Haskellの楽しみ

田中英行
<tanaka.hideyuki@gmail.com>
第0回スタートHaskell 2011/07/24

自己紹介

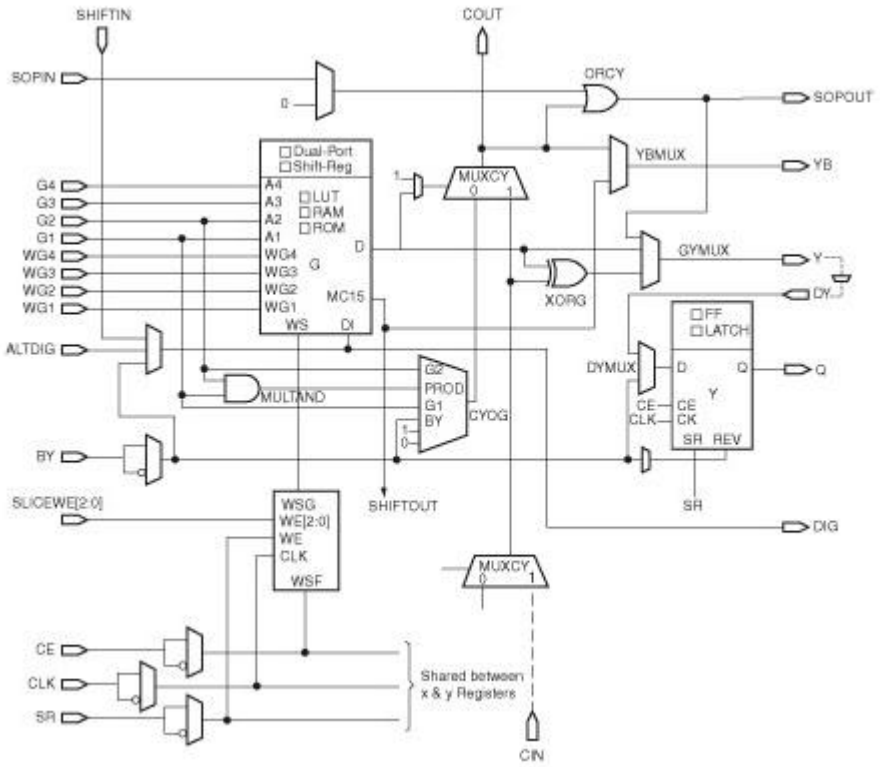
- 田中英行 (a.k.a @tanakh, id:tanakh)
- ㈱Preferred Infrastructure(PFI)勤務
- 競技プログラミング愛好家
  - ICPC , ICFP Contest, GCJ, TopCoder元赤
- Haskell愛好家
  - MessagePack Haskell実装
  - Monadius現メンテナ

Haskellの楽しみ

- Haskellの楽しげなアプリ・ライブラリをご紹介

Lava(1)

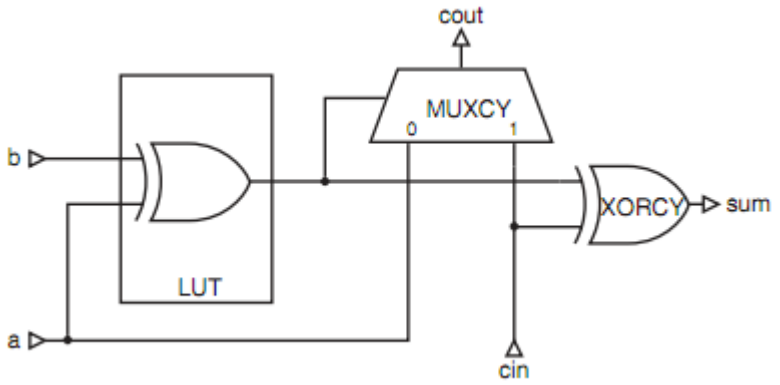
- ハードウェア記述言語
- http://www.raintown.org/lava/
- 実際にxilinx等で利用されている



Industrial HASKELL GROUP

Lava(2)

```
module Main
where
import Lava
import Xilinx

------------------------------------------------------

oneBitAdder :: (Bit, (Bit, Bit)) -> (Bit, Bit)
oneBitAdder (cin, (a,b))
  = (sum, cout)
    where
    part_sum = xor2 (a, b)
    sum = xorcy (part_sum, cin)
    cout = muxcy (part_sum, (a, cin))

------------------------------------------------------

oneBitAdderCircuit :: (Bit, Bit)
oneBitAdderCircuit
  = (outputBit "sum" sum, outputBit "cout" cout)
    where
    a = inputBit "a"
    b = inputBit "b"
    cin = inputBit "cin"
    (sum, cout) = oneBitAdder (cin, (a,b))

------------------------------------------------------

main :: IO ()
main
  = do nl <- netlist "tutorial1" oneBitAdderCircuit
       writeNetlist nl virtex2 [vhdl, edif]
```



Haskore(1)

- http://www.haskell.org/haskellwiki/Haskore
- http://hackage.haskell.org/package/haskore
- 音楽ライブラリ
  - 作曲
  - 演奏
  - 操作

Haskore(2)

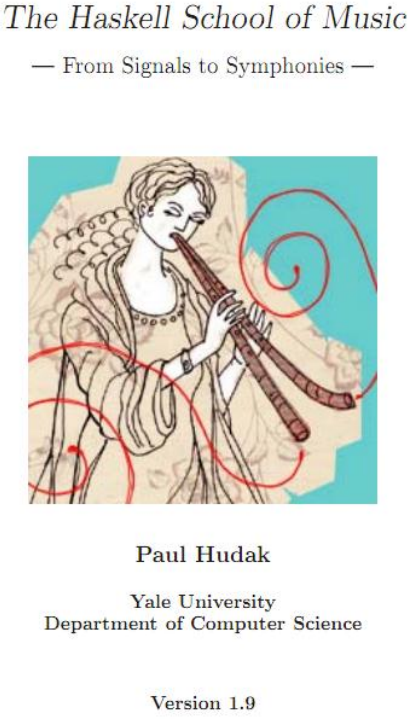
- デモ
  - http://www.youtube.com/watch?v=d2JvOwS26Zg
  - http://video.google.com/videoplay?docid=5849699036632847795

```
chords =
    (c 0 qn () =:= e 0 qn () =:= g 0 qn ()) +:+
    (c 0 qn () =:= f 0 qn () =:= a 0 qn ()) +:+
    (d 0 qn () =:= g 0 qn () =:= b 0 qn ())

song =
    MidiMusic.fromMelodyNullAttr MidiMusic.AcousticGrandPiano
       (Music.transpose (-48) (Music.changeTempo 3 chords))
```

The `(=:=)` means parallel composition and `(+:+)` means serial composition of musical objects.

Haskore(3)

- The Haskell School of Music
  - 本が出るようです
  - http://plucky.cs.yale.edu/cs431/reading.htm

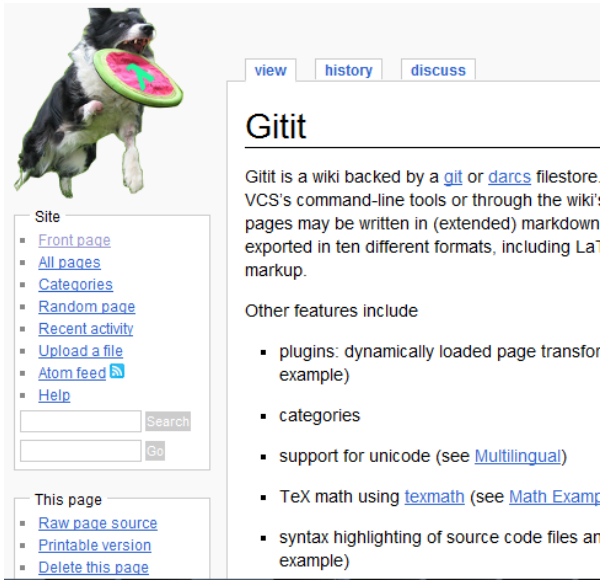


Pandoc(1)

- http://johnmacfarlane.net/pandoc/
- 様々なフォーマットに対応したドキュメントコンバータ
  - Input: markdown, reST, textile, HTML, LaTeX
  - Output: markdown, reST, textile, HTML, LaTeX, ConTeXt, PDF, RTF, DocBook XML, OpenDocument XML, ODT, GNU Texinfo, Media Wiki markup, groff man pages

Pandoc(2)

- 利用例
  - Gitit (http://gitit.net/)
  - Git/darcsをバックエンドにしたWiki
  - ドキュメント変換機能を組み込み
    - 記事を様々なフォーマットで記述
    - 記事を様々なフォーマットで取得

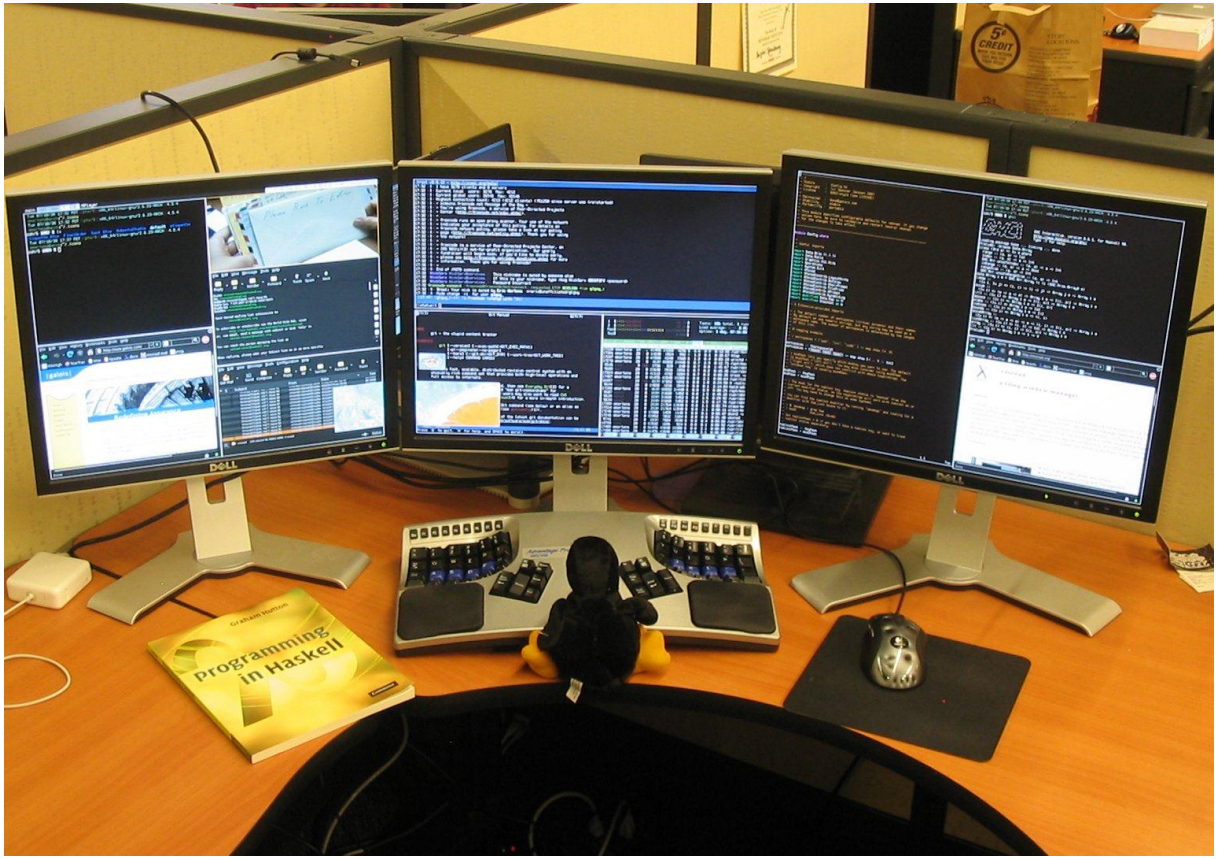


Xmonad(1)

- http://xmonad.org/
- タイル型WindowManager
  - 割と人気があるらしい
- Haskellで書かれてる
- ソースが小さく良く検証されている
- Haskellで設定ファイルが書ける

Xmonad(2)

- http://www.youtube.com/watch?v=AyNkBLhIpQk



Yesod(1)

- http://www.yesodweb.com/
- HaskellのフルスタックWebフレームワーク

Yesod(2)

- いいところ
  - 安全 → TypeSafeURL, no-XSS, no-SQL inject, …
  - 高速 → LLよりずっとはやい！
  - 記述力 → Template Haskellなどの活用
- 最近の技術にも対応
  - MongoDBバックエンド
  - BrowserID
  - Herokuへのデプロイ

Yesod(3)

- アプリ例
  - http://www.haskellers.com/
  - http://floating-winter-480.herokuapp.com/
  - https://hachicode.com/

Repa(1)

- http://repa.ouroborus.net/
- 並列配列
- 自動的に並列化
- 画像処理などに便利

```
mmMult :: (Num e, Elt e)
       => Array DIM2 e
       -> Array DIM2 e
       -> Array DIM2 e

mmMult a b = sum (zipWith (*) aRepl bRepl)
    where
        t    = transpose2D b
        aRepl = extend (Z :.All :.colsB :.All) a
        bRepl = extend (Z :.rowsA :.All :.All) t
        (Z :.colsA :.rowsA) = extent a
        (Z :.colsB :.rowsB) = extent b
```

Repa(2)

- デモ

Paraiso(1)

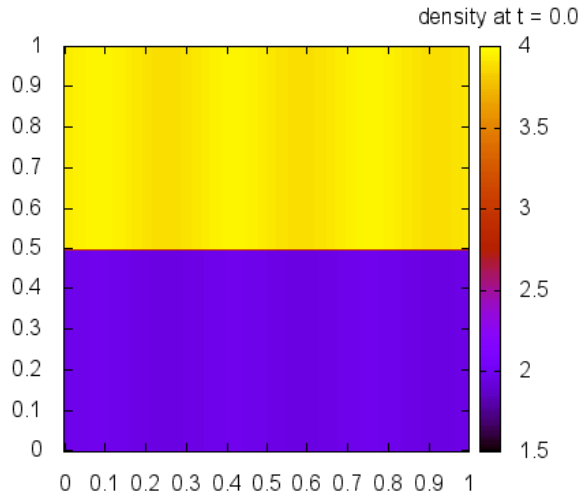
- 分散＋GPGPUコード生成DSL
  - http://d.hatena.ne.jp/nushio/20110515
  - http://hackage.haskell.org/package/Paraiso
  - Monadiusの作者作

Paraiso(2)

- 流体計算

ありのままの姿で書かかれた2次のルンゲ・クッタ法を！

```
interpolate order i cell = do
  let shifti n =  shift $ compose (¥j -> if i==j then n else 0)          -- i軸方向、nマスの平行移動を定義
  a0 <- mapM (bind . shifti ( 2)) cell                                   --  2マス平行移動を全ての変数に適用、
  a1 <- mapM (bind . shifti ( 1)) cell                                   --  1マス平行移動を全ての変数に適用、
  a2 <- mapM (bind . shifti ( 0)) cell                                   --  0マス平行移動を全ての変数に適用、
  a3 <- mapM (bind . shifti (-1)) cell                                   -- -1マス平行移動を全ての変数に適用、
  intp <- sequence $ interpolateSingle order <$> a0 <*> a1 <*> a2 <*> a3 -- 各自由度ごとに補完関数を適用。
```



STM(1)

- Software Transactinal Memory
- 並行制御の仕組み
  - Cf. ロック、メッセージパッシング
- Haskellの標準ライブラリ！
- 正しい並行プログラムをゆるく書ける
  - ロックはいろいろな問題がある
    - デッドロック
    - ロックし忘れ
    - ロック粒度
    - etc…

STM(2)

```
deposit :: Account -> Int -> STM ()
deposit acc amount = withdraw acc (- amount)

type Account = TVar Int

withdraw :: Account -> Int -> STM ()
withdraw acc amount
  = do { bal <- readTVar acc
       ; writeTVar acc (bal - amount) }
transfer :: Account -> Account -> Int -> IO ()
-- Transfer 'amount' from account 'from' to account 'to'
transfer from to amount
  = atomically (do { deposit  to   amount
                   ; withdraw from amount })
```

STM(3)

- 軽量スレッド
- 高速IOマネージャ
- STM
  - この3つで、Haskellでの並行プログラミングはとても簡単！

告知！

- コミケで本出します！
  - タイトル：簡約！λカ娘！（参照透明な海を守る会）
  - 関数プログラミングに関する
    イカ娘アンソロジー本
    になる予定です！
  - Pandoc使ってます
  - C80 2日目 東地区 S-20 a
    YUHA様ブース
  - よろしくお願いします

画像は作成途中のものです→