VOICE RECORDER: PROJETO, VALIDAÇÃO E PROTOTIPAÇÃO DE UM IP PARA COMPRESSÃO DE ÁUDIO

Gabriel Marcilio, Emilio Wuerges, Luiz C. V. dos Santos

Universidade Federal de Santa Catarina
Florianópolis, SC, Brasil
{gaheris, emilio, santos}@inf.ufsc.br

ABSTRACT

Within the platform-based design paradigm, IP reuse and ESL modeling are key mechanisms to cope with SoC complexity and time-to-market. To achieve proper productivity gains, IP models are required in different levels of abstraction. This paper describes the design, validation and prototyping of an IP in three distinct abstraction levels. The IP, called VoiceRecorder, implements compression and decompression algorithms based upon the DPCM technique. Experimental results are reported not only for validation purposes but also for evaluation of productivity gains at higher levels of abstraction.

1. INTRODUÇÃO

O projeto de Systems-on-Chip (SoCs) baseado em plataforma [2] requer o reuso de IPs e a modelagem em níveis de abstração cada vez mais altos, como o nível de sistema (ESL). Para atingir ganhos de produtividade adequados, os IPs de uma plataforma deveriam ser modelados em diferentes níveis de abstração. Modelos em níveis mais altos de abstração são usados nas fases iniciais do projeto (por exemplo, para a exploração do espaço de projeto), enquanto modelos menos abstratos são usados nas fases finais (por exemplo, para a síntese).

Este trabalho descreve a modelagem de um IP em três níveis distintos de abstração. O IP, denominado Voice Recorder realiza a compressão e a descompressão de mensagens de áudio utilizando a técnica DPCM [1] (*Differential Pulse Coded Modulation*). O projeto do IP é realizado através de refinamentos sucessivos a partir de uma descrição algorítmica, passando por uma descrição funcional, até chegar a uma descrição RTL sintetizável, que viabiliza a rápida prototipação.

O restante deste artigo é organizado da seguinte forma. A Seção 2 descreve os algoritmos utilizados para compressão e descompressão. A Seção 3 descreve o projeto e a validação do IP, enquanto que a Seção 4 descreve sua prototipação. Os resultados experimentais são apresentados na Seção 5. A Seção 6 resume as conclusões e aponta direções para investigações futuras.

2. A TÉCNICA DE COMPRESSÃO

A idéia básica da técnica DPCM pode ser assim resumida: se a taxa de amostragem do sinal for suficientemente alta, sua amplitude não deverá variar muito entre amostras consecutivas. Assim, ao invés de se codificar o valor absoluto de cada amostra, codifica-se a *diferença* entre os valores de amostras consecutivas.

No IP projetado adotou-se a representação em 8 bits para as amostras e a representação em 4 bits para as diferenças. Assim, a compressão consiste na codificação DPCM das amostras do sinal de áudio, enquanto que a descompressão consiste na decodificação DPCM das diferenças.

O algoritmo de compressão adotado é mostrado na Figura 1. As variáveis `current` e `previous` (linhas 1 e 2) armazenam, respectivamente os valores da amostras corrente e anterior. A variável `difference (linha 3)` armazena o valor da diferença entre as duas amostras mais recentes. Os valores dessas três variáveis são representados em 8 bits.

A variável `compressed` (linha 4) deveria em princípio armazenar o valor da variável `difference` codificado em apenas 4 bits. Entretanto, pode ocorrer que aquele valor não seja representável usando apenas os quatro bits menos significativos da variável `difference`. Neste caso, o valor residual não representável em quatro bits é armazenado na variável `offset` (linha 5) para ser compensado na codificação da amostra seguinte.

As funções `readSample()` e `saveSample()` invocadas nas linhas 8 e 13, respectivamente, realizam a leitura e escrita de amostras sucessivas na memória. A função `existMoreSamples()`, invocada na linha 6, retorna um valor verdadeiro se há amostras pendentes a serem processadas e, em caso contrário, retorna falso.

Depois da inicialização (linhas 1 a 5), um laço de iteração (linhas 6 a 14) processa todas as amostras de uma mensagem. No corpo do laço, a amostra corrente é lida e, em seguida, calcula-se o valor da diferença a ser comprimida (linhas 8 e 9). O valor da diferença, adicionado ao valor residual da iteração anterior, é truncado em quatro bits (linha 10). O valor do resíduo não representável nesta iteração é então calculado (linha 11). Ao final, o valor da amostra comprimida é armazenado em

memória e a amostra corrente torna-se a amostra anterior para a próxima iteração (linhas 12 e 13).

```
1) int current = 0;
2) int previous = 0;
3) int difference = 0;
4) short compressed =0;
5) int offset = 0;
6) while( existMoreSamples( ) )
7){
8)     current = readSample();
9)     difference = current – previous;
10)    compressed = difference + offset;
11)    offset = offset + difference –
       compressed;
12)    previous = current;
13)    saveSample(compressed);
14) }
```

Figura 1: Algoritmo para compressão

A descompressão é mais simples do que a compressão, uma vez que ela não tem que realizar compensação alguma. O algoritmo de descompressão adotado para o IP é mostrado na Figura 2. O papel das variáveis e funções nele utilizadas são similares às descritas para o algoritmo de compressão.

Depois da inicialização (linhas 1 a 3), o laço de iteração (linhas 5 a 10) processa todas as amostras de uma mensagem. O valor da amostra corrente é obtido através da soma do valor da amostra anterior com o valor da diferença (linhas 6 e 7). Em seguida a amostra descomprimida é armazenada em memória e a amostra corrente torna-se a amostra anterior para a próxima iteração (linhas 8 e 9).

```
1)    int difference = 0;
2)    int previous =0;
3)    int current =0;
4)    while(existMoreSamples( ))
5)    {
6)      difference = readSample();
7)      current = previous +
        difference;
8)      saveSample(current);
9)      previous = current;
10)}
```

Figura 2: Algoritmo para descompressão

3. PROJETO E VALIDAÇÃO DO IP

3.1 Metodologia do Projeto

Para o projeto do IP, adotou-se uma *metodologia top-down* [2]. Esta metodologia consiste na elaboração de uma versão em alto nível de abstração do sistema desejado, e a partir desta, realizar refinamentos sucessivos até se obter uma versão sintetizável do IP. Os níveis de descrição utilizados neste trabalho são: algorítmico, funcional e RTL (SystemC e VHDL).

3.2 Modelagem do sistema

O IP foi projetado para ser usado como componente de um sistema embarcado cuja aplicação requeira compressão, descompressão e armazenamento de áudio. Por simplicidade, o sistema foi modelado como se tivesse apenas dois blocos: o IP projetado, denominado `VoiceRecorder`, e o bloco denominado `Application`, que encapsula os demais componentes do sistema.

As principais funções do bloco `Application` são: amostragem de mensagem proveniente do meio externo; monitoração da operação solicitada pelo usuário; envio de amostras uma a uma para o `VoiceRecorder`; especificação da operação solicitada (compressão ou descompressão), aquisição de amostras processadas pelo VoiceRecorder e reconstrução de mensagem.

Ao final do projeto, o `VoiceRecorder` será realizado em hardware e o bloco `Application` será implementado em software para ser executado na CPU da plataforma de prototipação.

3.3 Descrições do IP

As descrições em todos os níveis foram validadas, utilizando o mesmo conjunto de estímulos. Dada uma descrição do IP em um determinado nível de abstração, os resultados aos estímulos a ela aplicados devem corresponder aos resultados obtidos no nível imediatamente superior, tomado como referência e previamente validado.

3.3.1 Descrição Algorítmica

Nesta descrição o bloco `VoiceRecorder` é implementado (em linguagem C++) na forma de uma classe onde são encapsulados os algoritmos das Figuras 1 e 2 e estruturas de dados para armazenamento das amostras.

3.3.2 Descrição Funcional

A descrição funcional do IP (em SystemC) consiste essencialmente de um módulo `VoiceRecorder` que, por sua vez, é composto de três módulos: `Formatter`, `Controller` e `Memory`, conforme ilustra a Figura 3.

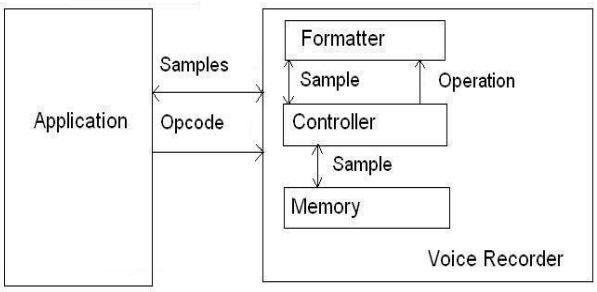

Figura 3: A estrutura do IP inserido no sistema

O módulo Memory armazena as amostras, tanto no formato comprimido, quanto no descomprimido.

O módulo `Formatter` implementa os algoritmos mostrados nas Figura 1 e 2. Dada uma amostra (`sample`) e um sinal de controle (`operation`), o módulo `Formatter` realiza a operação selecionada (compressão ou descompressão) e devolve a amostra processada (`sample`).

O módulo `Controller` despacha amostras provenientes da aplicação para armazenamento no módulo `Memory` e comanda a transferência de amostras para o módulo `Memory` e vice-versa, bem como comanda a operação do módulo `Formatter` .

3.3.3 Descrição RTL

Esta descrição representa o sistema digital síncrono que implementa o bloco `VoiceRecorder`, com precisão de ciclos e de bits (4 e 8 bits para amostras comprimidas e descomprimidas, respectivamente).

Como resultado, nela é modelado o protocolo de comunicação entre os módulos `Controller`, `Memory` e `Formatter`, requerendo a criação de canais adicionais entre eles para sincronizar a comunicação. Cada um dos módulos `Controller` e `Formatter` é decomposto em termos de um *datapath* e uma unidade de controle.

Ademais, inseriu-se um novo módulo para prover a interface entre o `VoiceRecorder` e o barramento da plataforma de prototipação.

Uma primeira descrição RTL foi escrita em SystemC, como refinamento da descrição funcional. Em seguida, essa descrição foi manualmente traduzida para VHDL de forma a viabilizar a prototipação, uma vez que não se dispunha de ferramentas de síntese a partir de SystemC.

4. PROTOTIPAÇÃO

4.1 Plataforma Alvo

Para implementar o IP a plataforma utilizada foi uma placa APEX 20k da ALTERA, utilizando a porta serial para enviar e receber dados da placa. A CPU utilizada foi o NIOS I[3] e o barramento utilizado foi o AVALON. Nessa plataforma, verificou-se que o protótipo pode operar à freqüência máxima de 80 MHz.

4.2 Driver de Acesso ao IP

A interação do bloco `Application` com o IP é realizada através de um *driver* de acesso, o qual possui dois métodos, *Compress* e *Decompres,* descritos a seguir.

Dada uma mensagem a ser comprimida, composta por palavras de 8 bits, o método *Compress* envia uma amostra por vez ao `VoiceRecorder`. Somente quando todas as amostras tiverem sido enviadas, o método envia o comando de compressão para o IP. Quando o IP sinaliza o final da compressão, o método lê a mensagem já comprimida e armazenada no módulo `Memory`, disponibilizado-a para a aplicação.

Dada uma mensagem a ser descomprimida, composta por palavras de 4 bits, o comportamento do método *Decompress* é análogo ao anterior, exceto por comandar uma operação de descompressão.

5. RESULTADOS EXPERIMENTAIS

5.1.Configuração experimental

As simulações das descrições foram realizadas em um Athlon XP, operando a 1.4 GHz. Para a modelagem e compilação utilizaram-se SystemC-2.0.1 e GCC-3.3.5. As simulações VHDL foram feitas na ferramenta Quartus-4.2.

5.2. Procedimento de validação

Para validar a funcionalidade do IP, utilizaram-se duas mensagens distintas: a primeira com 256 amostras de 8 bits; a segunda com 256 amostras de 4 bits. A correção do IP foi verificada comandando a compressão da primeira mensagem e, em seguida, sua descompressão para verificar se a mensagem original foi recuperada. Um procedimento dual foi realizado para a segunda mensagem.

Este procedimento foi repetido para todas as descrições e os resultados esperados foram obtidos em todos os casos testados.

5.3. Avaliação do desempenho relativo

Para se avaliar o ganho de produtividade potencial ao se usar modelos de mais alto nível de abstração, foram medidos os tempos de compilação e de simulação de cada descrição, conforme mostra a Tabela 1.

Note que o tempo de compilação da descrição Funcional é cerca de 3 vezes menor do que o da descrição RTL (SystemC), enquanto o tempo de simulação é cerca de 2 vezes menor.

O tempo para a síntese do protótipo foi da ordem de 35 minutos e o tempo de execução dos testes foi de 0,084s, para os mesmos estímulos utilizados nas simulações.

Tabela 1: Desempenho relativo das descrições

Descrição	Tempo de Compilação [s]	Tempo de simulação [s]
Algorítmica (C++)	2,68	0,008
Funcional (SystemC)	23,9	0,014
RTL (SystemC)	86,3	0,032

6. CONCLUSÕES E TRABALHOS FUTUROS

Os resultados do procedimento de validação fornecem evidências experimentais da corretude do IP projetado. Os tempos de compilação e simulação das descrições permitem estimar a ordem de grandeza do ganho de produtividade potencial obtido com a utilização de níveis mais abstratos de modelagem. Por exemplo, para o

VoiceRecorder descrito em SystemC, o ganho de produtividade entre os níveis funcional e RT é da ordem de 3 vezes.

Pretende-se disponibilizar o IP VoiceRecoder ao domínio público no futuro próximo. Para isso, será preciso submetê-lo a um processo de certificação preliminar à sua integração em repositório de domínio público, como por exemplo o do Projeto BrazilIP [4] ou o do Projeto ArchC[5], versão 2.0.

REFERÊNCIAS

[1] WOLF, Wayne Hendrix. *Computers as components: principles of embedded computing system design*. MK Publishers, 2001.

[2] SANGIOVANNI-VINCENTELLI, A.; MARTIN, G. "*Platform-Based Design and Software Design Methodology for Embedded Systems"*. IEEE Design & Test of Computers, v. 18, n.6, p.23-33, 2001.

[3] *NIOS Embedded Processor. System Development*. http://www.altera.com/products/ip/processors/nios/.

[4] *BrazilIP*. Disponível em http://www.brazilip.org.br

[5] *The ArchC Architectural Description Language*. http://www.archc.org.