複数の高位合成ツールに対応したシステムレベル設計ツールの拡張

名古屋大学

稲石 日奈子　山本 椋太　本田 晋也

Graduate School of Information Science, Nagoya Univ.
Parallel & Distributed Systems Lab.

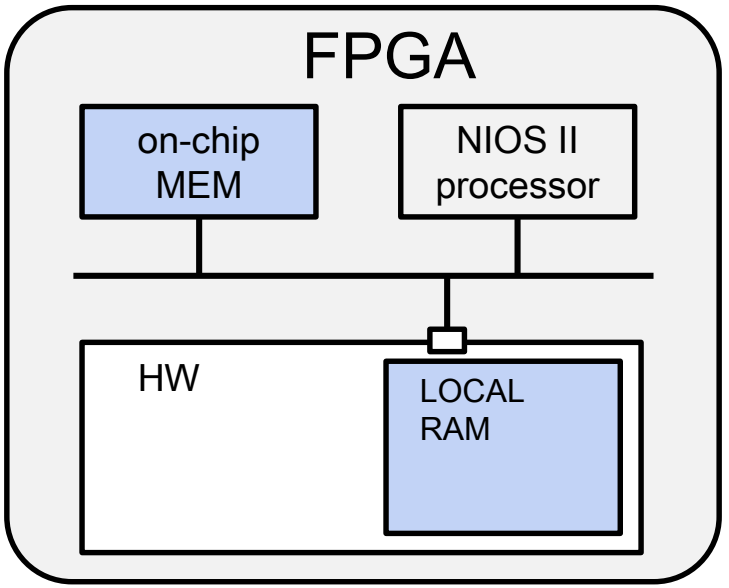
研究背景

- 我々の研究グループはFPGAによる効率の良い深層学習の推論器を検討している
  - C ソースコードを用いた高位合成（HLS）によって設計
- FPGA の主なメーカーに Intel（Altera）と XILINX がある
  - これまで我々のツールは Intel の FPGA をサポートしてきた
    - Intel の提供してる周辺ツールの完成度が高かったため
    - 近年 Intel はサーバ向けのハイエンド FPGA に注力しており，組込み向けの FPGA のリリースが滞っている
- そこで XILINX の FPGA を使用する
  - XILINX は，推論や自動運転向けの FPGA やツールに注力している
  - Vivado HLS など無償で利用できる XILINX の周辺ツールも充実してきた

Graduate School of Information Science, Nagoya Univ.
Parallel & Distributed Systems Lab.
PDSL

研究環境

- HLS ツール
  - CyberWorkBench（CWB）：ver.6.1.45
  - Vivado HLS：2018.2
- システムレベル設計ツール
  - SystemBuilder
- FPGA Board
  - DE2-115
    - Cyclone IV EP4CE115

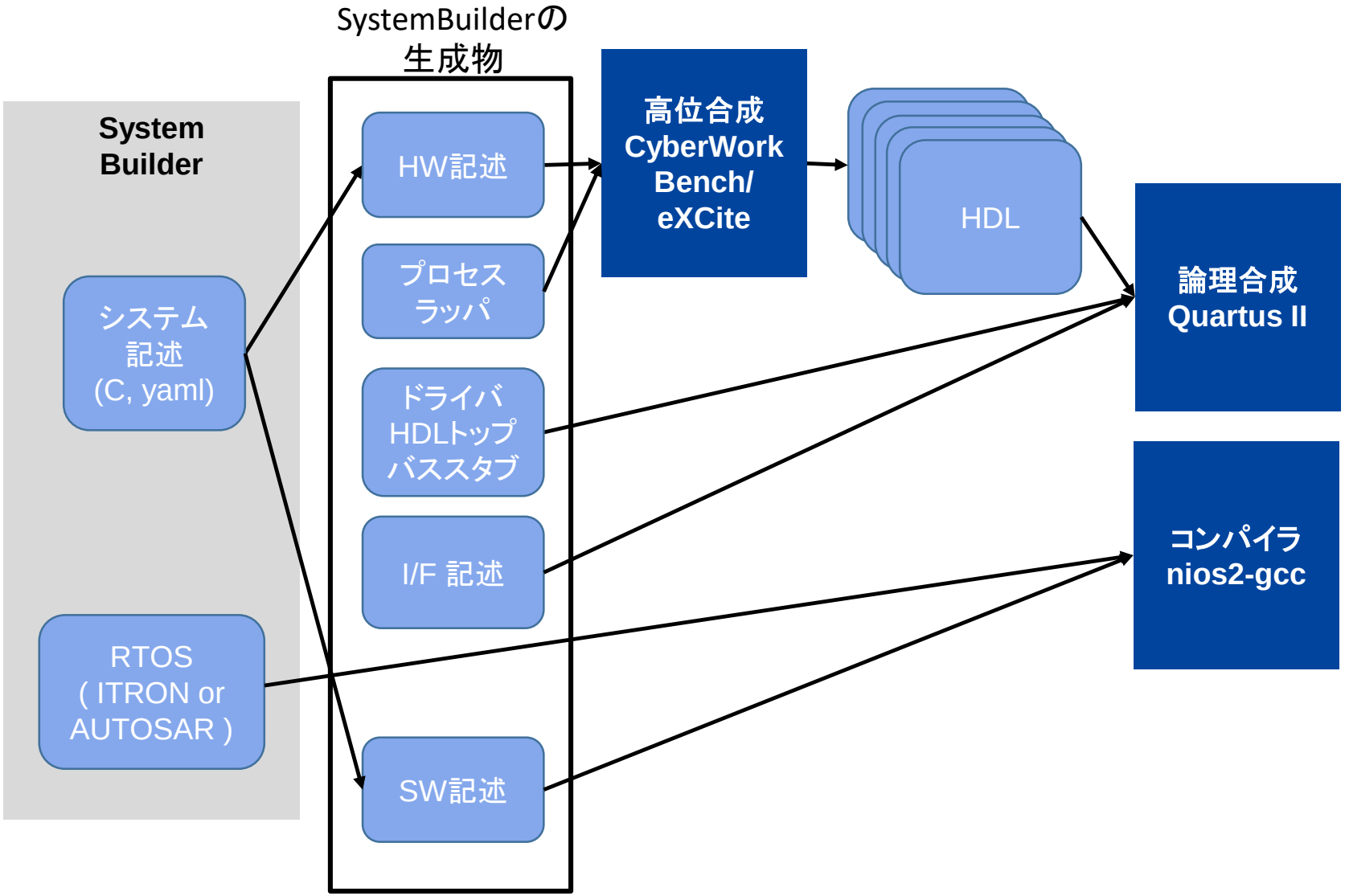


SystemBuilder [1]

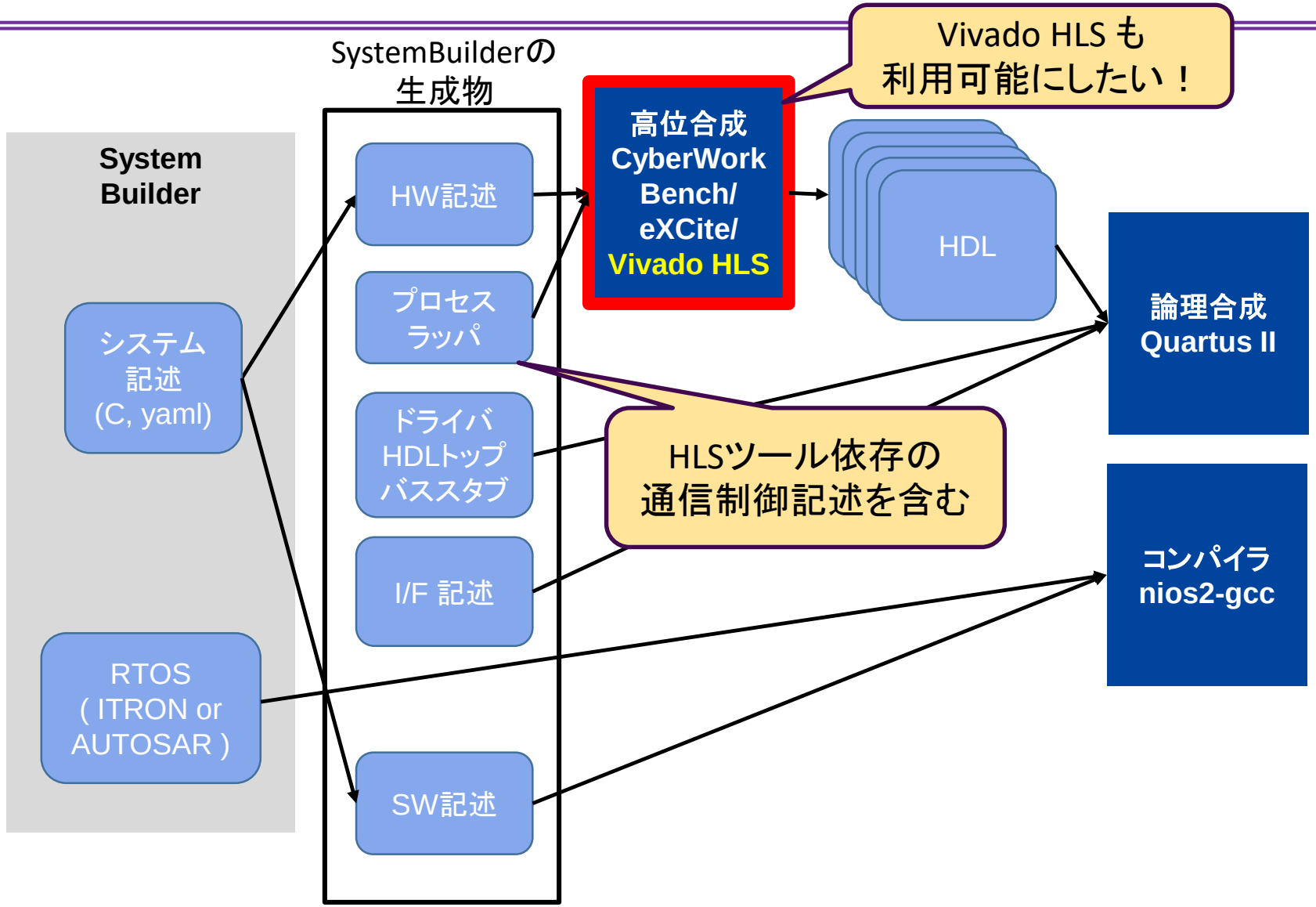
- システムレベル設計ツール
- 通信プリミティブや SW/HW 分割方法を記述したファイルを作成する
- 内部で HLS ツールを呼び出している
  - 現在，eXCite と CWB（CyberWorkBench）を使用している
    - eXCite，CWB は XILINX の FPGA にも対応しているが使用料が非常に高価である
    - そのため SystemBuilder を共同研究先の企業で使用する際の障害になっていた
  - そこで XILINX の FPGA 向けに XILINX が無償で提供している Vivado HLS の使用を検討している

[1]本田晋也, 冨山宏之, 高田広章. システムレベル設計環境: SystemBuilder. 電子情報通信学会論文誌 D, Vol. 88, No. 2, pp. 163–174, 2005.

SystemBuilder を用いた開発フロー



SystemBuilder を用いた開発フロー

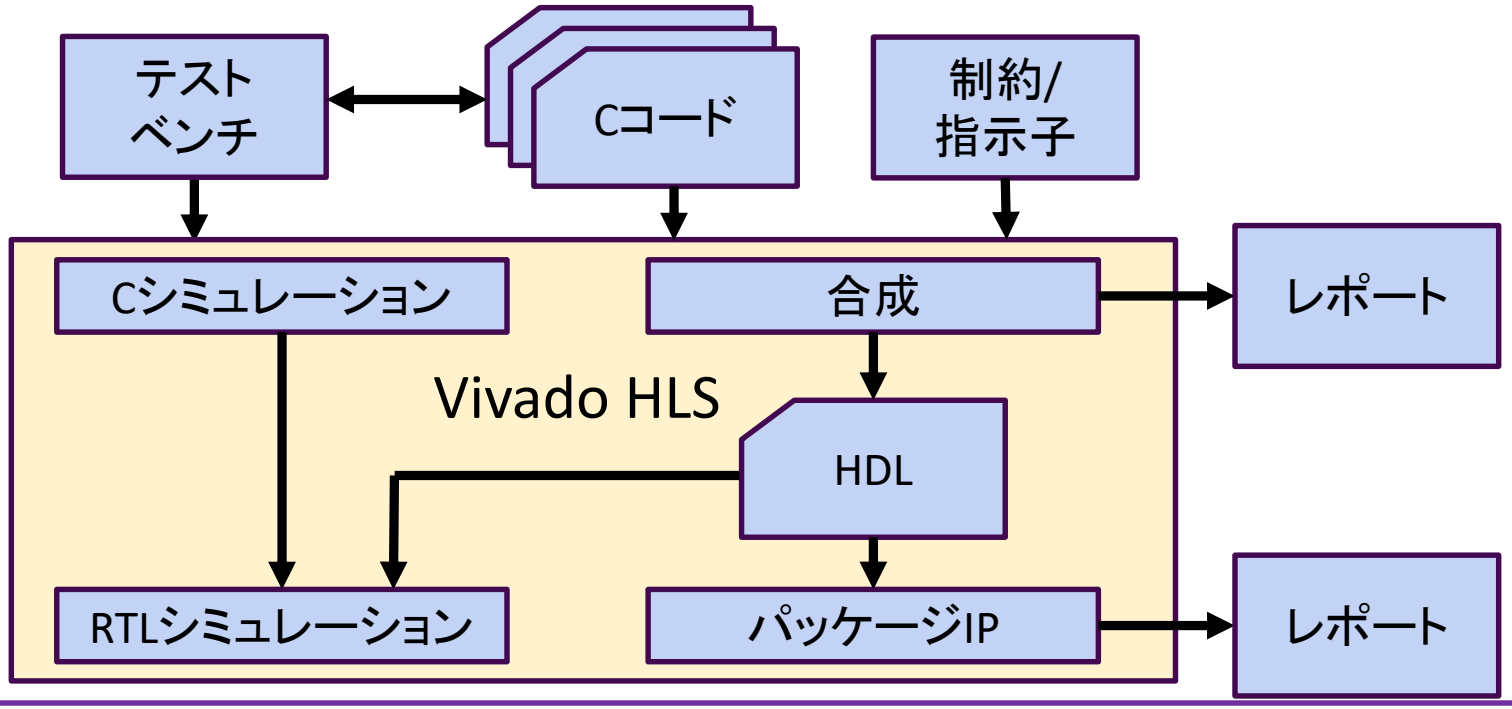


研究目的

- Vivado HLS をSystemBuilder で使用できるようにする
  - SystemBuilder で使用しているプロセス間通信機構の Vivado HLS 対応
- 現在使用しているHLSツール（eXCite，CWB）と同等程度の実装を目指す

Vivado HLS

- XILINX が提供している HLS ツール
  - C 関数を FPGA で実装可能な IP ブロックに合成する
    - 入力：C 関数，制約，指示子，テストベンチ
    - 出力：HDL 記述，各種レポート



Vivado HLS

- 指示子などを用いて目的に合わせた
  複雑な実装が可能である
  - ループ，配列の最適化やパイプライン処理など
  - 実装の最適化のために HW の知識が必要

- 開発者が指定した関数の引数，戻り値，
  グローバル変数に対してポートが生成される
  - 指示子によって生成されるポートの種類を
    指定できる

Vivado HLS のプロセスラッパの実装

- SystemBuilder の CWB 向けプロセスラッパの記述を参考に実装を行う
  - 現在よく使用されているプロセス間通信機構は，BC，NBC，MEM，PFBC である

- Vivado HLS には独自の記述方法がある
  - グローバル変数で宣言してポートを生成する場合，宣言に volatile が必要である
  - 指示子の適用範囲を指定するブロック領域がある

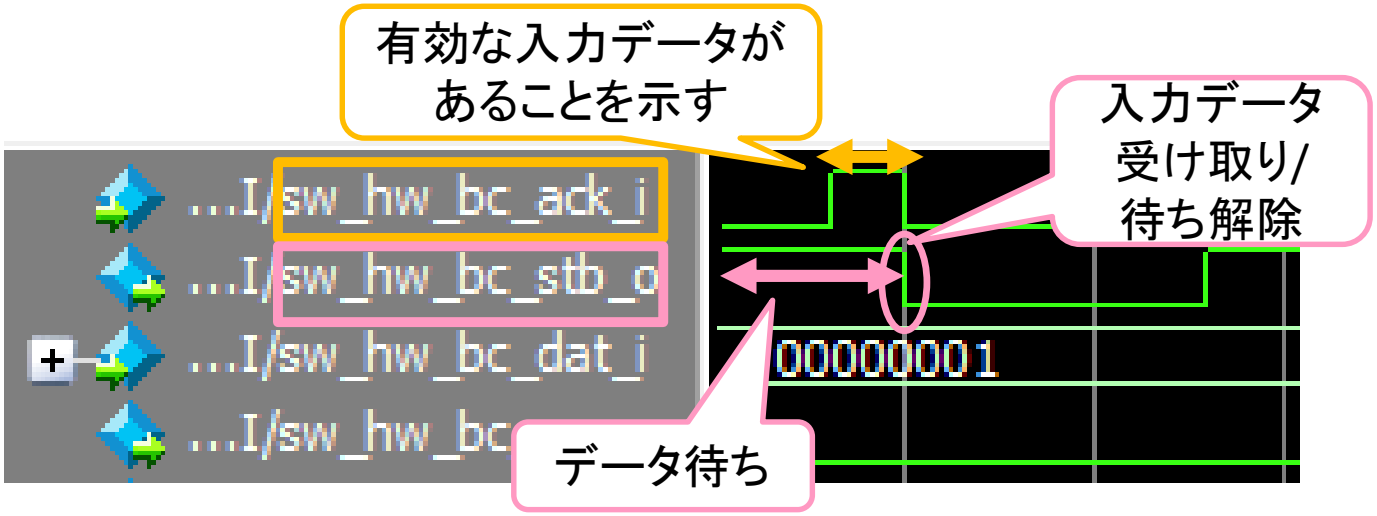
今回実装するプロセスラッパ

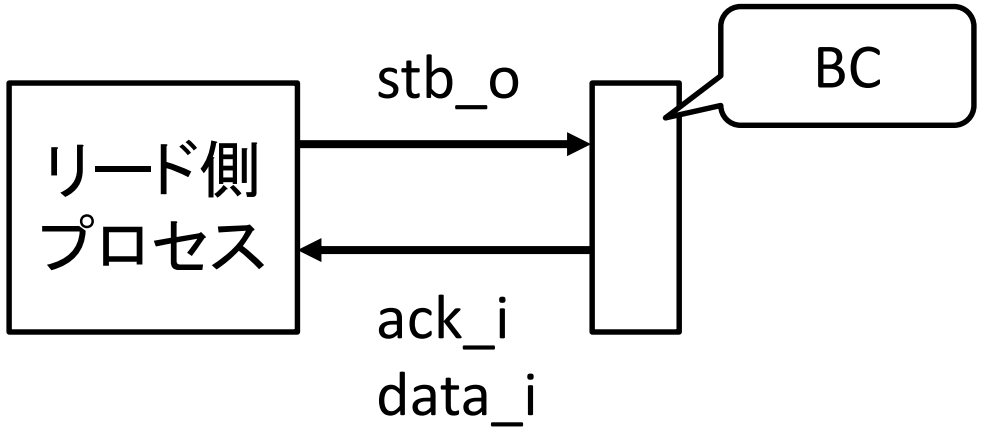
- BC（Blocking Channel）
  - ハードウェアとしては FIFO に相当する
  - リード：バッファが空のときにリード側プロセスを待ち状態にする
  - ライト：バッファがいっぱいのときにライト側プロセスを待ち状態にする
- NBC（Non-Blocking Channel）
  - ソフトウェアとしては共有変数，ハードウェアとしてはレジスタに相当する
  - プロセスは待ち状態にならない
- MEM
  - 内部または外部メモリに対してアクセスする（R/W）

Graduate School of Information Science, Nagoya Univ.
Parallel & Distributed Systems Lab.
PDSL

BC の動き（リード）

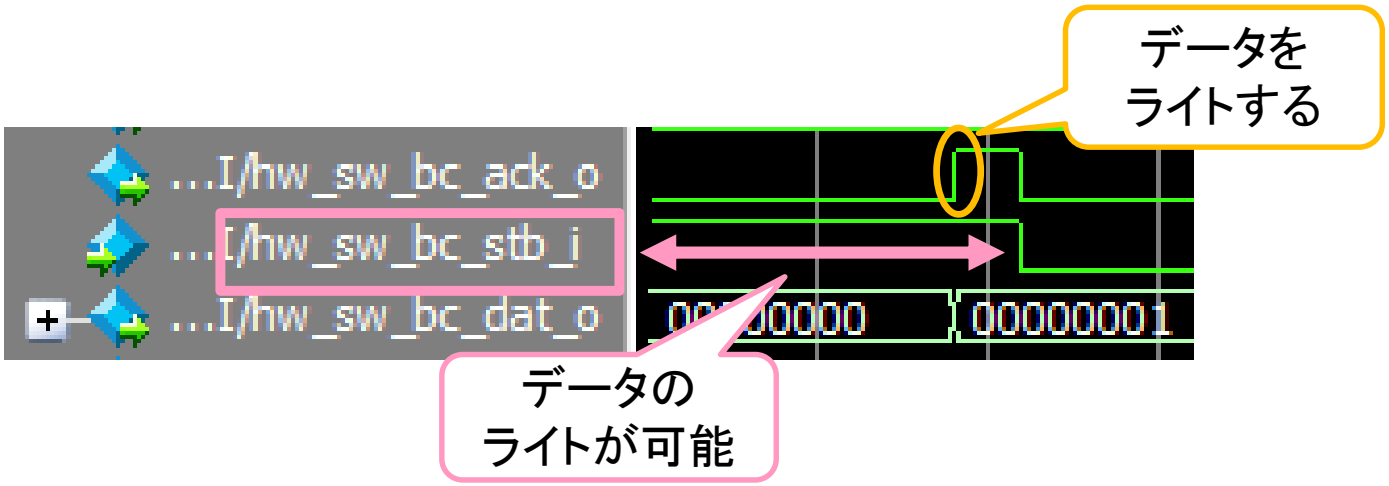
- CWB で生成されたHDLのシミュレーション結果

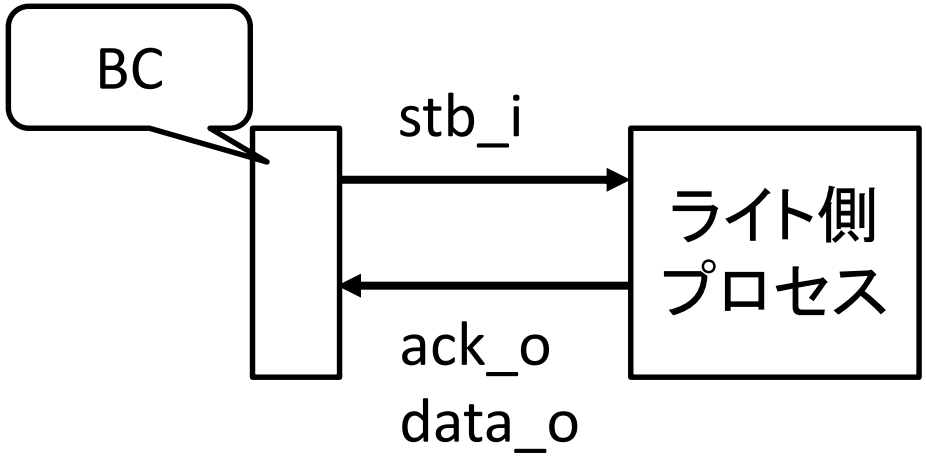




BC の動き（ライト）

- CWB で生成されたHDLのシミュレーション結果





Vivado HLS 向け BC 実装の流れ

- ポートの生成方法
  - 宣言方法
  - ポートに対する適切な指示子の選択
- クロックの挿入方法
- 実行順序の固定

Vivado HLS での BC 実装：ポートの生成

```
volatile uint1 *sw_hw_bc_ack_i;
volatile uint1 *sw_hw_bc_stb_o = 0;
volatile uint32 *sw_hw_bc_dat_i;
volatile uint1 *sw_hw_bc_we_o = 0;

void SW_HW_BC_READ(uint32 *data_ptr){
    BC_READ:{
#pragma HLS protocol fixed
        vivado_bcchan32_read(sw_hw_bc, data_ptr);
    }}

void HW_RCVMOD_E(void){
#pragma HLS inline region recursive

#pragma HLS INTERFACE ap_none port=sw_hw_bc_ack_i
#pragma HLS INTERFACE ap_none port=sw_hw_bc_stb_o
#pragma HLS INTERFACE ap_none port=sw_hw_bc_dat_i
#pragma HLS INTERFACE ap_none port=sw_hw_bc_we_o
    hw_rcvmod();
}
```

グローバル変数
（volatile必須）

使用しない
ポートの生成を
抑制する指示子

Vivado HLS での BC 実装：順序の固定

```
volatile uint1 *sw_hw_bc_ack_i;
volatile uint1 *sw_hw_bc_stb_o = 0;
volatile uint32 *sw_hw_bc_dat_i;
volatile uint1 *sw_hw_bc_we_o = 0;

void SW_HW_BC_READ(uint32 *data_ptr){
    BC_READ:{
#pragma HLS protocol fixed
        vivado_bcchan32_read(sw_hw_bc, data_ptr);
    }}

void HW_RCVMOD_E(void){
#pragma HLS inline region recursive

#pragma HLS INTERFACE ap_none port=sw_hw_bc_ack_i
#pragma HLS INTERFACE ap_none port=sw_hw_bc_stb_o
#pragma HLS INTERFACE ap_none port=sw_hw_bc_dat_i
#pragma HLS INTERFACE ap_none port=sw_hw_bc_we_o
    hw_rcvmod();
}
```

指示子の適用範囲を指定するブロック記述

ブロック外の処理と並列に実行することを抑制する記述

モジュール内の関数をインライン展開する指示子

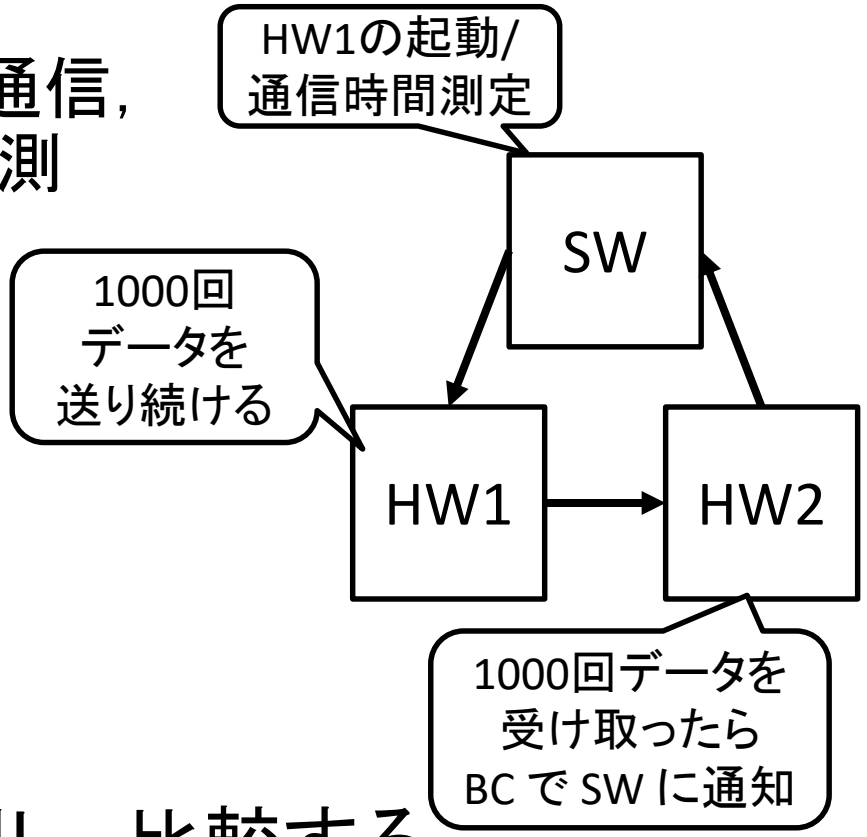
その他のプロセスラッパの実装

- NBC，MEM ともに BC のプロセスラッパに変更を加える形で実装した

- NBC
  - プロセスの待ちを行うポートの削除
    - リードなら stb_o，ack_i，ライトなら stb_i，ack_o
- MEM
  - プロセスの待ちを行うポートの削除
  - アドレスを指定するポートの追加

実装の評価：通信時間の計測

- 実装した各プロセスラッパの通信時間を計測する
  - BC を用いたプロセス間通信，1000回の合計時間を計測
  - FIFO には十分な深さが用意されており，ライト時に待ちが発生しないものとする



- 現在 SystemBuilder が対応している CWB でも同様に通信時間を計測し，比較する

計測結果

- 実機を用いて計測した通信時間の平均値

	CWB	Vivado HLS
通信時間[us]	6169.2	6175.6

  - 1000回の送受信で約6[us]の差はあるが，同程度の通信時間の実装となった

まとめと今後の課題

- Vivado HLS 向けプロセスラッパの実装を行った
  - BC，NBC，MEM の実装を行った
- 他のプロセスラッパでも通信時間の比較を行う
- 複雑なソースコードに対しても実装したプロセスラッパを適用してみる