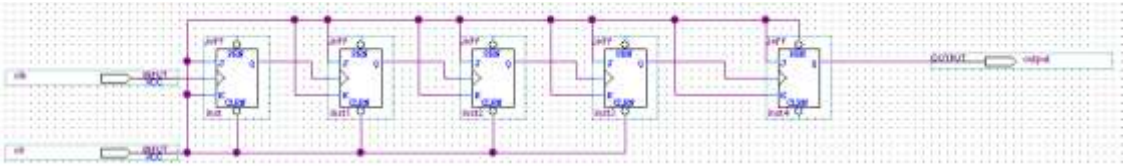

Figura B.3 - Implementação da unidade delay_jk

B.1 Análise da unidade controle

Com as informações a respeito de cada unidade, podemos começar a entender o funcionamento da unidade controle.

Certas condições iniciais devem ser supostas, estas são: A entrada *en* deve está em nível-alto, isto impede que o sinal de saída PE esteja inicialmente ativo, e as entradas *clr* e *from_transmit* devem estar em nível-baixo.

a) Para o funcionamento ter início, é necessário que en esteja em nível baixo e *clr* em alto.

b) Enquanto a unidade controle estiver ociosa, teremos um sinal de *preset* para o FFD (inst9) e de *clear* para a unidade *delay_jk* (inst11) proveniente da porta *xnor* (inst5), que estará com sua saída em nível-alto neste momento.

c) Quando ocorrer uma mudança de nível na entrada *send*, a *xnor* terá sua saída em nível-baixo devido a este sinal está sendo atrasado pelo FFD (inst8) em uma de suas entradas.

d) O nível-baixo na *xnor* ativará *delay_jk*, fazendo com que em sua saída fique em nível-baixo e só retorne a nível alto após 16 transições de subida em sua entrada *clk*.

e) Enquanto *delay_jk* estiver com sua saída em nível-baixo, o pino *to_PE* estará em nível-baixo.

f) Quando ocorrer uma subida em *delay_jk*, o FFD (inst8) atualizará sua saída e a porta *xnor* resetará *delay_jk*.

g) A unidade *div2* disponibilizará um sinal de *clock* com metade do período do *clock* de entrada do sistema. Este sinal só é fornecido aos pinos *to_shifter* e *to_PE* enquanto o FFD (inst9) não estiver em estado de *preset*.