PERS - Um Processador Específico para Redes de Sensores com Primitivas de Segurança

Alexandre Ponce de Oliveira
Univem
alexandreponce@terra.com.br

Edward David Moreno
Univem
edmoreno@fundanet.br

Kalinka J. Castelo Branco
Univem
kalinka@fundanet.br

Resumo

Este artigo propõe um processador específico para redes de sensores (chamado PERS). A grande limitação computacional de hardware dos sensores é um dos principais desafios encontrados para prover segurança para as redes de sensores. Esta pesquisa mostra um processador específico para redes de sensores utilizando a topologia ponto-a-ponto, sua arquitetura, conjunto específico de instruções (ISA), simulações e resultados de seu desempenho em FPGAs. Mostra também as comparações da versão inicial com essa versão final que inclui primitivas de segurança (criptografia), para que se tenha um bom nível de segurança nas informações trafegadas.

1. Introdução

As Redes de Sensores vêm crescendo cada vez mais, impulsionadas pela diminuição dos componentes eletrônicos, da evolução dos componentes de hardware e das comunicações sem fio. Estão sendo utilizadas para diferentes aplicações como: climáticas, químicas, biológicas, militares, educacionais, médicas entre outras. Antigamente o principal objetivo delas era a monitoração remota de ambientes inimigos ou de difícil acesso, tendo sido desenvolvidas e utilizadas em aplicações militares com o objetivo de monitorar o campo de batalha em busca de ameaças [7].

Em [4], uma rede de sensores é definida como uma classe particular de sistemas distribuídos, onde as comunicações de baixo nível não dependem da localização topológica da rede. Desta forma, possui características particulares como a utilização de recursos restritos de energia, topologia de rede dinâmica e uma grande quantidade de nós. Estas características dificultam a reutilização de alguns algoritmos desenvolvidos para outros tipos de sistemas distribuídos. As soluções para estes problemas, como a sincronização da rede, a eleição de um líder e a aquisição de informações que representam o estado da rede deve considerar também características como a precisão, eficiência e o custo das operações.

Numa rede de sensores típica os sensores individuais apresentam amostras de valores locais e disseminam informação, quando necessário, para outros sensores e eventualmente para o observador.

Um dos pontos mais críticos das Redes de Sensores são suas limitações de recursos, como por exemplo, pouca capacidade computacional, pouca memória e o principal, sua limitação de energia (oriunda de uma bateria). Esta limitação tem influência nessas redes, pois os sensores são utilizados em áreas de difícil acesso, impossibilitando uma possível manutenção. Devido a isso deve-se projetar essas redes visando baixo consumo de energia [7].

Em [13], são apresentados alguns fatores que criam diferentes desafios para a tecnologia de sensores:

- os nós encontram-se embutidos numa área geográfica e interagem com um ambiente físico;
- são menores e menos confiáveis que roteadores de redes tradicionais;
- geram dados detectados, ao contrário de roteadores de rede;
- podem ser móveis.

Esses fatores representam um novo paradigma para as operações de rede, ferramentas de software e protocolos para habilitar a programação e o uso efetivo de tais sistemas de computação embutida em redes [13].

A cooperação entre os nós sensores é de vital importância para o funcionamento da rede, ou seja, nenhum nó pode entrar na rede e se negar a encaminhar pacotes de dados ou de controle [5].

Além disso, os serviços de segurança como a inclusão de um algoritmo de criptografia nos dados trafegados na rede, são muito importantes para manter a privacidade dos dados que trafegam na rede, pois o acesso aos dados será possível apenas com o conhecimento da chave para decifrar a informação recebida.

Para obter-se segurança em uma rede de sensores devem-se cumprir determinados requisitos. Esses

requisitos são importantes, pois pesam na escolha do melhor algoritmo de criptografia a ser utilizado [11].

Os melhores algoritmos de segurança são aqueles que conseguem proteger da melhor forma a aplicação e também consumir o mínimo dos recursos que as redes de sensores possuem.

Os problemas e soluções citadas acima levam a um estudo aprofundado e minucioso das Redes de Sensores, Processadores de Redes e Segurança, de modo que todos esses fatores motivaram a elaboração deste artigo, pois os mecanismos de segurança em redes de sensores ainda têm muito que evoluir e ainda se mostram em aberto para pesquisa e desenvolvimento técnico.

O principal objetivo é desenvolver um processador específico para redes de sensores, incluindo primitivas de segurança (criptografia), pois a adaptação de um processador de rede já existente pode levar um tempo maior, uma vez que muitas instruções não seriam aproveitadas em função de sua aplicação em uma rede de sensores. Com isso, pode-se introduzir um determinado nível de segurança para proteger as informações que são transmitidas por sensores em uma rede, sem interferir no desempenho da rede e na durabilidade dos sensores.

2. Trabalhos Correlatos

O processador (PERS) foi desenvolvido com base no NPSoC - Um novo Processador de Rede [12], no Maté - Uma Máquina Virtual para Redes de Sensores [6] e no Criptoprocessador VLIW [10].

O NPSoC foi um processador de rede implementado em VHDL e prototipado em FPGAs [9], é baseado no RCNP Processador de Rede com Suporte a Multi-protocolo e Topologias Dinâmicas [1,2] e no R2NP (*Reconfigurable RISC Network Processor*) – Processador de Rede RISC Reconfigurável [3]. O NPSoC possui as características do RCNP, portanto, os autores modificaram a proposta inicial devido à complexidade na implementação das *microengines* do R2NP, dificultando assim a visualização de seu funcionamento. Seu conjunto de instruções segue o modelo RISC, onde, com apenas 32 instruções foi possível descrever alguns algoritmos de roteamento presentes nos NPs comerciais atuais [12].

Maté é uma máquina virtual que executa no TinyOS que é um sistema operacional projetado especificamente para uso em redes de sensores [6]. O Maté tem duas pilhas (uma pilha de operandos e uma pilha de endereço de retorno) a maioria das instruções opera unicamente na pilha de operandos, mas algumas instruções controlam o fluxo de programa e vários operandos embutidos. O Maté foi projetado para executar tanto nos hardwares MICA como no RENE2 e em conjunto com seus subcomponentes utiliza 1KB de RAM e 16 KB de memória com suas instruções [6].

O criptoprocessador de [10], foi projetado para executar preferencialmente algoritmos de criptografia simétricos e para isso, módulos especiais foram descritos e projetados para aumentar o desempenho e simplificar o programa fonte. Este criptoprocessador foi descrito utilizando a linguagem VHDL, suporta uma série de algoritmos simétricos, inclusive algoritmos atuais, que utilizam chaves de 128 bits ou maior. É importante salientar que os módulos especiais, que diferenciam este criptoprocessador, não são específicos para um determinado algoritmo de criptografia. Estes foram projetados de forma a serem pré-configurados de acordo com as características do algoritmo que será executado [10].

O NPSoC é um processador de rede, o Maté é um processador de rede específico para sensores, porém nenhum desses dois possui segurança, segurança essa que está contida no criptoprocessador. Com isso a pesquisa procurou desenvolver um processador que agregasse algumas características para no final ter-se um processador de rede específico para sensores com segurança, chamado PERS.

3. PERS – Um Processador Específico para Redes de Sensores

Com base nos trabalhos correlatos estudados, foi definido um conjunto reduzido de instruções que dá suporte ao funcionamento do processador através de seus registradores, ULA (Unidade Lógica Aritmética), a UC (Unidade de Controle), PC (Contador de Programa), RI (Registrador de Instruções) e as instruções específicas de roteamento que fazem o envio com segurança na transmissão dos dados, pois utilizamos o algoritmo de criptografia DES (*Data Encryption Standard*) [8].

O PERS é um processador simples com o propósito de coletar informações de um determinado local (por exemplo, temperatura) através de sensores, processá-las e enviá-las para a estação base para assim executar ações necessárias de acordo com as informações coletadas.

Sua arquitetura foi modificada em relação ao NPSOC [12] para se adequar à necessidade de funcionamento de uma rede de sensores, que possui características diferentes com relação a topologia, pois utilizamos a topologia ponto-a-ponto. Sua arquitetura ficou então definida como segue:

1. Uma porta de entrada;
2. Uma porta de entrada e saída;
3. Uma porta de saída;
4. Quatro registradores de propósito geral;
5. Oito registradores específicos de roteamento;

6. Quatro registradores específicos de criptografia;
7. Uma ULA (Unidade Lógica e Aritmética);
8. Contador de Programa (PC);
9. Unidade de Controle (UC);
10. Registrador de Instruções (RI).

Na figura 1 verifica-se que a Unidade de Controle controla todo o processador. Neste processador, usou-se o método de FSM (*Finite State Machine*), ou seja, máquina de estados finita, onde seus estados são bem definidos e respeitam o ciclo de busca, decodificação e execução dos computadores Von Neumann. O ciclo se executa da maneira seguinte:

Passo 1: É feita a busca de uma instrução. Imediatamente, soma-se 1 para o contador de programa.

Passo 2: Decodifica a instrução;

Passo 3: Se a instrução tem dados, eles são armazenados em registradores internos;

Passo 4: Executa a instrução;

Passo 5: Armazena os resultados em registradores internos; Retornando ao passo 1, ou seja, busca a próxima instrução, dando continuidade ao ciclo de execução.

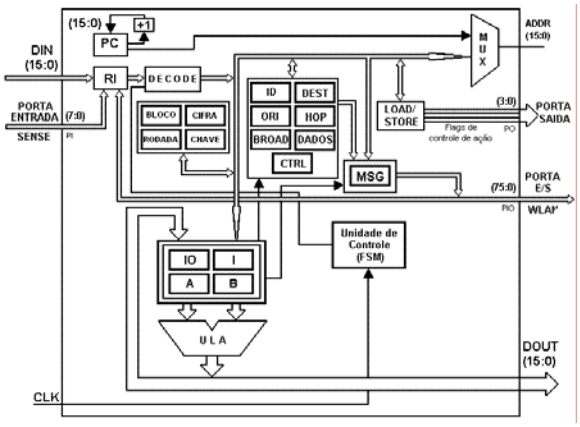

Figura 1. Arquitetura detalhada do PERS.

Observa-se na figura 1 que a arquitetura do processador possui três portas: uma de entrada que será ligada ao sensor; uma de entrada e saída que será ligada na rede; e outra de saída que vai tomar alguma decisão de acordo com a variável coletada pelos sensores.

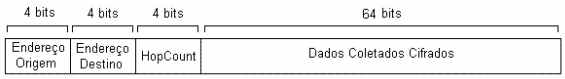

Figura 2. Composição da Mensagem de Dados.

A mensagem de dados que é transmitida pela rede dos sensores possui 76 bits, a figura 2 ilustra a composição da mensagem de dados.

Observa-se que são necessários 64 bits para a informação cifrada devido ao algoritmo criptográfico DES [8], 4 bits para o endereço origem, 4 bits para o endereço destino e 4 bits para o hopcount (quantidade de saltos). Essas informações são importantes para identificar quem está enviando e para qual destino a mensagem de dados será enviada. Este tamanho foi definido porque limitamos nossa rede em 16 nós sensores e com isso, necessitam de 4 bits para seu endereçamento.

A cifragem dos dados ocorrerá no momento que o sensor coletar a informação, ou seja, o nó origem vai coletar e cifrar os dados. Os dados permanecerão cifrados até que chegue à estação base onde será decifrado para análise da variável coletada. Esta instrução pode ou não ser usada pelo programador, através das instruções CRIPTA e DECRIPTA.

3.1 Conjunto de Instruções do PERS

O conjunto de instruções do Processador Específico para Redes de Sensores PERS segue o modelo RISC, ao todo são 30 instruções, e foi desenvolvido para efetuar transmissão de dados coletados por sensores e enviar para a rede utilizando a topologia ponto-a-ponto.

Importante salientar que as instruções do processador foram definidas após um estudo do Maté [6] que é uma máquina virtual para Redes de Sensores, analisando seus programas de envio e recebimento de mensagens e fazendo uma adaptação para a topologia Ponto-a-Ponto.

As instruções são necessárias para montagem do cabeçalho de dados utilizando endereços de origem e destino dos nós que compõem os sensores da rede, para controlar a quantidade de saltos efetuados pelos dados entre os nós sensores da rede, para desmembrar a mensagem de dados, retirando o endereço de origem e destino e também a informação coletada, bem como instruções para efetuar testes caso os sensores não tenham coletado informações de um determinado local ou ambiente.

Inclui-se flags de controle de aplicação de 4 bits para efetuar alguma ação caso o nó seja o destino da mensagem de dados e também instruções lógicas e aritméticas, de movimentação e de desvio. Por fim, instruções para enviar a mensagem de dados cifrada para a rede. Esses estudos nos levaram ao conjunto de instruções do PERS (ver na tabela 1).

As instruções OR, XOR, ADD, SUB são utilizadas para fazer operações básicas com registradores no processador e a instrução INV será utilizada para fazer *broadcast*, pois vai inverter o endereço destino para enviar para todos os nós da rede.

As instruções INC e DEC são utilizadas para incrementar e decrementar o *hopcount* que é o controle necessário para saber quantas vezes um determinado

Tabela 1 – Conjunto de Instruções do PERS.

Lógicas e Aritméticas		
01	AND	Compara dois registradores
02	OR	Compara dois registradores
03	XOR	Idem OR, porém nega o resultado
04	ADD	Soma dois registradores
05	SUB	Subtrai dois registradores
06	INV	Nega a constante destino (0000) e coloca no registrador BROAD
07	INC	Incrementa o HopCount
08	DEC	Decrementa o HopCount
Movimentação		
09	LDID	Atribui um valor para o registrador ID
10	LD	Atribui ao registrador DADOS o conteúdo (dados) da porta PIO
11	STR	Envia uma ação de controle para a porta PO
12	MOV	Atribui ao registrador A o conteúdo do registrador B
Desvio		
13	JMP	Desvio incondicional
14	JD	Desvia se o nó for o destino
15	JPES	Desvia se não existe pacote de dados no sensor
16	JPEW	Desvia se não existe pacote de dados na Rede
Especiais/Rede		
17	SENSE	Atribui ao registrador I o conteúdo da porta PI
18	WLAN	Atribui ao registrador IO o conteúdo da porta PIO
19	SET_ID	Seta no registrador MSG o conteúdo do registrador ID
20	SET_DEST	Seta no registrador MSG o destino do pacote
21	SET_HOP	Seta no registrador MSG o hopcount inicial do pacote
22	SET_SENSE	Seta no registrador MSG o conteúdo do registrador I
23	SET_ORI	Seta no registrador ORI o se conteúdo do registrador ID for maior
24	GET_DEST	Atribui ao registrador DEST o conteúdo do nó destino do registrador IO
25	GET_HOP	Atribui ao registrador HOP o conteúdo de saltos do nó do registrador IO
26	GET_ORI	Atribui ao registrador ORI o conteúdo do nó origem do registrador IO
27	CHK_DEST	Checa se o nó é o destino do pacote
28	CHK_DATA	Atribui ao registrador CTRL uma ação de controle
29	PUT	Envia o pacote de dados do registrador MSG para a porta PIO
30	PUT_LAN	Envia a mensagem do registrador MSG para a porta PIO
31	CRIPTA	Cifra o conteúdo da porta PI e atribui no registrador MSG
32	DECRIPTA	Decifra o conteúdo da porta PIO e atribui no registrador DADOS

pacote de dados foi transmitido de sensor para sensor.

A instrução LDID (*Load* ID) carrega o endereço do nó origem no registrador ID, necessário para compor a mensagem completa que será enviada pela rede. O endereço é encaminhado junto com a instrução. LD atribui direto ao registrador DADOS o conteúdo da porta de entrada e saída PIO (porta da rede onde os dados são coletados). A instrução STR atribui direto para a porta de saída PO o conteúdo do registrador CTRL e a instrução MOV atribui o valor do registrador B para o registrador A.

JMP, JD, JPES e JPEW se referem aos saltos no programa caso seja necessário. A instrução JMP faz um desvio incondicional, a instrução JD faz um desvio caso o nó seja o endereço destino do pacote e por fim, as instruções JPES (*Jump* de entrada dos sensores) e JPEW (*Jump* de entrada da Rede) fazem um desvio caso não exista informação disponível no sensor ou na rede respectivamente.

As instruções especiais/rede são utilizadas para o roteamento, recepção e transmissão de dados entre os registradores e portas de entrada e entrada/saída.

A instrução SENSE atribui ao registrador I (nome atribuído por ser um registrador que vai armazenar informação apenas de entrada, ou seja, *Input*) os dados que estão na porta PI que foram coletados pelos sensores.

A instrução WLAN atribui ao registrador IO (nome atribuído por ser um registrador que vai armazenar informação de entrada e saída, ou seja, *Input* e *Output*) os dados que estão na porta PIO e que foi transmitida pela rede.

As instruções que começam com SET montam a mensagem de dados que será transmitida na rede pela instrução PUT. As instruções que começam com GET desmembram as informações da mensagem de dados que foi transmitida na rede para saber se o nó é o destino e também incrementar o *hopcount* para depois ser remontada e enviada pela instrução PUT_LAN.

Importante salientar que o endereço destino da mensagem de dados é sempre o endereço “0000”, pois a finalidade de todos os nós é enviar a mensagem de dados para a estação base que vai analisar e processar as informações coletadas.

A instrução CRIPTA será utilizada para cifragem dos dados que são coletados pelos sensores de rede. Após a cifragem ele é enviado para o próximo nó até chegar ao destino, onde o dado será decifrado. A instrução DECRIPTA será utilizada para decifragem dos dados que são coletados na rede, caso o nó sensor seja o destino. Após a decifragem será tomada alguma ação de acordo com o dado coletado.

Por fim, a instrução CHK_DATA tem a função de analisar a variável coletada pelo sensor (ex. temperatura) e irá tomar alguma ação de acordo com as regras estabelecidas (conforme exemplo da tabela 2).

Tabela 2 – Tabela de controle de ação do PERS.

Ação	Flag	
1	0000	LED0 (Luz Verde)
2	0001	LED1 (Luz Amarela)
3	0010	LED2 (Luz Vermelha)
4	0011	Ar Condicionado
5	0100	Alarme
6	0101	Alarme de Ação

4. Resultados - Simulação e Prototipação

Foi utilizada a topologia Ponto-a-Ponto para analisar o desempenho do programa representado na tabela 3. O objetivo desta topologia é condicionar a porta de saída de um nó na porta de entrada de outro nó de modo seqüencial, exemplificando: o nó 16 só pode enviar para o nó 15 que por sua vez recebe o pacote e envia para o nó 14. Cada nó desta topologia representa um processador PERS.

Como pode-se observar na tabela 3, o programa descreve um laço que fica capturando dados em sensores; caso não exista dados disponíveis na porta que é ligado aos sensores é efetuado um salto para buscar dados na porta que é ligada à rede, pois pode ser que outro sensor tenha transmitido dados pela rede. Interessante observar que após a prototipação em FPGAs desse processador, todas as instruções, exceto a CRIPTA e DECRIPTA (com 19 ciclos), requerem dois ciclos para serem executadas, isso porque a Unidade de Controle trabalha com máquina de estado finita e a versão inicial de nossa implementação possui apenas os estados de busca e execução.

Importante salientar é a repetição por cinco vezes da instrução CHK_DATA no programa devido à necessidade de atender as regras de ações descritas na tabela 2, como foi dito, o programa deixa a cargo do usuário a definição das ações.

Com a definição do conjunto de instruções e do programa teste que simula uma topologia ponto-a-ponto, implementamos o processador na ferramenta Xilinx Foundation Series [14]. A seguir mostram-se algumas simulações das instruções implementadas. Por exemplo, na figura 2 se observa, através de simulação, o envio de um pacote de dados que foi coletado por um sensor. A primeira instrução simulada é a SENSE que atribui o valor da porta de entrada (PI) no registrador I, ou seja, armazena em um registrador específico os dados coletados pelo sensor. Depois se executa a instrução LDID que armazena no registrador ID o endereço do nó sensor origem, o endereço do ID vem acompanhado na instrução.

Tabela 3 – *Assembly* de comunicação do processador PERS.

	PERS	Ciclos
1	SENSE	2
2	CRIPTA	19
3	JPES, 08	2
4	LDID, 00	2
5	SET_ID	2
6	SET_DEST	2
7	SET_HOP	2
8	PUT	2
9	WLAN	2
10	JPEW, 00	2
11	LDID, 00	2
12	GET_DEST	2
13	CHK_DEST	2
14	JD, 20	2
15	GET_ORI	2
16	SET_ORI	2
17	GET_HOP	2
18	INC_HOP	2
19	PUT_LAN	2
20	JMP, 00	2
21	DECRIPTA	19
22	CHK_DATA, 20	2
23	CHK_DATA, 25	2
24	CHK_DATA, 30	2
25	CHK_DATA, 35	2
26	CHK_DATA, 45	2
27	STR	2
28	JMP, 00	2
	Total	90

A partir da terceira instrução que é a SET_ID, se dá início à montagem da mensagem de dados (76 bits), conforme demonstrado na figura 2, coloca-se o valor do registrador ID nos primeiros 4 bits do registrador MSG (que equivale à mensagem de dados). A instrução SET_HOP informa a quantidade de saltos que a mensagem de dados vai ter durante a transmissão, considerando que a primeira transmissão então vai começar com valor um. Depois a instrução SET_SENSE coloca o conteúdo do registrador I na mensagem, terminando de montá-la.

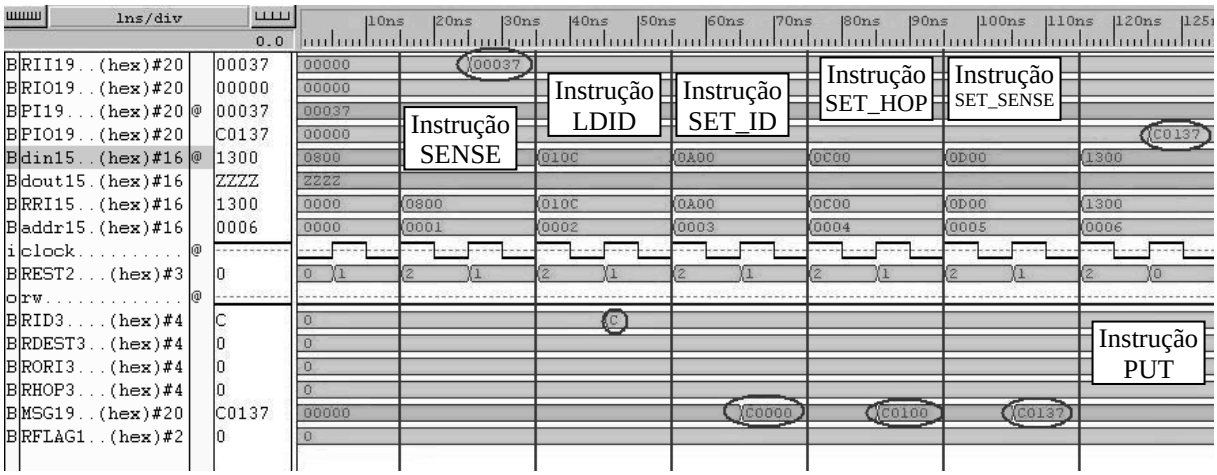

Figura 3. Simulação de envio de Mensagem de Dados.

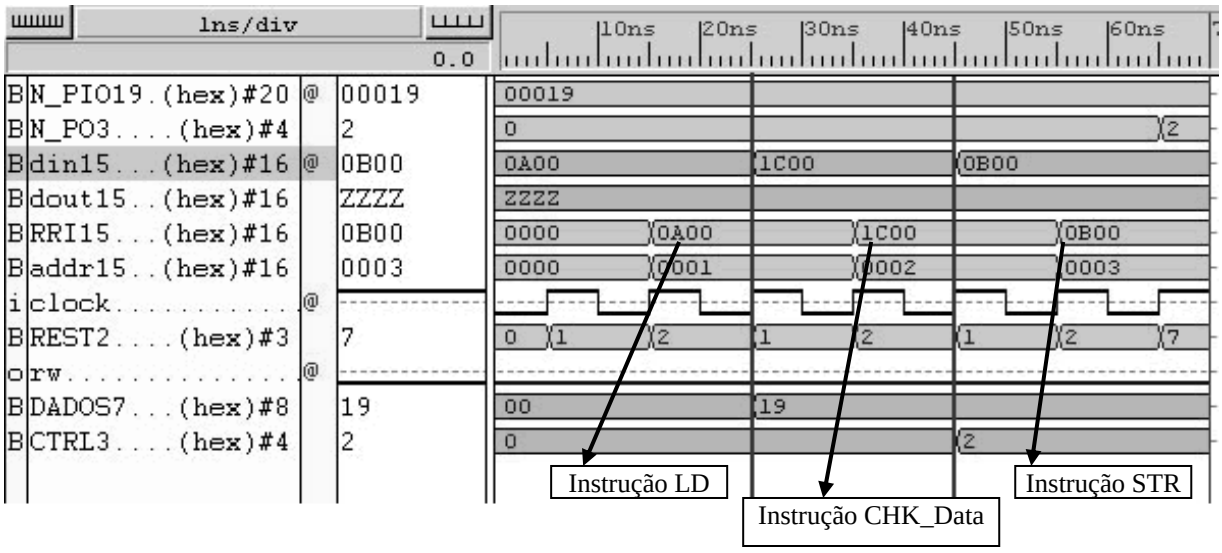

Figura 4. Envio de uma ação de controle.

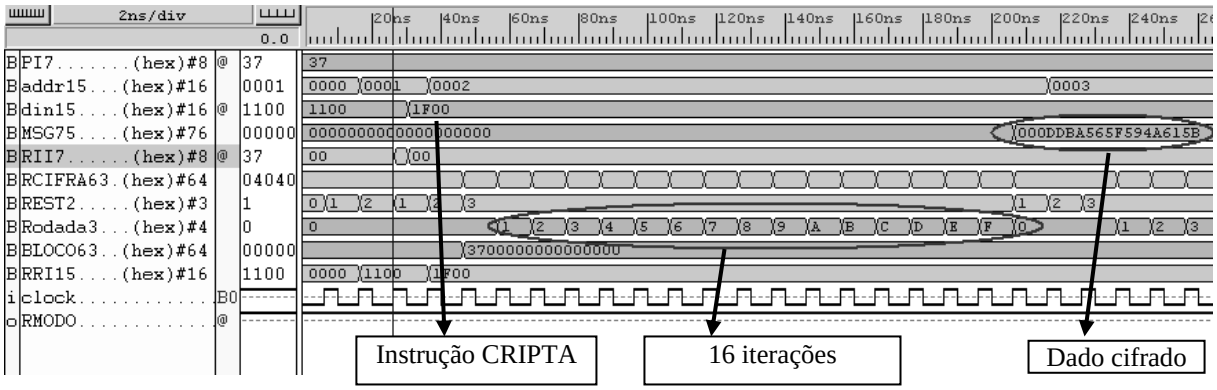

Figura 5. Envio de um dado cifrado.

Com a mensagem completa, composta pelo endereço do nó origem, endereço do nó destino, *hopcount* da mensagem e os dados da mensagem, a instrução PUT envia a mensagem de dados para a porta de entrada e saída (PIO), que é a porta da rede.

Na figura 3 se verifica a simulação do envio de uma ação de controle após analisar a variável (ex. temperatura) coletada por um sensor. A primeira é a instrução LD que atribui o valor da porta de entrada (PIO) no registrador DADOS, depois a instrução CHK_DATA analisa a variável (ex. temperatura) coletada e atribui no registrador CTRL um valor associado que consta na tabela 2, finalizando, a instrução STR envia o conteúdo do registrador CTRL para a porta de saída (PO).

Na figura 4 se observa a simulação da cifragem de um dado após ser coletado pelo sensor de um nó da rede. O dado cifrado é armazenado no registrador MSG que depois será preparado com o cabeçalho (endereço do nó origem, endereço do nó destino e *hopcount*) para ser enviado ao próximo nó da rede.

A seguir, apresenta-se um detalhamento das instruções que foram simuladas nas figuras 2-4, a saber: LDID, SENSE, SED_ID, SET_HOP, SET_SENSE, PUT, LD, CHK_DATA, STR, CRIPTA e DECRIPTA, descritas em VHDL e prototipadas em um FPGA:

```
when ldid => ID <= RI(3 downto 0);
             EST <= busca;
when sense => I <= PI;
             EST <= busca;
when set_id => MSG(19 downto 16) <= ID;
             EST <= busca;
when set_hop => MSG(11 downto 8) <= count;
             EST <= busca;
when set_sense =>
             MSG(7 downto 0) <= I(7 downto 0);
             EST <= busca;
when put => PIO <= MSG;
             EST <= inicializa;
when ld => DADOS <= PIO(7 downto 0);
             EST <= busca;
when chk_data =>
             if DADOS < "00010101" then
                          CTRL <= "0000";
             elsif DADOS < "00011000" then
                          CTRL <= "0001";
             elsif DADOS < "00011010" then
                          CTRL <= "0010";
             elsif DADOS < "00011111" then
                          CTRL <= "0011";
             elsif DADOS < "00100100" then
                          CTRL <= "0100";
             elsif DADOS < "00101001" then
                          CTRL <= "0101";
             else
                          EST <= inicializa;
             end if;
             EST <= busca;
when str => PO <= CTRL;
             EST <= inicializa;
when cripta => BLOCO <= I&PAD;
             EST <= CRIPTO;
```

Como o algoritmo DES já estava implementado em VHDL [10], tivemos que adicioná-lo como um componente do processador PERS, usando o código que o declara como um componente:

```
component des
        port (clk     : in  std_logic;
        modo    : std_logic;
        iteracao : std_logic_vector(3 downto 0);
        K       : std_logic_vector(55 downto 0);
        din     : std_logic_vector(63 downto 0);
        saida   : out std_logic_vector (63 downto 0));
end component;
```

Para finalizar, é preciso chamar o componente declarado no código do PERS, usando a seguinte declaração:

```
U1:DES port map (Clock, MODO, Rodada,
                 CHAVE, BLOCO, CIFRA);
```

Após a implementação, síntese e teste do processador PERS foram analisados os resultados obtidos através da ferramenta Xilinx Foundation Series [14], que oferece informações quanto ao tempo de propagação do circuito e freqüência máxima do processador, assim como o espaço ocupado pelo circuito no FPGA. A prototipação em FPGA foi realizada usando a placa Spartan 2 – XC2S200E, e os resultados estão contidos na tabela 4.

Tabela 4 – Estatísticas Espaciais do PERS.

Spartan 2 XC2S200E	**Total**	**Ocupado**	**%**
CLBs	2352	757	32
Flip Flops	4704	88	1
Luts de 4 entradas	4704	1386	29

É bom lembrar que a utilização dos recursos do FPGA são dados através de [9]:

(i) **CLB** *(Configurable Logic Bloc*k): Matriz de blocos lógicos configuráveis, unidade lógica de uma FPGA; (ii) **Flip Flop**: Circuito digital básico que armazena um bit de informação, sua saída só muda de estado durante a transição do sinal do *cloc*k; (iii) **Luts:** Muitos FPGAs modernos são modelados como tabelas de busca *(lookup table* - LUTs) programáveis. A LUT constitui a configuração de cada elemento da matriz.

Comparando os resultados da versão final com a primeira versão desenvolvida no início da pesquisa (sem criptografia), verificamos um aumento de 23% de CLBs e Luts de 4 entradas devido à inclusão de criptografia. A medida temporal do processador, quanto ao tempo de propagação no circuito é de 15,969 ns o que corresponde a uma Freqüência Máxima do Processador de 62,621 MHZ, enquanto a versão inicial

sem criptografia foi de 11,801 ns = 84,739 MHZ. Assim, podemos observar então que a inclusão de criptografia influenciou diretamente no desempenho do PERS, pois apresentou uma queda significativa de 26 % na freqüência do processador. Esta queda é significativa, porém é irrelevante considerando a importância de se inserir segurança na rede de sensores.

5. Conclusões e Trabalhos Futuros

A grande dificuldade encontrada neste trabalho foi obter um nível de segurança que impeça os diversos tipos de ataques existentes nas redes de sensores sem que isso prejudique a vida útil dos mesmos, devido às suas limitações de recursos, principalmente, a limitação de energia.

As dificuldades encontradas durante este trabalho serviram de estímulo para a proposta e implementação do nosso PERS um Processador Específico para Redes de Sensores com Primitivas de Segurança.

Os dados de desempenho demonstrados e analisados mostraram que o processador proposto (PERS) sofreu um grande impacto com a inclusão do algoritmo de criptografia DES, porém a inclusão da criptografia é necessária para que os dados transmitidos estejam seguros de algum tipo de ataque.

Este impacto correspondeu na diminuição de 26% na freqüência do processador, porém a inserção de segurança na transmissão dos dados torna esta queda irrelevante.

Temos vários trabalhos futuros que visam o melhoramento de nosso processador, porém a seguir listamos os mais importantes: (i) inclusão de outros algoritmos de segurança (por exemplo, RC5 e AES); (ii) suporte do processador para outras topologias existentes (por exemplo, árvore binária e anel unidirecional), com isso, será necessário incluir novas instruções de roteamento para efetuar *multicast* e *broadcast* na rede; (iii) diminuição no envio da mensagem com os dados de 76 bits para 20 bits. Para isso será necessário quebrar a mensagem em 4 ou 5 pacotes, pois em cada pacote de dados sempre deverá conter o endereço de origem e destino. Com esta divisão precisa-se ter uma atenção especial para o sincronismo dos pacotes de dados.

Referências

[1] H. C. Freitas, e C. A. P. S. Martins, “Projeto de Processador com Microarquitetura Dedicada para Roteamento em Sistemas de Comunicação de Dados”, Workshop em Sistemas Computacionais de Alto Desempenho, Ouro Preto, 2000.

[2] H. C. Freitas, e C. A. P. S. Martins, “Processador de Rede com Suporte a Multi-protocolo e Topologias Dinâmicas”, Workshop em Sistemas Computacionais de Alto Desempenho, Ouro Preto, 2001.

[3] H. C. Freitas, e C. A. P. S. Martins, “R2NP: Processador de Rede RISC Reconfigurável”, Workshop em Sistemas Computacionais de Alto Desempenho, Ouro Preto, 2002.

[4] J. S. H. Heidemann, R. Govindan, C. Intanagonwiwat, D. ESTRIN, e D. GANESAN, “Building efficient wireless sensor networks with low-level naming”, In Proceedings of the Eighteenth ACM Symposium on Operating Systems Principles, Banff, Alberta, Canada, ACM Press, 2001, pp. 146-159.

[5] L. Hu, e D. Evans, “Secure aggregation for wireless networks”, In Workshop on Security and Assurance in Ad hoc Networks, Virginia, 2003.

[6] P. Levis, e D. Culler, “Mate’ – a Virtual Machine for Tiny Networked Sensors”, In Proceedings oh the 10th International Conference on Architectural Support for Programming Languages and Operating Systems, San Jose, California, 2002, pp. 85-95.

[7] A. A. F. Loureiro, J. M. S. Nogueira, B. R. Linnyer, e A. F. Raquel, “Redes de Sensores Sem Fio”, Mini Simpósio Brasileiro de Computação, Jornada de Atualização à Informática, Florianópolis, 2002.

[8] Moreno, E. D., F. D. Pereira, e R. B. Chiaramonte, Criptografia em Software e Hardware – Implantação e Desempenho, Editora Novatec, Marília, 2005.

[9] Moreno, E. D., F. D. Pereira, C. G. Penteado, e R. A. Pericini, Projeto, Desenvolvimento e Aplicações de Sistemas Digitais em Circuitos Programáveis (FPGAs), Editora Bless, Marília, 2003.

[10] F. D. Pereira, “Um Criptoprocessador VLIW para Algoritmos Criptográficos Simétricos”, Dissertação de Mestrado em Ciência da Computação do PPGCC da UNIVEM, Marília, 2004.

[11] A. Perrig, R. Szewczyk, V. Wen, D. Culler, e J. D Tygar, “SPINS: security protocols for sensor networks”, In Proceedings of the Seventh Annual International Conference on Mobile Computing and Networking, Rome, Italy, ACM Press, 2001, pp. 189-199.

[12] R. P. Prado, NPSoC – Arquitetura e Protótipo de um Novo Processador de Rede, Dissertação de Mestrado em Ciência da Computação do PPGCC da UNIVEM, Marília, 2004.

[13] J. A. Stankovic, “A network virtual machine for real time-coordination”, Networked Embedded Software Technology Kickoff Meeting, Napa, Canadá, 2001.

[14] Xilinx Development Systems, Synthesis and Simulation Design Guide – Designing FPGAs with HDL, 1998.