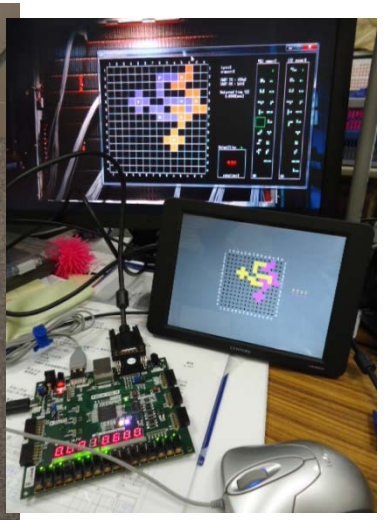
同期FIFOのFPGAベンダ非依存記述と高速化設計

立命館大学
理工学部 電子情報工学科
大学院 理工学研究科 電子システム専攻
泉　知論

by t-izumi@se.ritsumei.ac.jp / 無断複製・転載を禁ず

FPGAベンダと設計環境

- 各社競い合ってよりよい設計環境、ライブラリを提供。
- 一方で、ユーザはベンダ依存の設計を強いられる。



AでもXでもどちらでも使いまわせる設計をしたい

（少なくとも、情報処理のコア部分、汎用部品、基本部品は）

脱ベンダ依存計画

XILINX	非依存	ALTERA
	高位合成ツール	
VIVADO HLS	サードパーティーのツール Impulse	C2H
	ソフトコアプロセッサ	
MICRO BLAZE	公的検定試験用プロセッサ COMET II	NIOS II
	基本ライブラリ	
CORE GENERATOR	ベンダ非依存記述	MEGA WIZARD

by t-izumi@se.ritsumei.ac.jp / 無断複製･転載を禁ず

ほぼ同じ v.s. 全く同じ

- FIFO は基本部品
- Altera も Xilinx も IP を提供
- 機能、入出力はほぼ同じ
- 仕様の確認、互換性の確認
- 必要なら極性、論理、タイミングの微修正
- 書き換えても1行～数行
- ライブラリ生成ツール、パラメタ設定、生成、登録
- 誰が互換性を保証するか

v.s.

- ファイルをコピーするだけ
- 何も考えない、何も作業しない、何も心配しない

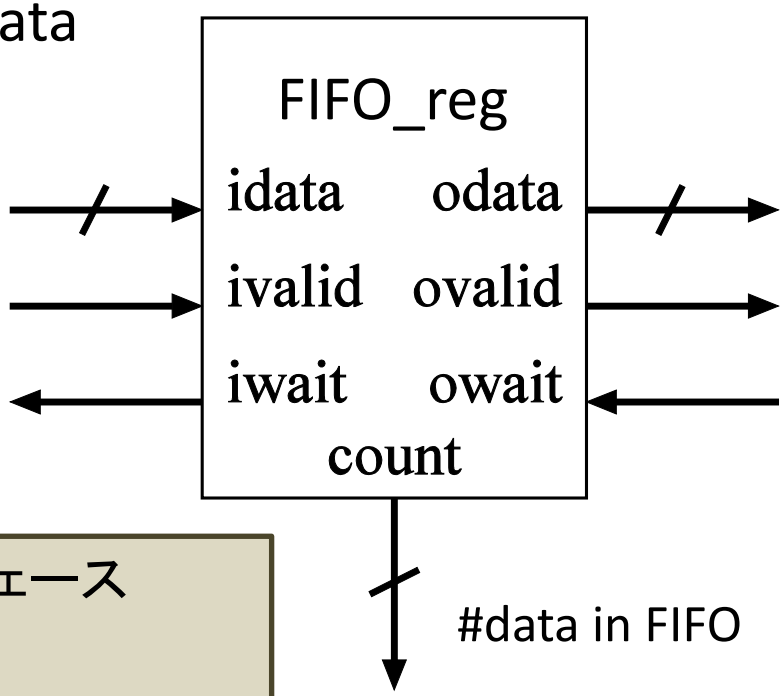
FIFOとデバイスへの依存性

- ストリーミング処理、データ駆動処理に必須
- 小さなFIFOはレジスタで実装可能→デバイス非依存コードが容易にできる
- 大きなFIFOはメモリで実装→若干複雑な動作→通常はIP利用→デバイス依存
- 単純なRAMモデルは比較的移植性が高い
- デバイス非依存な RTL コードを開発

  ※ここでは Verilog HDL を使用

(1) register 高速 FIFO

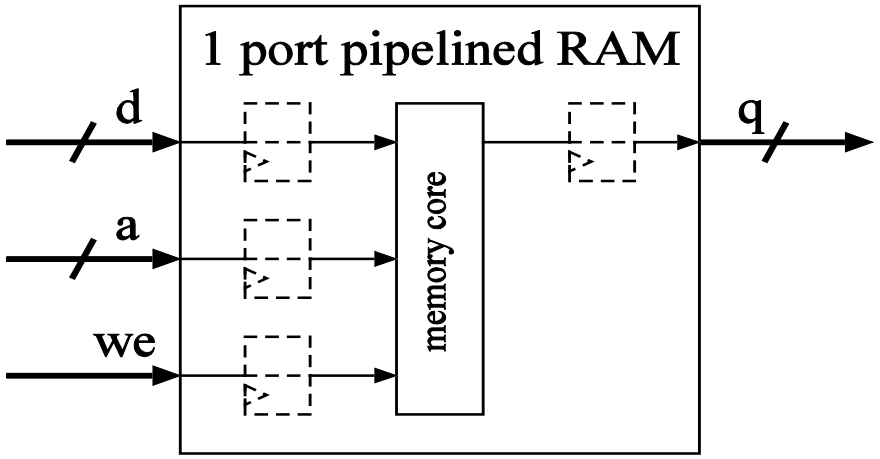
- レジスタで実装、小容量で高速
  - スループット : 1 cycle per 1 data
  - レイテンシ : 1 cycle
- どこでも通る記述



補足:いずみ研的ストリームインターフェース
- データ線＋制御線
- 送信側の valid, 受信側の wait
- 転送成立 iff (valid==1)&&(wait==0)

基本部品 : 汎用 block RAM

- 1 port pipelined read/write
- 0, 1, or 2 cycles delay
- 「よくある」仕様
- 平易な記述で BRAM に割り当てられる



description for 2cycles delay

```
always @(posedge clk) begin
  a_reg<=a;
  d_reg<=d;
  we_reg<=we;
  if (we_reg) begin
    mem[a_reg]<=d_reg;
    q_reg<=mem[a_reg];
  end else begin
    q_reg<=mem[a_reg];
  end

assign q = q_reg;
```

汎用メモリ記述と使用リソース

	no delay		1cycle delay		2cycles delay	
Xilinx XC5VLX30 ISE 14.4	Logic (Slice)	Memory (36k BRAM)	Logic (Slice)	Memory (36k BRAM)	Logic (Slice)	Memory (36k BRAM)
	172	0	10	1	10	1
Altera EP4CE115 Quartus II 12.0	Logic (LE)	Memory (bit)	Logic (LE)	Memory (bit)	Logic (LE)	Memory (bit)
	46043	0	99	32768	65	32768

ロジックリソースで
メモリを構成

```
always @(posedge clk)
  if (we) mem[a]<=d;

assign q = mem[a];
```

```
always @(posedge clk) begin
  a_reg<=a;
  d_reg<=d;
  we_reg<=we;
  if (we_reg)
    mem[a_reg]<=d_reg;

assign q = mem[a_reg];
```

```
always @(posedge clk) begi
  a_reg<=a;
  d_reg<=
  we_reg<
  if (we_r
    mem[a
    q_reg<=mem[a_reg];
  end else begin
    q_reg<=mem[a_reg];
  end

assign q = q_reg;
```

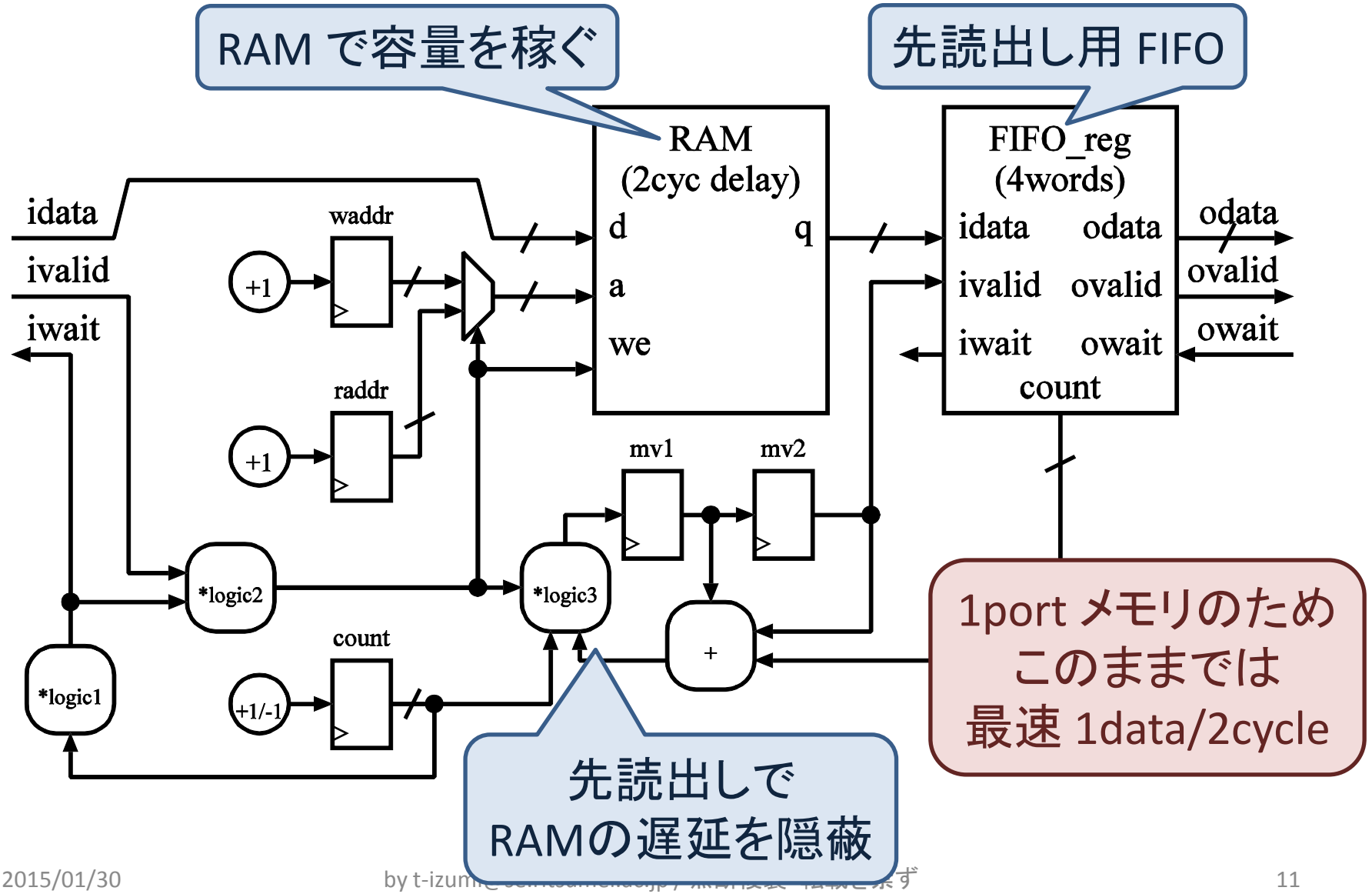
ブロックRAMで
メモリを構成

8bit × 4096 words
メモリ＋α を合成した結果

(2) block RAM低速FIFO

- 2 cycles delay の pipelined single port memory を使用
- メモリ読出しのレイテンシを先読み出しバッファで隠蔽
- 相対的に大容量で低速
  - スループット : 2 cycles per 1 data
  - レイテンシ : 4 cycles
- ベンダ非依存記述

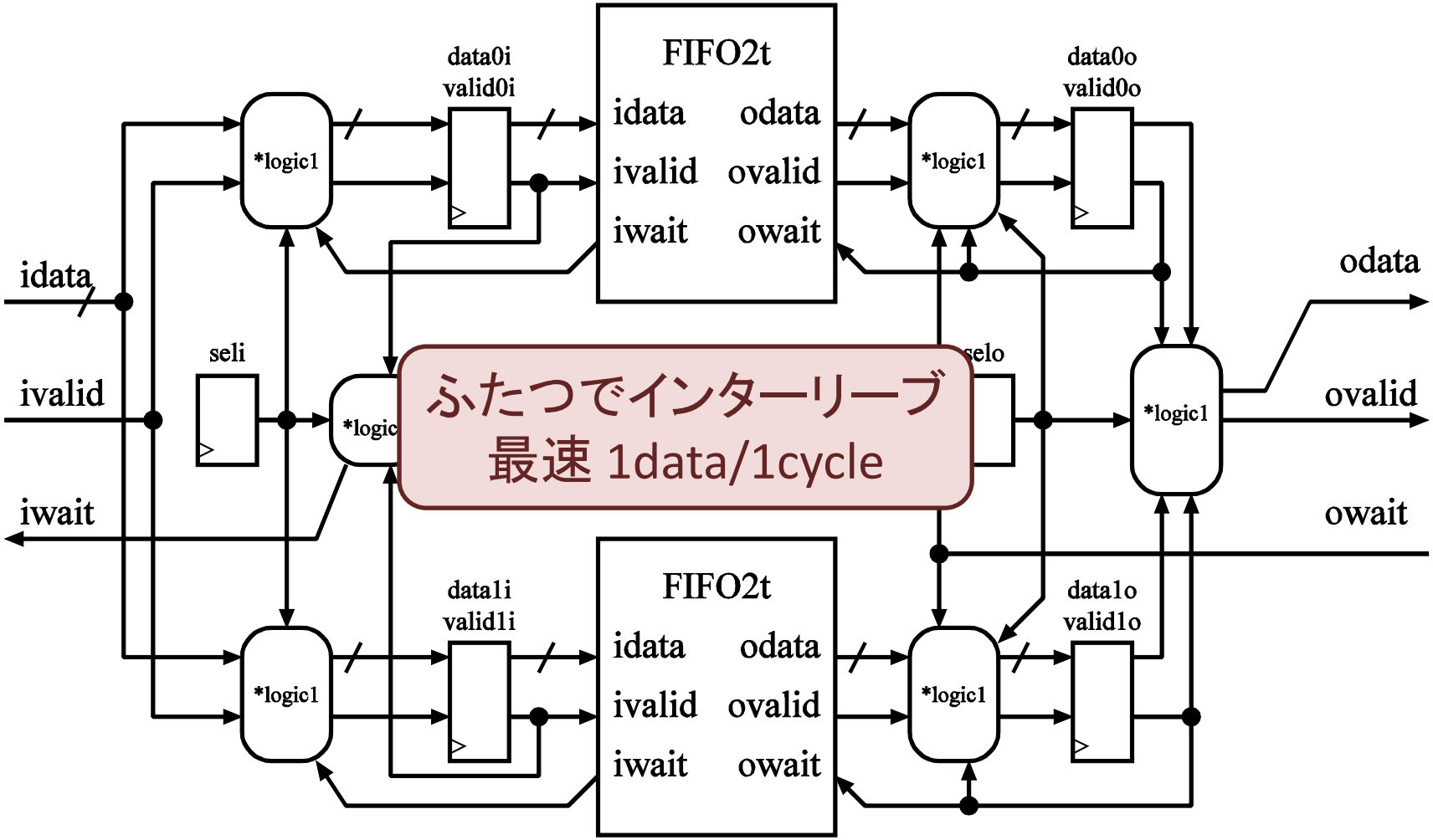
2t FIFO


RAM で容量を稼ぐ
先読出し用 FIFO
RAM
(2cyc delay)
d
q
a
we
FIFO_reg
(4words)
idata
odata
ivalid
ovalid
iwait
owait
count
idata
ivalid
iwait
waddr
+1
raddr
+1
mv1
mv2
*logic1
*logic2
*logic3
count
+1/-1
+
1port メモリのため
このままでは
最速 1data/2cycle
先読出しで
RAMの遅延を隠蔽

(3) block RAM高速FIFO

- 2t FIFO ふたつでインターリーブする
- 相対的に大容量で高速
  - スループット : 1 cycle per 1 data
  - レイテンシ : 6 cycles
- ベンダ非依存記述

インターリーブ FIFO



合成＆実装

XILINX XC7A100T-1CSG324C ISE 14.4			ALTERA EP4CE115F29C7 Quartus II 12.0	
IP Generator	非依存記述	記述	非依存記述	Mega Core
43 slices*	75 slices*	ロジック量	272 LEs*	78 LEs*
2 RAMB36s*	2 RAMB36s*	メモリ量	65536 bits*	65536 bits*
235 MHz	126 MHz	最大クロック	62 MHz	289 MHz

- 8bit×8192words のFIFOを実装
- リソース量*は全体の合成結果から周辺のみの合成結果を引いた値
- IP Generator, Mega Core の機能／インターフェースをほぼ同じにした
  (common clock, read timing, data count, sync reset)
  （throughput, latency は未確認）

まとめ

- 同期FIFOのFPGAベンダ非依存なライブラリ化設計
  - register 1cycle/data FIFO
  - bRAM 2cycle/data FIFO
  - bRAM 1cycle/data FIFO
- 十分に小さく、速い
- チューンの余地あり
- 2port RAM の検討
- Verilog HDL 記述、ビット幅、容量パラメタ設定可能
- 公開しています

  www.ritsumei.ac.jp/se/re/izumilab/dist/fifo4any.htm