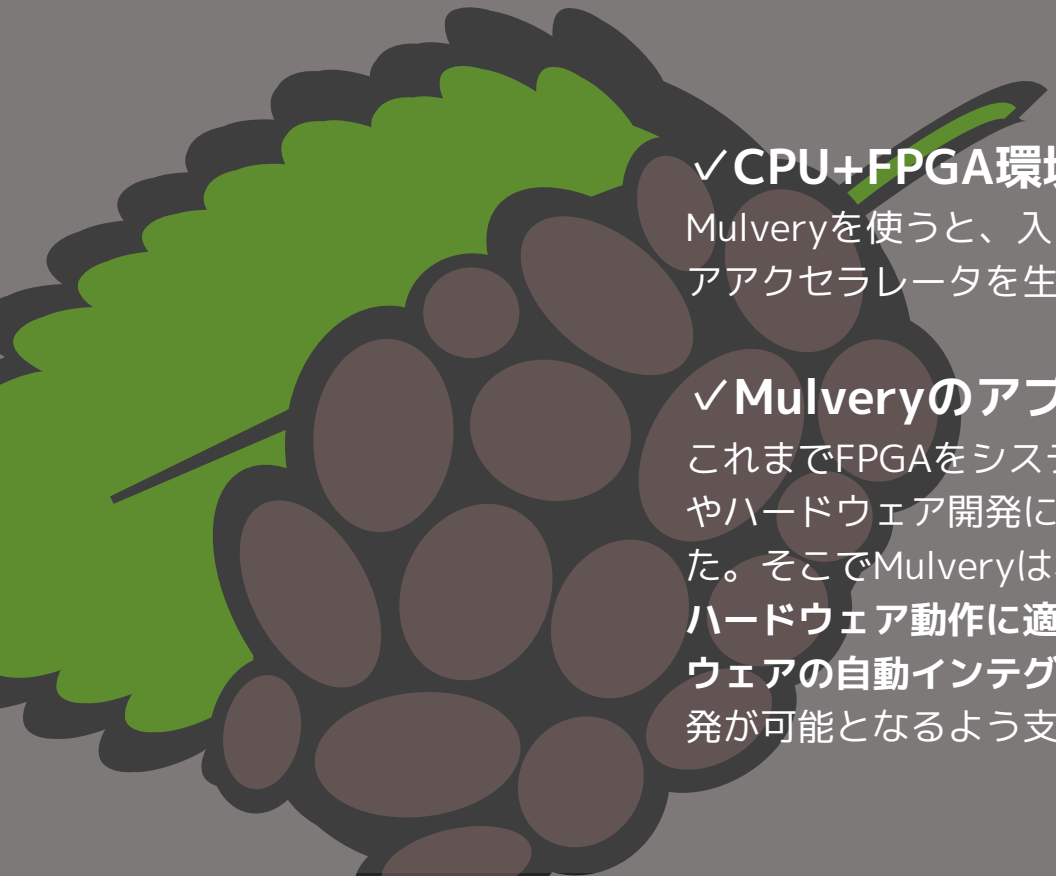
2017年度未踏IT人材発掘・育成事業

CPU＋FPGAプラットフォームのための Rubyベースの開発環境

-Rubyで書けるCPU＋FPGAコデザイン環境Mulvery-

照屋大地（東京農工大学工学府情報工学専攻）
呉屋寛裕（富士通クラウドテクノロジーズ株式会社）



✓CPU+FPGA環境でRubyが高速に動作するフレームワーク

Mulveryを使うと、入力したRubyプログラムに合わせて自動的にハードウェアアクセラレータを生成し、プログラムを高速に動作させます。

✓Mulveryのアプローチ

これまでFPGAをシステムに導入することは、コンピュータアーキテクチャやハードウェア開発に熟達しないエンジニアにとって非常に難しい課題でした。そこでMulveryは、RubyとReactiveXを使って書いたプログラムから**ハードウェア動作に適した部分を自動的に抽出**し、**ソフトウェアとハードウェアの自動インテグレーション**を行うことで簡単にCPU+FPGA環境での開発が可能となるよう支援します。

Mulvery

✓本プロジェクトの成果

本プロジェクトでは、Reactive Programmingのパラダイムを用いたプログラムから効率よくハードウェアを合成するための仕組みを開発しました。Mulveryでは、ReactiveXと同じAPIを用意しています。

ハードウェア開発 ＋ **ソフトウェア開発**

```
module event_count #(
  parameter BASE_ADDR = 8'h00
)(
  input CLK,
  input RST,
  input [31:0] event,
  input v_event,
  output [31:0] smem_data,
  output [7:0] smem_addr,
  output smem_write
);

  reg [31:0] smem_data;
  reg [7:0] smem_addr = BASE_ADDR;
  reg smem_write;

  assign smem_data = acc;

always @(posedge CLK) begin
  if (v_event) acc = acc + 1;
end
endmodule
```

```
int main(void){
  int ary = [1, 2, 3];
  int fd_mem = open(UIO_MEM, O_RDWR);
  int fd_io = open(UIO_IO, O_RDWR);
  volatile unsigned int *mem = (volatile
unsigned int *)mmap(NULL, UMEM_SIZE,
PROT_READ|PROT_WRITE, MAP_SHARED,
fd_mem, 0);
  volatile unsigned int *io = (volatile unsigned
int *)mmap(NULL, UMEM_SIZE,
PROT_READ|PROT_WRITE, MAP_SHARED,
fd_io, 0);

  for (int i = 0; i < 3; i++){
    *io = ary[i];
    cache_clean((char *)mem, 32);
    msync((void *)mem, UMEM_SIZE,
MS_SYNC);
    volatile int *c = (volatile int *)&mem[size];
    printf("%d¥n", *c);
  }
}
```

Mulvery

```
ary = [1, 2, 3]
events = Mv::Observable.from_array(ary)
count = events
          .map(){ |e| 1 }
          .scan(){ |c, x| c + x }
          .subscribe(){ |c| p c }
```

✓IoTアプリケーション開発への応用

Mulveryは、センサやアクチュエータのような外部デバイスを大量に扱うアプリケーションの開発を得意とします。Mulveryを使えば、Rubyで記述されたマイコン上のWebサーバから1024個のフルカラーLEDを同時に制御するようなアプリケーションも実現可能です。