ハードウェアのモード制御を記述可能な小規模組込みシステム向けFRP言語

瀧本哲史・森口草介・渡部卓雄（東京工業大学 情報理工学院）

XStorm[1]

- 小規模組込みシステム向けFRP言語
  - 時間変化する値（時変値）を組み合わせてプログラムを構築
- 状態遷移系に基づいた状態依存動作の簡潔な記述が可能
- ドライバ(C++)に記述した入出力コールバックをランタイムが呼び出し

```
type GpsMode = On | Sleep

switchmodule GPSClock {
  in  tick: Bool, gpsData: GpsData
  out time(zeroAM): Time, gpsMode: GpsMode
  init Tick

  state Tick { # タイマを用いて時刻管理
    out node time = if tick then add1s(time@last) else time@last
    node invalid = getTime(gpsData)
    out node gpsMode = if time == zeroAM then On else Sleep
    switch: if time == zeroAM then Adjust else Tick }

  state Adjust { # GPSを用いて時刻合わせ
    out node time = getTime(gpsData)
    out node gpsMode = Sleep
    switch: Tick } }
```

XStorm側

FRPにおける周辺機器のモード制御の現状

- **従来のFRP言語ではモード制御の直接的な記述が不可能**
- **バグを発生させやすい対処法を強いられる**

例題：省電力なGPS時計システム
- 普段はタイマーを用いて時刻を管理
- GPS情報を用いて定期的に時刻合わせ
- 時刻合わせをしない時はGPSモジュールをスリープさせ省電力化

XStorm側からフラグを出力，それをもとにドライバで適切なモード制御を行う必要がある

GPSモジュールがスリープの時にGPS情報にアクセスするコードがコンパイルエラーにならない

```
GpsMode currGpsMode = SLEEP;

void input(int *tick, GpsData *gpsData) { # 入力コールバック
  *tick = readTick();
  if (currGpsMode == ON) *gpsData = fetchGpsData(); }

void output(Time *time, GpsMode *gpsMode) {# 出力コールバック
  displayTime(time);
  if (currGpsMode == SLEEP && *gpsMode == ON) wakeupGps();
  if (currGpsMode == ON && *gpsMode == SLEEP) sleepGps();
  currGpsMode = *gpsMode; }
```

ドライバ側

XStormによるGPS時計システムの実装

提案手法：周辺機器のモード制御を言語レベルでサポートしたFRP言語

モード制御を言語に組込みの言語機構とすることで，モード制御処理の自動生成や機器への不正なアクセスの静的な防止を可能にした

XCios：入出力機器のモード制御を表現可能なXStormの拡張
- 状態遷移系の状態単位で入出力機器のモードの切り替えが可能
- 注釈に基づいてモード制御処理を含んだランタイムを生成

モード型：
- 機器がとるモードと対応する列挙型，入出力の型を修飾
- 各モードにおける値の参照の可否を設定

モード注釈：状態ごとの各入出力のモードを指定

フック：各入出力に指定されたモードが変わった際にランタイムから適切に呼び出されるモード制御用コールバック

```
mode Gps = Sleep | accessible On

switchmodule GPSClock {
  in  tick: Bool, gpsData: 'Gps GpsData
  out time(zeroAM): Time
  init Tick

  state Tick with gpsData >= Sleep {
    out node time = if tick then add1s(time@last) else time@last
    # node invalid = getTime(gpsData)
    switch: if time == zeroAM then Adjust else Tick }

  state Adjust with gpsData >= On {
    out node time = getTime(gpsData)
    switch: Tick } }
```

XCios側

gpsDataにSleepモードが指定されているのでgpsDataの参照はコンパイルエラーになる！

ランタイム側が適切にコールバックを呼び分けるので，周辺機器のモード制御漏れや不正なタイミングでの機器へのアクセスの心配無し！

```
void hook_gpsData_Sleep_to_On() { wakeupGps(time); }
void hook_gpsData_On_to_Sleep() { sleepGps(time); }

void input_tick(int *tick) { *tick = readTick(); }
void input_gpsData(GpsData *gpsData){ *gpsData = fetchGpsData(); }
void output_time(Time *time) { displayTime(time); }
```

ドライバ側

XCiosによるGPS時計システムの実装

関連研究

Energy types[2]
- XCiosと似た言語機構を用いて期待するエネルギーの消費度合いを明示的に注釈
- 注釈をもとにDVFSのコードを自動挿入やエネルギー消費度合いに不整合がないことの検査を実施

今後の課題

- DVFS等の入出力機器以外の周辺機器のモード制御
- 自律的にモードを変える周辺機器への対応
- 処理系の実装・ランタイムのオーバーヘッドの評価

ぜひ研究へのご意見下さい！
似ている研究や言語機構教えてください！

[1] 松村, 渡部: 組込みシステム向けFRP言語における状態依存動作のための抽象化機構,情報処理学会論文誌プログラミング(PRO), Vol. 13, No. 2(2020),pp. 1–13.
[2] M. Cohen, et al.: Energy types, OOPSLA '12, pp. 831-850.

日本ソフトウェア科学会 第26回プログラミングおよびプログラミング言語ワークショップ (PPL2024), Mar. 2024