プログラマブルSoCを用いた 自動運転システムのためのプラットフォームの構築

～ZybotR2-Z2の開発～

工藤　裕也[†]　高田　厚志[†]　津田　壮士[††]　堺　拓実[††]　泉　知論[†,††]

† 立命館大学 大学院 理工学研究科 電子システム専攻
†† 立命館大学 理工学部 電子情報工学科
E-mail: †t-izumi@se.ritsumei.ac.jp

あらまし　自動運転システムのFPGAによる組込み実装を目指し，プログラマブルSoCを活用した情報処理プラットフォームを構築する．カメラI/Fから動画像処理フロントエンドまでをプログラマブルロジック(PL)上に実現し，プロセッサ(PS)や外部メモリを介さずデータ転送の効率化をはかる．その間は標準的なストリームインターフェースを採用して，高位合成を活用した画像フィルタの修正や追加を容易にする．複雑なアルゴリズムはプロセッサ上のソフトウェアとして実装できる．DMAで主記憶上に転送されたフレームバッファに対し，ソフトウェアの実行速度に応じた任意のフレームレートで処理できる．テンプレートマッチングやモーター制御は，PLとPSの連携により処理の高速化・効率化をはかる．本稿では，指定された経路を車線を遵守して走行する自動運転システムを実装した．Xilinx Zynq XC7Z020上で3～30frame/sec程度の処理性能を達成した．

キーワード　FPGA，自動運転，HW/SW協調設計，高位合成，画像処理，リアルタイム処理，Zynq

1. まえがき

近年，自動運転技術が注目を集めている．自動運動のレベルは運転支援，部分的自動運転，条件付き自動運転，高度自動運転，完全自動運転の5段階で定義されている．現状は衝突しそうな状況を判断し自動ブレーキをかけるなどの運転支援のみで，システムが自律的に運転する自動運転には至ってない．自動運転には，人間と同様の認知，判断，操作が求められ，センサーデータに対する高度な情報処理と走行制御が課題となる．システムとしては，状況の変化に対する応答時間を担保するリアルタイム性と，小型，低消費電力，低価格を実現する高効率な実現方法が求められる．

自動運転にはカメラを用いた画像認識は必須であるが，画像処理・画像認識は演算負荷が高く，一般的な組込みプロセッサでは大幅なフレームレートの低下・レイテンシの増大を招き，リアルタイム性を損なう．そこで，我々はプログラマブルSoCを用いた高効率な自動運転システムの実現を目指す．

プログラマブルSoCは，組込みプロセッサ(PS)とプログラマブルロジック(PL)の両方をワンチップに搭載し，それらを密に連携させることができるデバイスである．一般的にはPS上でソフトウェアとして実現する機能群について，その一部をハードウェアアクセラレータとしてPLにオフロードすることで高速化をはかることができる．しかし，機能のいくつかを単純にオフロードするだけでは，PSあるいは外部メモリとPLとのデータ転送がオーバーヘッドとなり充分な効率向上が得られない．そこで，我々はシステム全体の処理の流れ・データの流れを考慮して各種処理・データバッファをPS, PL, および外部メモリに配置し，プラットフォームとして構築する．

データ量の多いカメラI/Fから動画像処理フロントエンドまでをPL上に実現し，PSや外部メモリを介さずデータ転送の効率化をはかる[1]．その間，標準的なストリームインターフェースを採用して，高位合成を活用した画像フィルタの修正や追加を容易にする[2]．複雑なアルゴリズムはプロセッサ上のソフトウェアとして実装し，DMAで主記憶上に転送されたフレームバッファに対し，ソフトウェアの実行速度に応じた任意のフレームレートで処理できる．テンプレートマッチングやモーター制御は，PLとPSの連携により処理の高速化・効率化をはかる．

最終的には，

（1）　路面の認識，経路決定，走行の制御

（2）　歩行者・障害物・信号等の認識，それに基づく行動決定

を実装する計画であるが，本稿では（1）を対象とし，指定された経路を車線を遵守して走行する自動運転システムを実装した．Xilinx Zynq XC7Z020上で3～30frame/sec程度の処理性能を達成した．この設計をテンプレートとして機能の修正・追加を行っていく．

2. 車体の構成

我々が開発している自動運転ロボット ZybotR2-Z2 を図 1 に，主要部品を表 1 に示す．

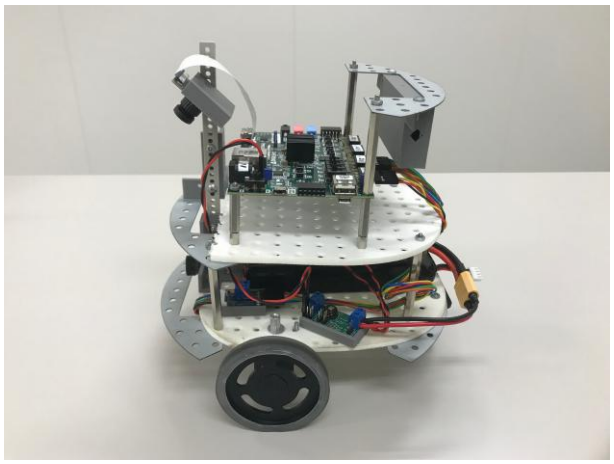

図 1　ZybotR2-Z2

表 1　使 用 部 品

Name	Model number	Manufacture	Quantity
FPGA board	Zybo Z7-20	Digilent	1
CMOS image sensor	Pcam 5C	Digilent	1
OLED display	OLED_DRV-S1	KYOHRITSU	1
Battery	RB-Sta-15	RobotShop	1
Voltage regulator	VRM(Rev. B)	Digilent	1
H bridge	Pmod HB5	Digilent	2
Motor	IG220053X0085R	Digilent	2
Wheel	Wheel Kit	Digilent	2
Tire	Sticky Rubber Tires	Digilent	2

これは Xilinx 社のプログラマブル SoC である Zynq を中核とする研究教育用のロボット・カー Zybot-R [3] を改良したもので，主な変更点は，ターゲットボードを Zybo から Zybo Z7-20 への換装とイメージセンサー Pcam 5C の採用である．車体は軽量化のためアルミフレームからプラスチックフレームに変更し，カメラをより高くやや下向きに配置することで路面と光軸の角度を大きくして白線を認識しやすくするとともに，反射光の影響を軽減している．移動は左右車輪と摺動棒で行う．車輪モータは Zybo の Pmod コネクタに接続した H ブリッジモジュール HB5 で駆動する．モータはギア (1:53) とロータリーエンコーダを装備している．また，デバッグ用の文字表示器を搭載し Pmod コネクタで接続する．電源は 3cell 11.1V のリチウムポリマーバッテリを用い，直接モーターに供給するとともにレギュレータ VRM で 5.0V に降圧して Zybo に供給する．Pmod には Zybo 内のレギュレータで生成された 3.3V が供給される．

3. プラットフォームの構成

本プラットフォームで使用している Zybo Z7-20 には，Xilinx 社のプログラマブル SoC である Xilinx Zynq XC7Z020 が搭載されている．プログラマブル SoC の特徴として，プロセッサと FPGA の両方を扱えるという点の他に，PS-PL 間のデータ通信を，ユーザが自由に設計出来るという点が挙げられる．本プラットフォームは，この特徴を活かした最適な処理のフローの構築を狙いとしている．図 2 に本プラットフォームのシステムアーキテクチャを示す．

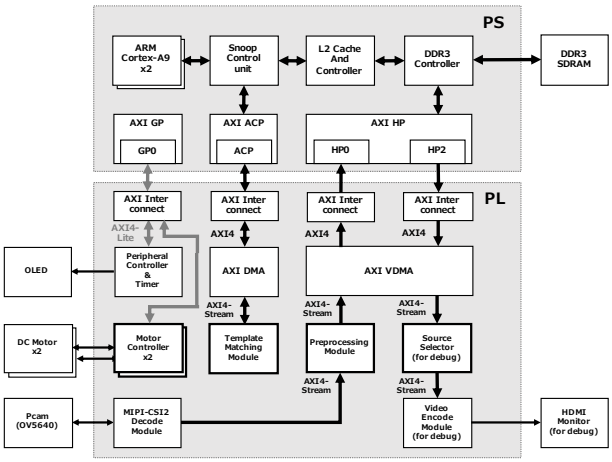

図 2　システムアーキテクチャ

本プラットフォームでは，CMOS イメージセンサやモータ等の主要部品をすべて PL 側に接続し，映像データの前処理やモータ制御を PL 側で行うことで，CPU の負荷を低減しつつ，メモリアクセスの回数を削減することを達成した．以下にシステムアーキテクチャの要点を説明する．

はじめに，映像データに関して，本プラットフォームでは AXI VDMA が映像データの PS-PL 間のデータ転送を行っている．CMOS イメージセンサからのデータは，図 2 内の MIPI-CSI2 Decode Module によって AMBA 規格の AXI4-Stream の Video 形式にデコードされた後，AXI VDMA によって DRAM に転送される．AXI VDMA はトリプルバッファリングによって映像データの書き込みと読み出しを同時に行っているため，デバッグ用の Source Selector および Video Encode Module は，CMOS イメージセンサからの入力と同様のフレームレートで映像データをモニタに出力することが可能となっている．MIPI-CSI2 Decode Module，Video Encode Module の構成，および AXI VDMA の制御に関しては，Digilent 社の Elodg 氏が作成した Zybo Z7-20 Pcam 5C Demo Project [4] を参考とされたい．

次に，モータ制御に関して，本プラットフォームでは図 2 内の Motor Controller が各モータのフィードバック制御を行っている．モータの角速度を目標値とし，モータに付随している 2 相クロックエンコーダの出力信号と，その角度分解能から導出した実測角速度を用いて目標値との偏差を導出している．ここでは，出力値を PWM 信号のパルス幅とし，また，フィードバック制御器には PID 制御器を採用した．SW から PID ゲイン，サンプリングレート，目標角速度を指定するだけでモータを操作することが可能となっている．

最後に，本プラットフォームは，デバッグ機能として時間計測用タイマと OLED ディスプレイを搭載している．これらの制御は，図 2 内の Peripheral Controller & Timer が行っている．

図 2 内の PL 部に存在するすべてのモジュールは，AXI4-Lite バスによって SW から制御することが可能である．本プラットフォームでは，これらのモジュールをより扱いやすくするために抽象度の高い API を用意している．

4. 自動走行システム

路面認識，歩行者・障害物・信号認識の結果を用いて適切な経路決定および行動決定を行い，車体の制御をそれに従って行うような自動走行システムの実装を我々は計画している．本章では，現段階で実装済みの路面認識アルゴリズムに関して説明する．

本システムの路面認識は，前処理，路面標識認識，車道外側線推定，ライントレースから成り立っている．CMOS イメージセンサから取得する画像サイズは 1280x720 であり，路面標識認識，ライントレースでは，これを 1/10 に縮小した画像データを使用する．

我々は前処理は 2 段階に分けており，1 段階目を Canny エッジ検出 [5] と二値化，2 段階目を座標変換（歪曲収差補正 [7] [10]・射影変換 [8] [10]・縮小処理）としている．

1 段階目は HW で実行している．映像データが DRAM に転送される前に，図 2 内の Preprocessing Module で処理が行われる．このモジュールは高位合成技術を用いて実装しており，Canny エッジ検出 [6] と二値化の処理を並列に実行する．我々は，それぞれの処理結果を特定のビットに格納し，同時に DRAM へ転送することでデータフローの効率化を図った．

2 段階目は SW で実行している．広角カメラレンズ特有の歪みを補正する歪曲収差補正，路面の平面図を得るための射影変換，後処理のための縮小処理を一度のラスタスキャンで行う．我々は，座標変換のテーブルを事前に作成することで処理の高速化を実現した．図 3 に入力画像と前処理後の画像の例を示す．

路面標識認識では，テンプレートマッチングを用いて路面標識の種類と位置を取得する．図 4 に今回使用している 8 種類のパターン画像を示す．

本システムでは，テンプレートマッチングの処理を HW で行っている．採用したテンプレートマッチングのアルゴリズムは，対象領域とパターン画像の SAD(Sum of Absolute Difference) を導出し，その値を用いてパターン画像と類似度の高い領域を探索するというものである．今回，対象画像とパターン画像がどちらも二値化画像であるため，パターン画像のピクセル数に等しい回数の XOR 演算で，対象画像の座標あたりの SAD が導出される．

本システムでは，パターン画像のピクセル数に等しい回数の XOR 演算を並列実行するテンプレートマッチングの回路を高位合成技術を用いて実装し，更に，この回路をパターン数だけ並列で動作させることで処理の高速化を図った．

車道外側線推定では，直線ハフ変換 [9] を用いて車道外側線の位置，車体に対する角度を推測する．本システムでは，直線

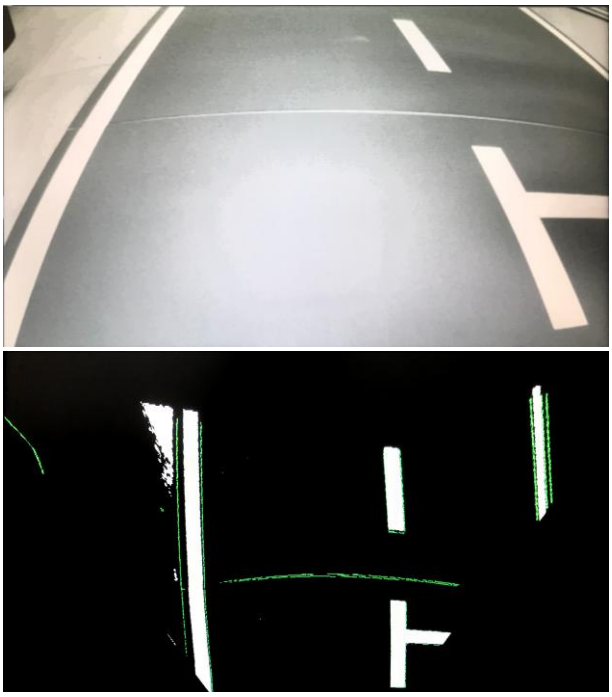

図 3　入力画像と前処理後の画像の例

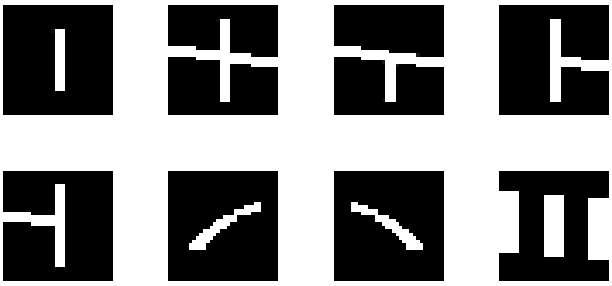

図 4　パターン画像 (32x32pix)

ハフ変換でエッジ画像を $\rho\theta$ 平面上に投影した後，直線のブレを考慮するためにローパスフィルタを実行する．その後，$\rho\theta$ 平面において，投票数が多く，ρ の差が車道の幅程度であり，かつ，θ の差が 0 に近い座標の組を探索することで車道外側線を推定する．

ライントレースでは，二値化画像のラベリング [10]，白線領域の推定，直線の導出を行っている．白線領域の推定では，ラベリング処理と同時に，各ラベル領域の横方向の重心，平均幅を導出し，前フレームで判定したライン領域の情報と照らしあわせ，もっとも白線らしい領域を推定している．直線の導出では，白線領域を一定間隔で縦方向にサンプリングし，最小二乗法で回帰直線を導出している．図 5 にライントレースの実行結果の例を示す．この例は，カーブ直前におけるライントレースの様子であり，赤い点がサンプリング点，赤い直線が最小二乗法で導出した回帰直線である．

路面認識の処理フローに関して，本システムでは，通常時は路面標識認識とライントレースのみを実行しており，発進時や交差点右左折時にのみ車道外側線推測を実行している．

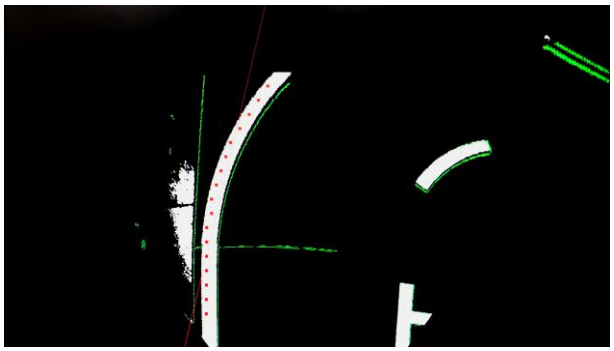

図 5　ライントレース実行結果の例

5. 性 能 評 価

本章では，３章で説明したプラットフォームのシステムアーキテクチャ，および４章で説明した路面認識アルゴリズムに関する性能評価について述べる．我々は，開発環境として Xilinx 社の Vivado Design Suite HLx Edition 2017.4 を使用している．表 2 に本システムのリソース使用量を，表 3 に各路面認識アルゴリズムの平均処理時間を示す．

表 2　リソース使用量 (使用率)

Module Name	LUT	FF	BRAM	DSP
Preprocessing Module	2042(3.84%)	3012(2.83%)	12(8.57%)	11(5.00%)
Template Matching Module	16231(31.0%)	20867(19.6%)	4(2.86%)	0(0.00%)
Motor Controller x2	3756(7.06%)	3884(3.65%)	0(0.00%)	56(25.5%)
Other	10938(20.6%)	14830(13.9%)	15.5(11.1%)	0(0.00%)
Total	32967(62.0%)	42593(40.0%)	31.5(22.5%)	67(30.5%)

表 3　平均処理時間

Name	SW	HW	SW/HW
Preprocessing (Canny edge detection, Binarization)	-	6.367ms	-
Preprocessing (Coordinate transformation)	25.73ms	-	-
Template matching (Eight times)	246.4ms	0.1244ms	1981
Outside line estimation	241.2ms	-	-
Line trace	0.6378ms	-	-

表 3 に関して，ここでは直進の路面に対する 5 回の実行の平均時間を示している．計測にはプラットフォームとして実装したデバッグ用のタイマを使用している．テンプレートマッチングの結果には，SW での実行結果も記載している．テンプレートマッチングの結果に関して，HW へのオフロードによって 2000 倍近い高速化が達成できていることが分かる．

路面認識の処理性能に関して，通常時の路面認識処理では，前述の通り路面標識認識とライントレースのみを行う．これより，通常時は約 30frame/sec の処理性能となる．この処理性能に関して，座標変換が大きなボトルネックとなっているため，我々は，座標変換処理の高速化が今後の課題であると認識している．

また，発進時や交差点右左折時に車道外側線推定を実行した際の路面認識の処理性能は約 3frame/sec である．この処理性能に関して，車道外側線推定を実行する際，車体は停止か徐行の状態であるため，路面認識システムとしてはリアルタイム性を満たすといえる．

6. ま と め

本稿では，自動運転システムのプラットフォーム，および，我々が計画している自動運転システムの実装の一部として路面認識について述べた．

我々が構築したプラットフォームは，プログラマブル SoC の特徴を活かすことにより，効率的なデータ転送を行う構成となっている．路面認識においては，計算量の多い処理の HW へのオフロードによって，リアルタイム性を満たす程度の高速化を実現した．

今後，歩行者・障害物・信号等の認識機能の実装を行っていく予定である．また，路面認識のアルゴリズムに関しても，深層学習を用いた，より堅牢性の高いアルゴリズムを新たに採用することを考えている．

文　献

[1] Yuya Kudo, Atsushi Takada, Tomonori Izumi, "a Prototype Platform on All Programmable SoC for Autonomous Moving Robots," Proceedings of the 62nd Annual Conference of the Institute of Systems, Control and Information Engineers (ISCIE), Kyoto, May 16-18, 2018.

[2] Atsushi Takada, Yuya Kudo, Tomonori Izumi, "An Evaluation of Image Processing Utilizing Libraries for FPGA High-Level Synthesis," Proceedings of the 62nd Annual Conference of the Institute of Systems, Control and Information Engineers (ISCIE), Kyoto, May 16-18, 2018.

[3] Zybot-R, `http://www.ritsumei.ac.jp/se/re/izumilab/lecture/16zybot/0_ZybotR.html`.

[4] Zybo Z7-20 Pcam 5C Demo Project, `https://github.com/Digilent/Zybo-Z7-20-pcam-5c`.

[5] John Canny, “ A Computational Approach to Edge Detection, ” IEEE Transactions on Pattern Analysis and Machine Intelligence, pp.679-698, 1986.

[6] Akira Yamawaki, Seiichi Serikawa, “ A describing method of an image processing software in C for a high-level synthesis considering a function chaining, ” IEICE trans. inf. & syst., vol.E101-D, no.2, February 2018.

[7] Jason P. de Villiers, F. Wilhelm Leuschner, Ronelle Geldenhuys, “ Centi-pixel accurate real-time inverse distortion correction, ” International Symposium on Optomechatronic Technologies, 2008.

[8] O. Chum, T. Pajdla, P. Sturm, “ The Geometric Error for Homographies, ” Computer Vision and Image Understanding, pp.86-102, 2005.

[9] Richard O. Duda, Peter E. Hart, “ Use of the Hough Transformation to Detect Lines and Curves in Pictures, ” Association for Computing Machinery, pp.11-15, 1972

[10] 奥富正敏編, “ ディジタル画像処理 [改訂新版], ” 公益 財団法人 画像情報教育振興協会 (CG-ARTS 協会), 2015.