高位合成可能なゲームライブラリの開発に関する一検討

山方 悠輝^a 山脇 彰^a

^a九州工業大学 工学府 電気電子工学専攻

1. はじめに

モバイル端末のバッテリー寿命を延ばすために、消費電力を削減できる処理のハードウェア化は、有望な手段である(1)。しかし、モバイル・アプリケーションごとに個別のカスタム・ハードウェア・モジュールを用意することは不可能である。FPGA のような動的再構成可能デバイス (2) は、アプリケーションに合わせて回路を動的に変更できるためモバイル端末の性質に適合する。

ゲームアプリケーションは高電力を要するモバイル・アプリケーションの 1 つである。研究では、ゲーム・ソフトウェア・ライブラリとして、FPGA で動的再構成可能なハードウェアモジュール群を開発し、ゲーム実行時におけるモバイル端末の省電力化を図る。

開発には、関数の追加や変更柔軟に対応できるよう、C プログラムをハードウェアに自動変換する高位合成 (HLS) (3) ツールを使用する。本稿では、矩形描画を題材に、HLS ツールが高性能ハードウェアモジュールを生成できるように、C 言語レベルでの記述法を調べる。

C 言語レベルでの記述において、HLS ツールが、矩形内を描画中に、メモリ書き込みをバースト転送にマッピングするように試みる。これによって、矩形の画素をメモリに連続的に生成することにより高性能を達成しうる。

2. 再構成可能なハードウェアを持つモバイル端末

モバイル端末は、コンテンツ配信サービスから様々なアプリケーションをダウンロードして、実行する。提案モバイル端末は、コンテンツ配信サービスから、ソフトウェアコードのみを含む従来のソフトウェアアプリケーションに加えて、ハードウェアコードをダウンロードする。ソフトウェアコードは、プロセッサで実行されるプログラムで、ハードウェアコードは、再構成部のハードウェアモジュールに構成する回路データである。ソフトウェアとハードウェアの協調動作によってモバイル端末の高性能化・省電力化を達成する。

3. 矩形描画関数

3.1 純粋なソフトウェア

図 1 は、矩形描画を実現するためのソフトウェアコー

```
1: // DW is display width
2: DrawRect(*f,x1,y1,x2,y2,col,f1){
3:   if(f1){ //filled rect
4:     for(y=y1*DW;y<=y2*DW;y+=DW)
5:       for(x=x1;x<=x2;x++) f[x+y]=col;
6:   }else{ //blancked rect
7:     for(y=y1*DW;y<=y2*DW;y+=DW){
8:       if(y==y1*DW || y==y2*DW)
9:         for(x=x1;x<=x2;x++) f[x+y]=col;
10:      else f[x1+y]=f[x2+y]=col;
11:    }
12:  }
13: }
```

図 1. 矩形描画関数の純粋ソフトウェア

ドである。引数の **f** は、表示されるフレームメモリへのポインタである。**x1** と **y1** は、矩形の左上頂点座標で、**x2** と **y2** は矩形の右上頂点座標である。**col** は、表示する矩形の色で、**f1** は、矩形が指定された色で塗りつぶされているかどうかを示すフラグである。

f1 を 1 に設定すると、**col** で塗りつぶされた矩形がフレームメモリに描画され、**f1** を 0 に設定すると、空白の矩形がフレームメモリに描画される。空白の矩形は、**col** で示された線で囲まれている。

```
1: // DW is display width
2: // BL is burst length
3: HLS_DrawRect(*f,x1,y1,x2,y2,col,f1){
4:   if(f1){ //filled rect
5:     FL1:for(i=y1;i<=y2;i++){
6:       idx_i=DW*i+x1;
7:       FL2:for(j=0;j<(x2-x1+1)/BL;j++){
8:         idx_j=idx_i+j*BL;
9:         // to infer burst transfer
10:        FL3:for(k=0;k<BL;k++)f[idx_j+k]=col;
11:      }
12:      idx_i+=((x2-x1+1)/BL)*BL;
13:      // draw the remaining pixels
14:      FL4:for(j=0;j<(x2-x1+1)%BL;j++){
15:        f[idx_i+j]=col;
16:      }
17:    }else{ // blancked rect
18:      BL1:for(i=y1;i<=y2;i++){
19:        idx_i=DW*i+x1;
20:        if(i==y1||i==y2){
21:          BL2:for(j=0;j<(x2-x1+1)/BL;j++){
22:            idx_j=idx_i+j*BL;
23:            // to infer burst transfer
24:            BL3:for(k=0;k<BL;k++)f[idx_j+k]=col;
25:          }
26:          idx_i+=((x2-x1+1)/BL)*BL;
27:          // draw the remaining pixels
28:          BL4:for(j=0;j<(x2-x1+1)%BL;j++){
29:            f[idx_i+j]=col;
30:          }else f[idx_i]=f[DW*i+x2]=col;
31:        }
32:      }
33:    }
```

図 2. HLS 向けの矩形描画関数

A Consideration to Develop a High-level Synthesizable
Software Game Library Yuki Yamagata and Akira
Yamawaki Kyushu Institute of Technology

3.2 HLS 用ソフトウェア

図 2 は、HLS がバースト転送を推論するために再記述した矩形関数を示す。この関数の引数は図 1 と同じである。HLS にバースト転送を推論させるために、ループ数が一定のループ文 (FL3、BL3) を追記した。これらのループは、バースト長 BL で繰り返されるメモリ書き込み操作を実現する。また、FL4、BL4 のループでは、画素数をバースト長で除算したときの残りの画素もフレームメモリに格納する。したがって、この機能は、画素数をバースト長で除算できない場合であっても、任意の数の画素を扱うことができる。

4. 実験および考察

4.1 実験環境

実機実験のために、図 3 に示すハードウェアプラットフォームを開発した。このプラットフォームには、HLS ハードウェア HLS_HW をカバーする HLS_wrap というラッパーがある。このラッパーにより、HLS ハードウェアはプラットフォーム上で簡単に交換できる。ハードウェアプラットフォームは、FPGA ボード、Digilent ZYBO に搭載されている Xilinx ZYNQ FPGA 上に構築されていて、ARM Cortex-A9 のエンベデッドプロセッサは 650MHz で、ハードウェアモジュールは 100MHz で動作する。

記述したソフトウェアは Xilinx SDK2016.4 で GCC (-O2) によってコンパイルされる。また、Vivado HLS 2016.4 を使用して HLS を実施した。Vivado HLS で生成されたものを含む VHDL プログラムは Vivado 2016.4 によって回路データに変換され、実装される。

HLS の前に、適切なループにパイプライン用のプラグマとハードウェアサイズを大幅に大きくするループのフラット化をオフにするプラグマを挿入した。

4.2 結果および考察

測定した実行時間を図 4 に示す。PSW は、組み込みプロセッサ上の図 1 に示す純粋なソフトウェアの実行時間である。HW_nB は、図 1 に示す純粋なソフトウェアから生成された HLS ハードウェアの実行時間である。HW_B16 は、図 2 に示す再記述されたソフトウェアから生成された HLS ハードウェアの実行時間でありバースト長が 16 のバースト転送を推論するものである。

図 3. ハードウェアプラットフォーム

図 4. 測定した実行時間

図 4 は、純粋なソフトウェアをバースト転送が推論できる記述に変えることにより、3.8 倍の性能を改善できることを示している。また、純粋なソフトウェアから生成されたハードウェアは、ソフトウェア実行のパフォーマンスよりも劣る。この事実は、HLS ツールが構造化されたハードウェアモジュールを生成するためには、C ソフトウェアレベルでのハードウェア特性の深い考察が非常に重要であることを示している。

5. 結論

モバイル機器のバッテリー寿命を延ばすために、高い計算負荷をハードウェアで実現することは、消費電力を大幅に削減するための有望な方法の 1 つである。再構成可能なデバイス上のハードウェアの動的再構成は、多くの種類のアプリケーションを動的に実行するモバイル端末の性質を満たすのに適した技術であると思われる。

HLS は、ハードウェア開発を簡単かつ迅速に行う有望な技術である。本論文では、HLS ツールを使用して高性能ハードウェアモジュールを生成できるように、高位合成可能なゲームライブラリの開発にあたって、C 言語での矩形描画機能の記述方法について検討した。実験結果は、我々の記述方法の有効性が、バースト転送を推論することによって、純粋なソフトウェア記述と比較して大幅に性能を改善できることを示している。

文 献

- (1) J. M. Mora-Gutiérrez, C. J. Jiménez-Fernández, and M. Valencia-Barrero, "Multiradix Trivium Implementations for Low-Power IoT Hardware", IEEE Transactions on Very Large Scale Integration (VLSI) Systems, Vol. 25, No. 12, pp. 3401-3405, 2017.
- (2) Anju P. Johnson, Sikhar Patranabis, Rajat Subhra Chakraborty, and Debdeep Mukhopadhyay, "Remote dynamic partial reconfiguration: A threat to Internet-of-Things and embedded security applications", Micro-processors and Microsystems, Vol. 52, pp. 131-141, 2017.
- (3) A. Canis, J. Choi, M. Aldham, V. Zhang, A. Kammoona, J.H. Anderson, S. Brown, and T. Czajkowski, "LegUp: High-Level Synthesis for FPGA-based Processor/Accelerator Systems", In ACM FPGA, pp. 33-36, 2011.