汎用FPGAモジュールを用いた ATLASミューオントリガー検出器用 読み出しシステムのアップグレード研究

東京大学素粒子物理国際研究センター

二ノ宮　陽一

神谷隆之，坂本宏，結束晃平，織田勧，
佐々木修A，池野正弘AB，内田智久AB，
蔵重久弥C,，菅谷頼仁C，福永力E，
他ATLAS日本TGCグループ

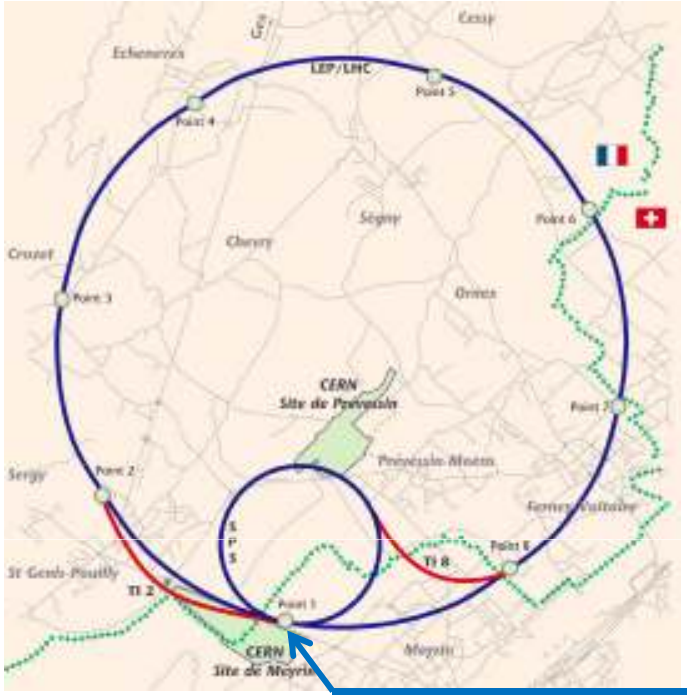
KEK素核研A，Open source consortium(Open-It)B，
神戸大理C，阪大理D，首都大E

概要

1. LHCとATLAS検出器について
2. TGC(Thin Gap Chamber)
   - L1トリガーについて
   - データラインについて
3. アップグレード計画
4. TGC読み出しシステムのアップグレード研究
   - HDLデザインの検証
   - MicroBlazeの検証
5. まとめ

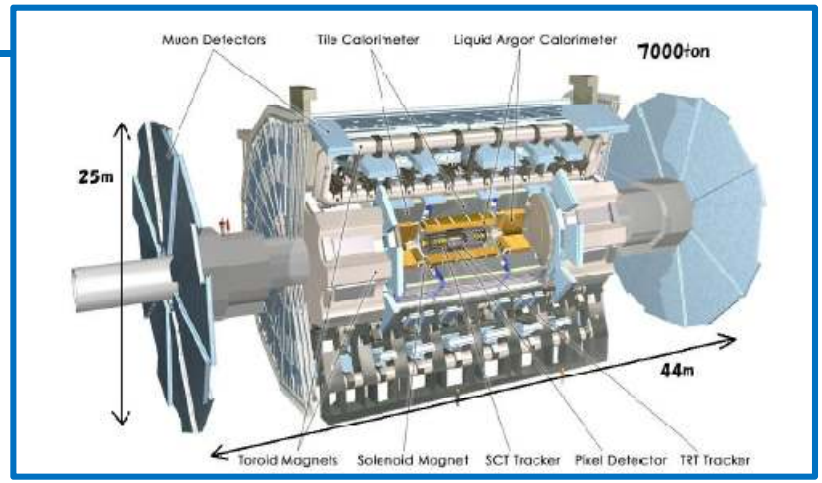
この開発はOSCプロジェクトの一つです
FPGAデザイン等は公開予定です

LHC加速器とATLAS検出器



ATLAS検出器

Higgs粒子や、標準模型を超える
物理現象の探索を行う汎用検出器
- -Tracker
- -Calorimeter
- -Muon Spectrometer
  - ・MDT　・CSC
  - ・TGC　・RPC



LHC加速器

・主リング長	26.66km
・ビームエネルギー	7TeV
・デザインルミノシティ	$10^{34}cm^{-2}s^{-1}$
・衝突頻度	40.08MHz
・バンチ数	2808個

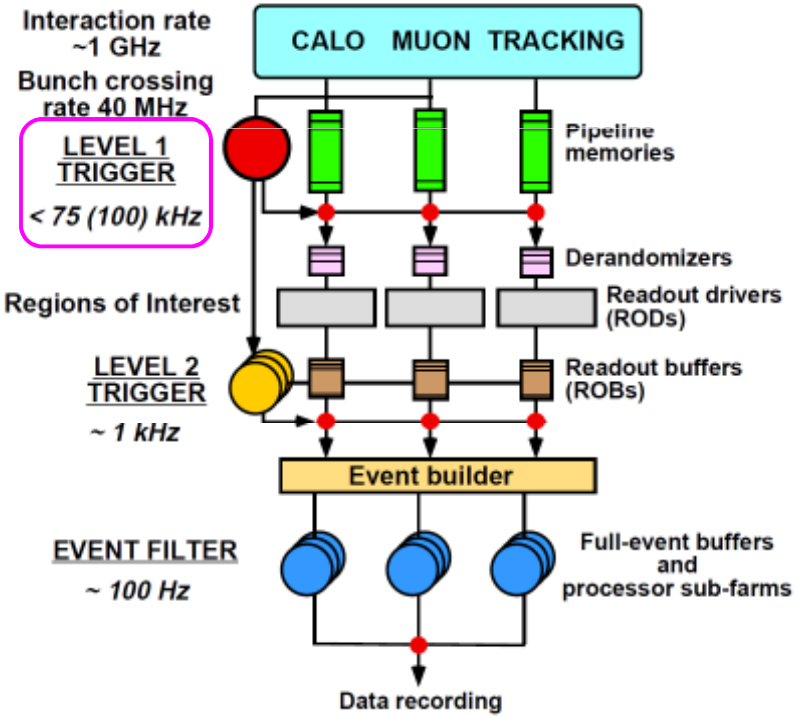
TGC(Thin Gap Chamber)

TGC(Thin Gap Chamber)

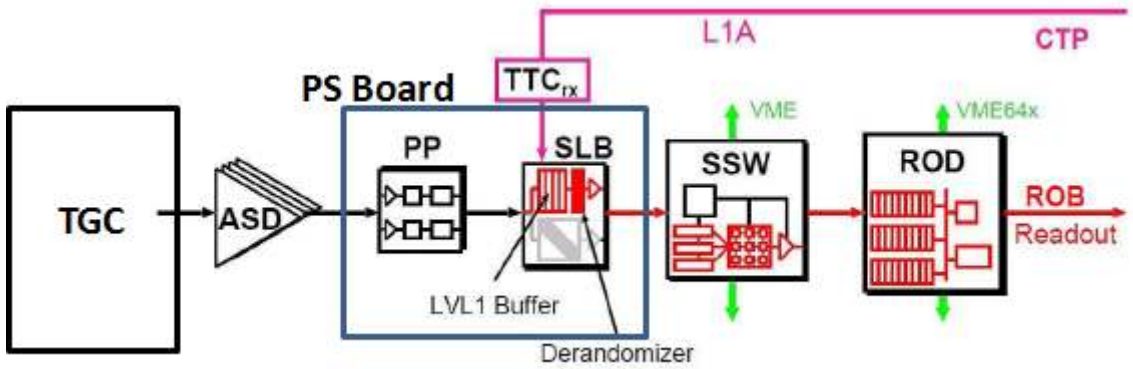
前後方部のトリガーを担う、MWPCの一種。wire間隔よりもwire-strip間の方が狭くなっている。25ns以下の時間分解能。wireがR方向、stripがφ方向の情報を出力。

Level 1 トリガー

TGC、RPC、カロリメータの情報を基に2.5μs以内での高速なトリガー判定。L1A信号を発行。トリガーレート75kHz。

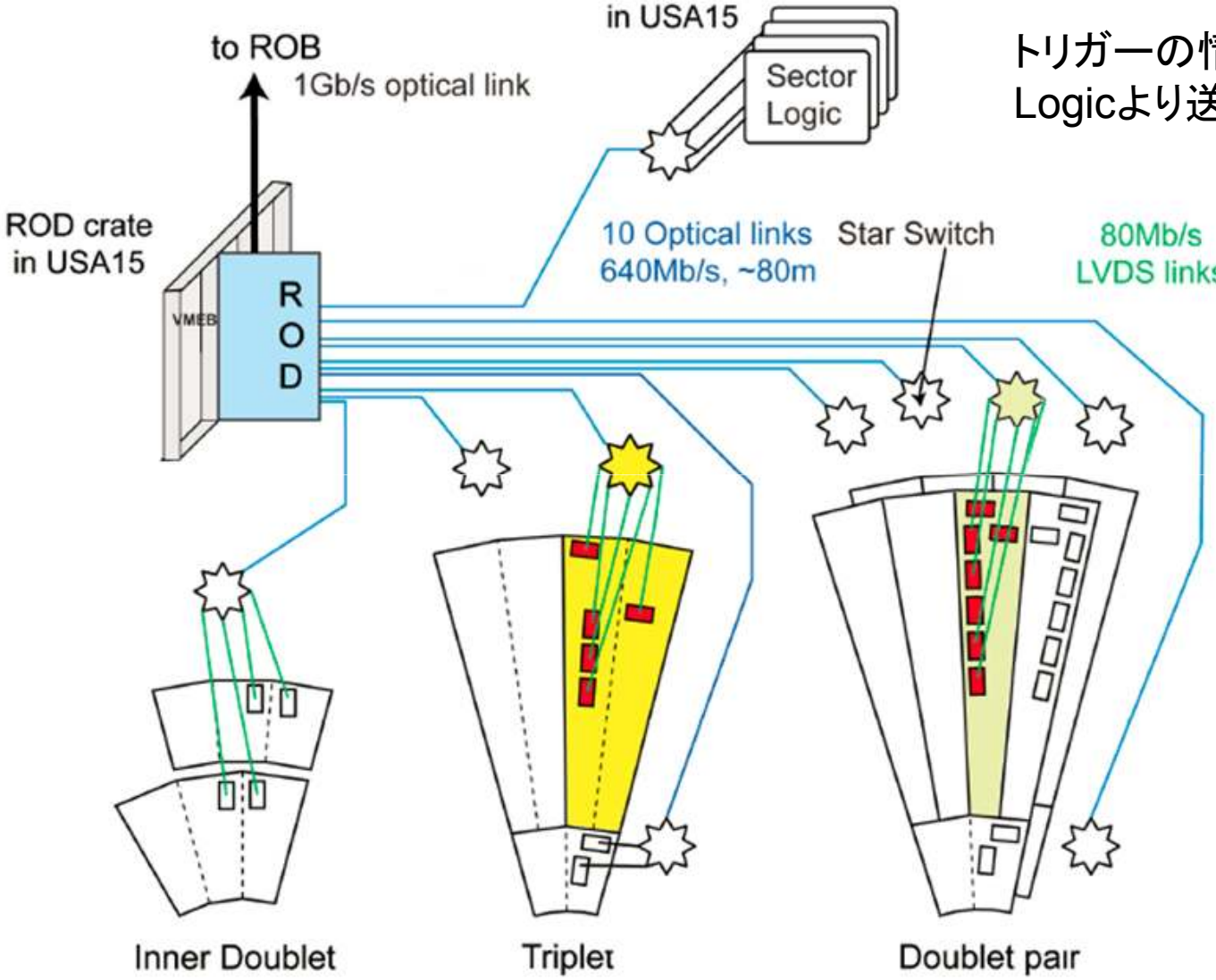


TGCのリードアウトライン



1. TGCからの信号をASDで増幅、整形、デジタル処理。
2. PSボード上のPP ASICにて遅延を調整。レベル1バッファーに保存。
3. L1Aを受け取った前後3バンチ分のデータがSSWへと送られる。
4. SSWでデータを圧縮。
5. RODでは、SSWのデータを受け取り、ATLAS共通のフォーマットに整形し直してからROS(Read Out System)のバッファへとデータを送信する。1つのRODで10個のSSWの入力を受ける。

TGCのリードアウトライン

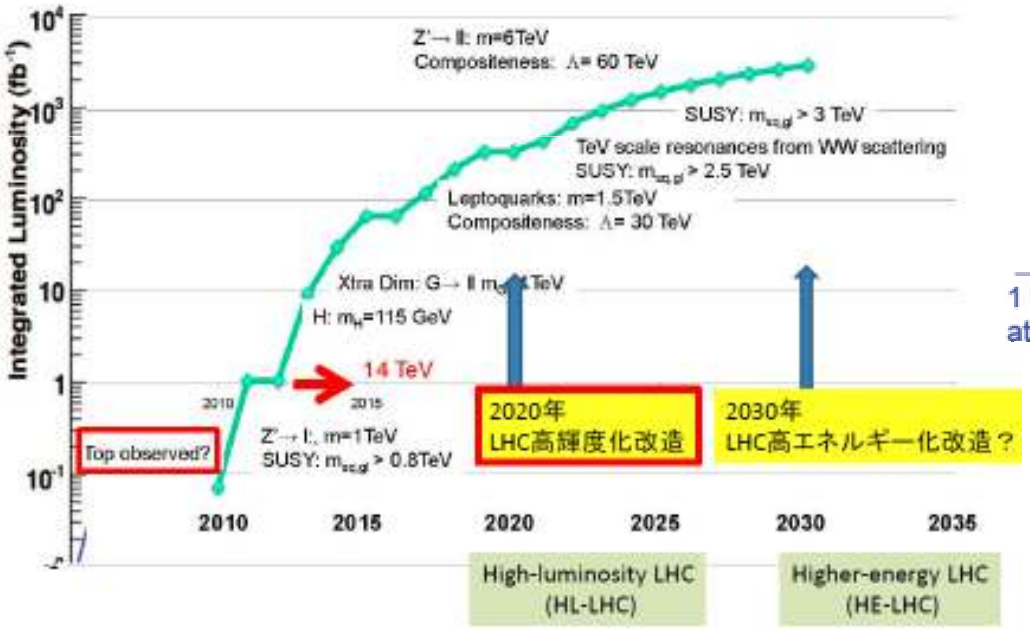


トリガーの情報がSector Logicより送られる。

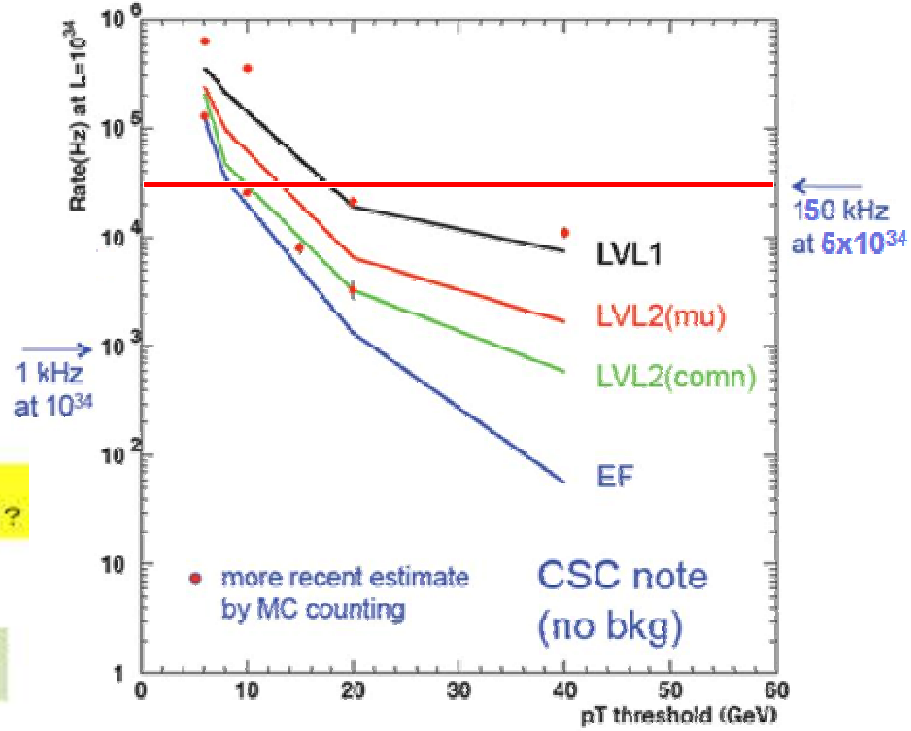
1つのRODでTGC12分の1を担当する。
従って、現在24枚のRODがインストールされている。

LHC加速器のアップグレード計画

LHCのルミノシティが$10^{34}cm^{-2}s^{-1}$から5 ×$10^{34}cm^{-2}s^{-1}$に。
→ Higgs粒子の精密測定や質量の大きい新粒子の発見等を目指す
レベル1トリガーレートは、現状の75kHzから150kHzへ。
→処理速度の向上が必要



2010年日本物理学会秋季大会 徳宿克夫



2010年日本物理学会秋季大会 川本辰男

RODのアップグレード

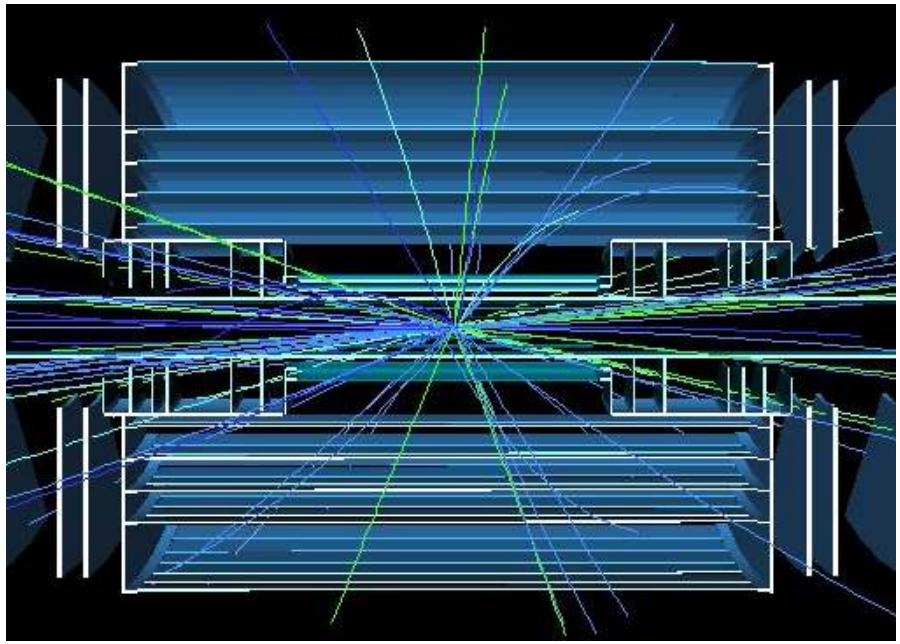
データ量の考察

HeaderとTrailerだけで、1152Byte存在。ルミノシティが1×10^{34} で、1セクターで10トラックあると、ヒットデータは、80Byte。計1232Byte。アップグレード後単純にトラックの数が5倍になったとすると、ヒットデータは400Byte。計1552Byte。

⇒ アップグレード後のデータサイズは、1.25倍

処理時間の考察

待ち行列理論で用いられる平均利用率ρを用いて、処理時間を決める。
λ = 平均到着率
(単位時間当たりの平均到着データサイズ)
μ = 平均サービス率
(単位時間当たりの平均処理データサイズ)
を用いて、ρは、
$\rho = \lambda / \mu$
と表すことができる。ρ = 0.5以下とするためには、1イベント分のデータを3.37μs以内で処理しなければならない。



ルミノシティ 0.2×10^{34}

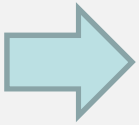
現RODの問題点と改善策

現RODの問題点

・搭載チップが古く、高レートに対応できない可能性がある。

- 規模が小さく、デザインの拡張が難しい。

・すべてHDL記述によって処理を行っている。

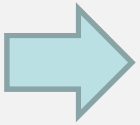


デザインの巨大化

・デバッグが困難。
・問題の特定がしにくい。
・拡張がむずかしい。

改善策

・CPU + ソフトウェアを用いた処理



HDLをモジュール化(単機能化)し、HDL間接続の動的制御を行う。

・デバッグが容易。
・拡張性が高い。

複雑な処理が必要な部分(エラー処理など)をソフトウェアが行う。

FPGAの規模の変化

現RODに搭載されているFPGA

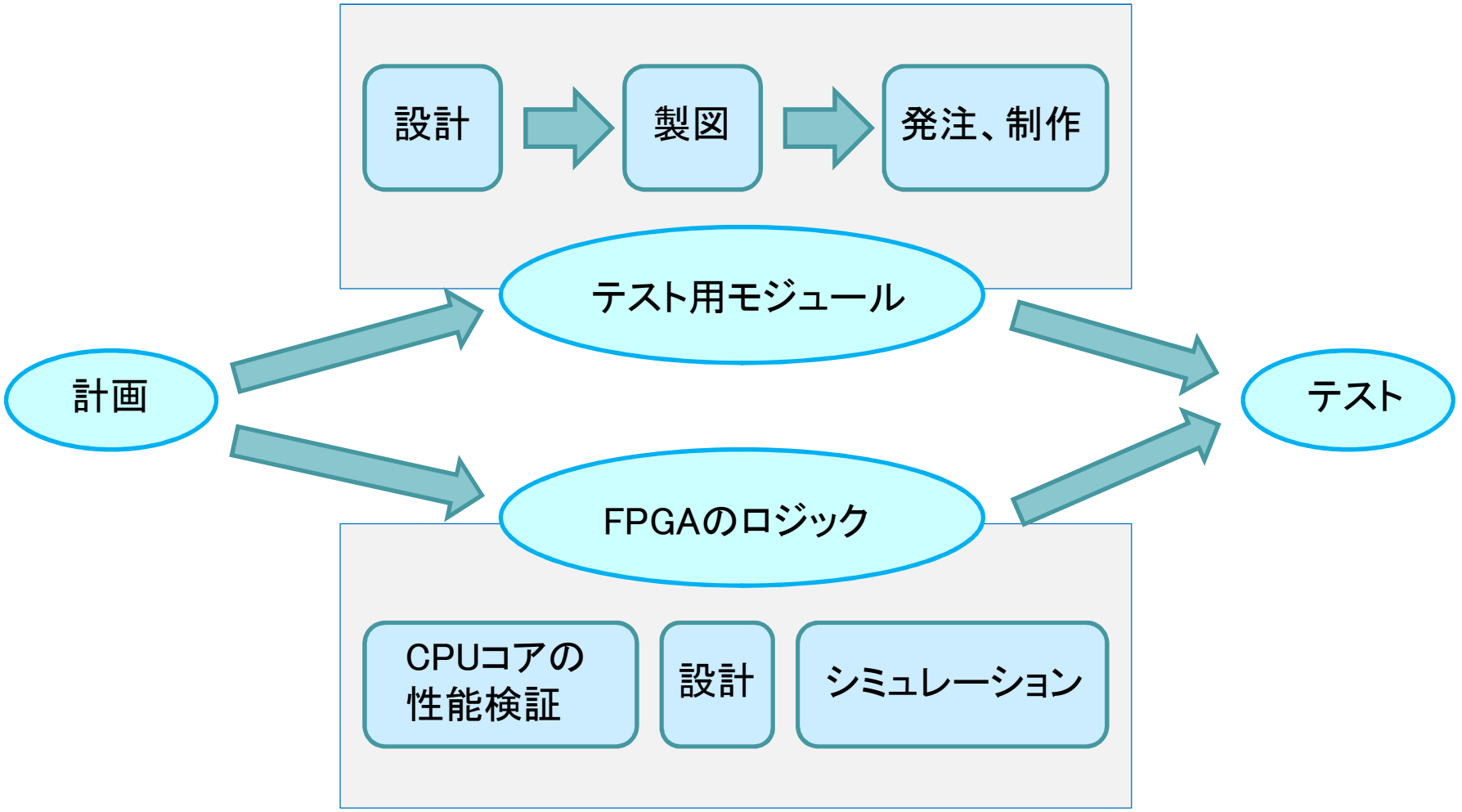
デバイス	システム ゲート	CLB (1 CLB = 4 スライス = 最大 128 ビット) アレイ 行 X 列	CLB スライス	CLB 最大分散 RAM (Kb)	乗算 ブロック	SelectRAM ブロック 18Kb ブロック	SelectRAM ブロック 最大 RAM (Kb)	DCM	最大 I/O パッド(1)
XC2V2000	2M	56 x 48	10,752	336	56	56	1,008	8	624

現在最新のFPGA

デバイス	ロジック セル	CLB (コンギギャブル ロジック ブロック) スライス (1)	CLB 最大分散 RAM (Kb)	DSP48E1 スライス (2)	ブロック RAM ブロック 18Kb (3)	36Kb	最大 (Kb)	MMCM (4)	PCI Express 用インター フェイス ブロック	イーサ ネット MAC(5)	トランシーバ 最大数 GTX	GTH	総 I/O バンク (6))	最大 ユーザー I/O(7)
XC6VLX550T	549,888	85,920	6,200	864	1,264	632	22,752	18	2	4	36	0	30	1200
XC6VLX760	758,784	118,560	8,280	864	1,440	720	25,920	18	0	0	0	0	30	1200
XC6VSX475T	476,160	74,400	7,640	2,016	2,128	1,064	38,304	18	2	4	36	0	21	840
XC6VHX565T	566,784	88,560	6,370	864	1,824	912	32,832	18	4	4	48	24	18	720

スライス数は7倍～10倍に。動作クロック数は430MHzから600MHzに。

新ROD開発計画



テスト用モジュールとFPGAのロジックはパラレルに作成

テスト用モジュールの作成

新RODのデザイン開発のためにFPGA搭載のVMEモジュールを作成。

特徴

- CPUコア、大規模なデザインに耐えられる大型のFPGAを搭載。(XC6LX150T)
- データの送受信に使用するメザニンカードスロットを搭載。
- 分散化、高速化に使用できる高速シリアル通信インターフェースを搭載。
- OS搭載も考慮し、外部メモリとコンソール、Ethernetインターフェースを搭載。

FPGAのロジック設計

新RODのデザイン
新RODでは、CPUコア（MicroBlaze）を搭載することを検討中。CPUコアを搭載することでソフトウェアを用いた動作が可能になる。

CPUコアを用いた処理

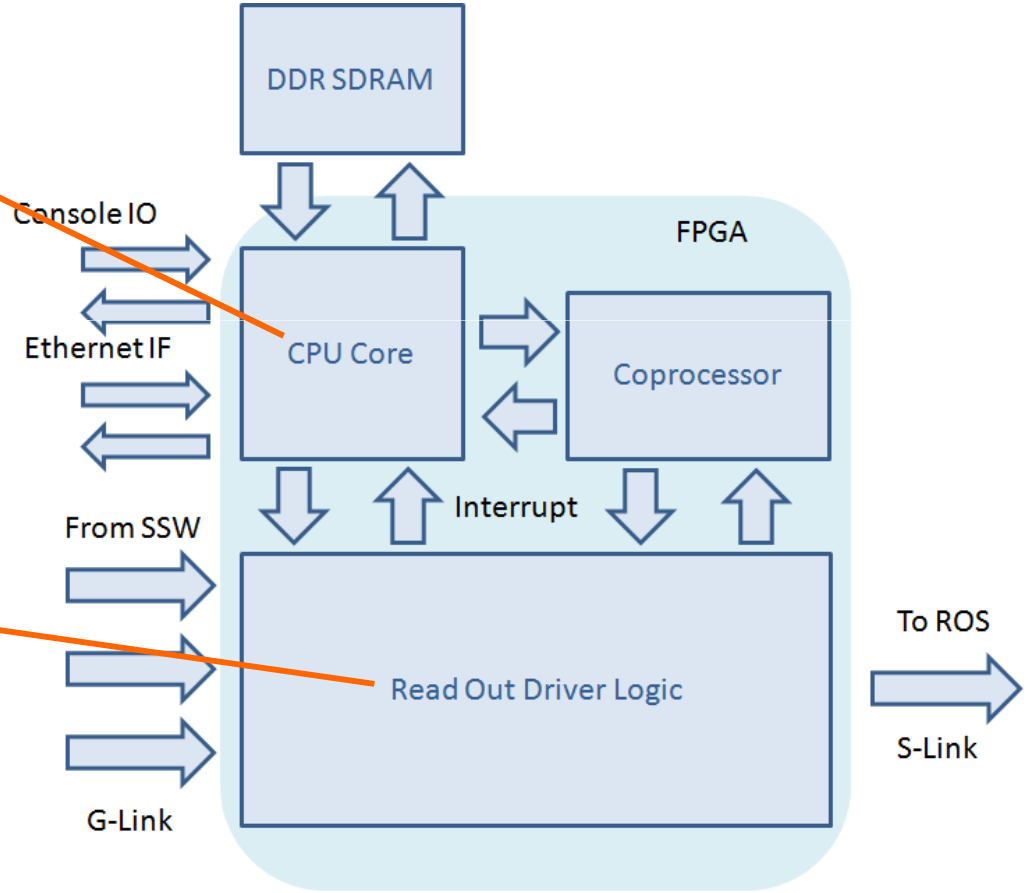
・エラーハンドラー

- データ収集のエラーにより起動
- データフローの停止
- エラー発生時のデータの保管
- エラー内容の解析
- データフローシーケンスの初期化
- データフローの再起動

・システム診断

Read Out Driver Logic

・HDLデザイン
・SSWからのデータを集めて、イベントビルディングを行い、データを送る。



Read Out Driver Logic

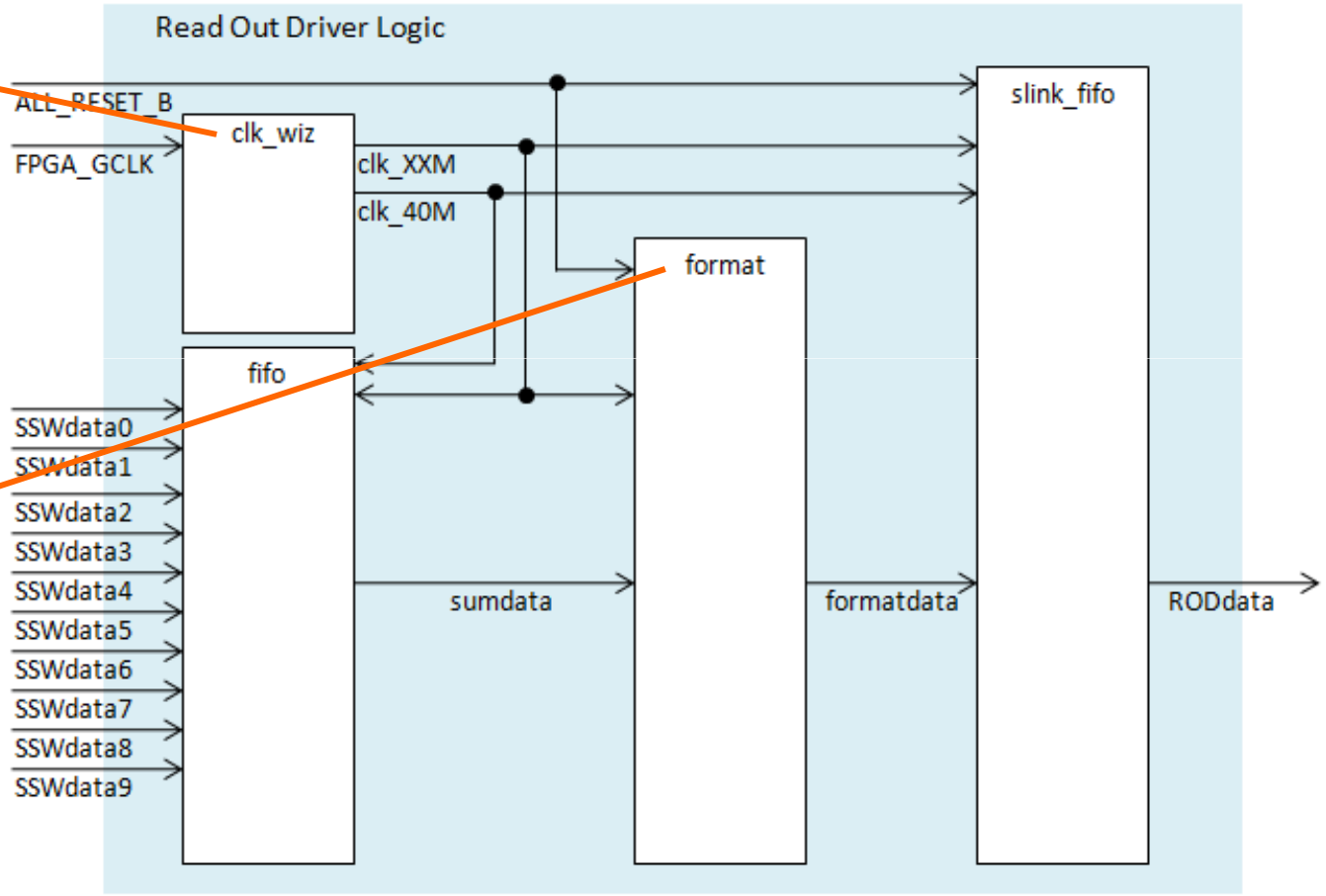
RODの機能
SSWからのデータをATLAS共通のフォーマットに整形してROSにデータを送信する。

Clock Wizard

Clock Generator。外部から供給されるクロックを定数倍して内部の処理を高速化する。

format

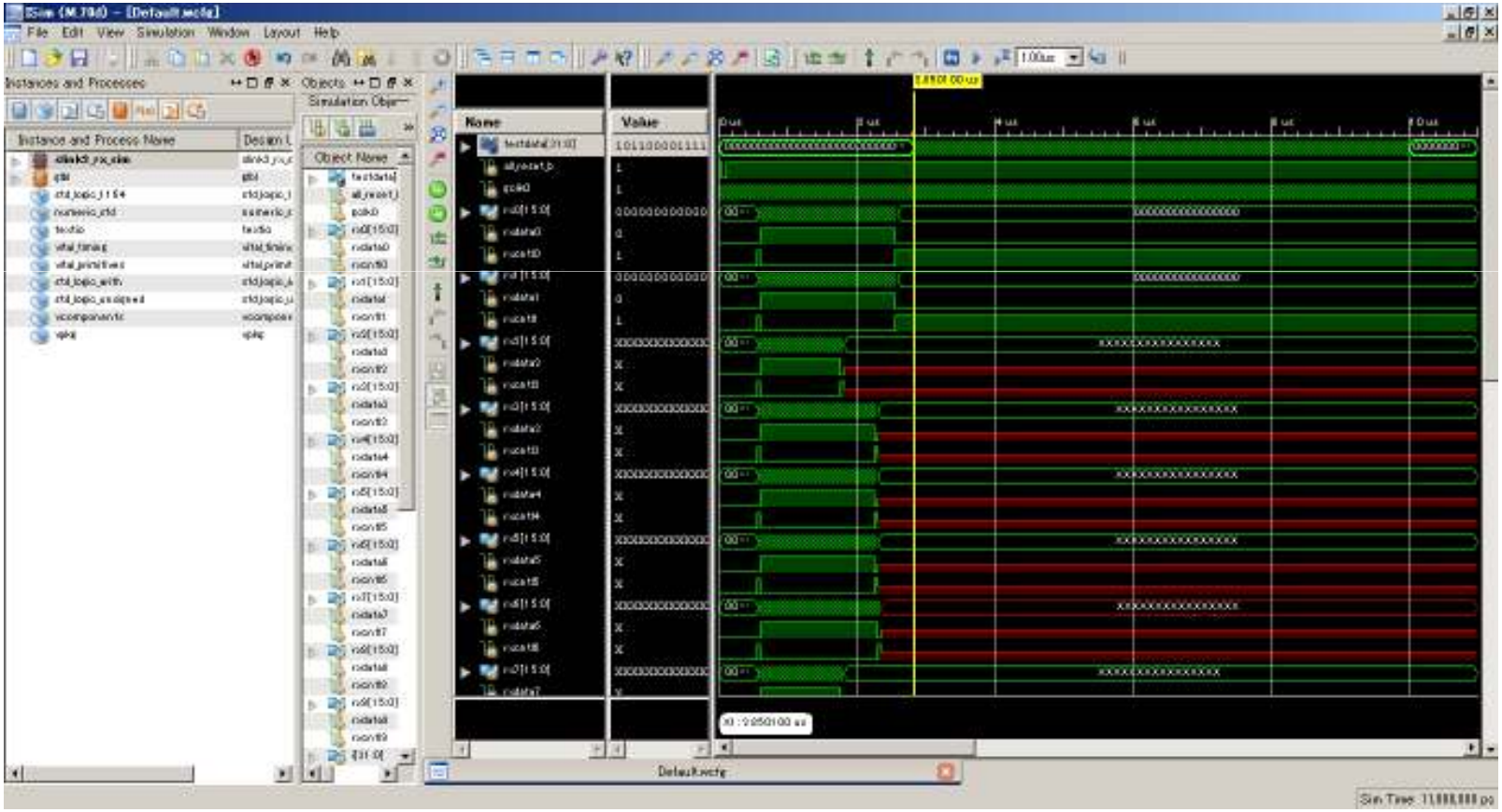
ATLASの共通フォーマットに従って、Headerとtrailerをつける



Read Out Driver Logicのシミュレーション

clk_wiz生成する内部クロックを何Hzで作動させたらよいか、ダミーデータを用い、データ受信から整形するのにかかる時間をシミュレーションを行い求めた。

⇒ 120MHz以上で動作させれば、3.37μs以内で処理可能。



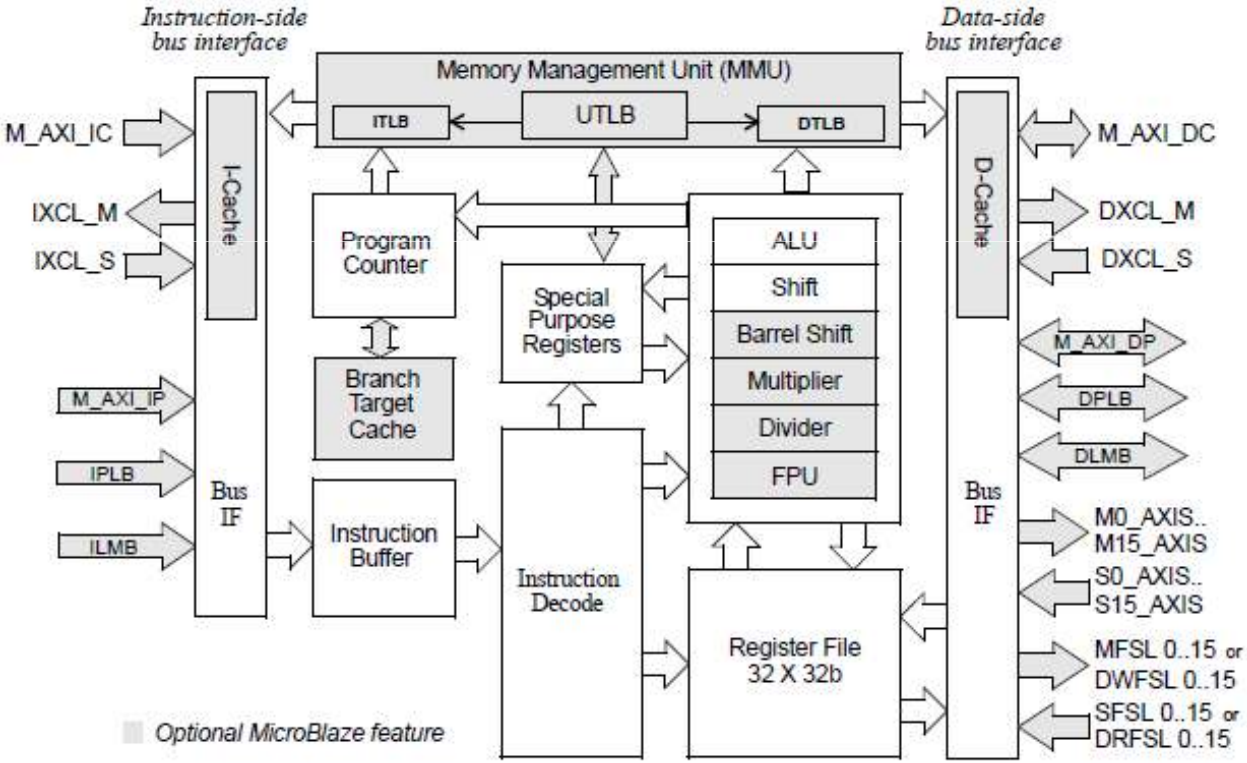
CPUコアの性能評価

MicroBlazeついて

Xilinx社製FPGAに搭載可能なソフトプロセッサコア
32bit RISC型、OSを搭載可（μITRON、uClinux）

ユーザ定義

・クロック周波数
  – Spartan6、最大230MHz
    スライス数 23038
・パイプライン段数
  – 3段 or 5段
・周辺インターフェイス
・メモリー管理ユニット



CPUコアの性能評価

組み込みの利点

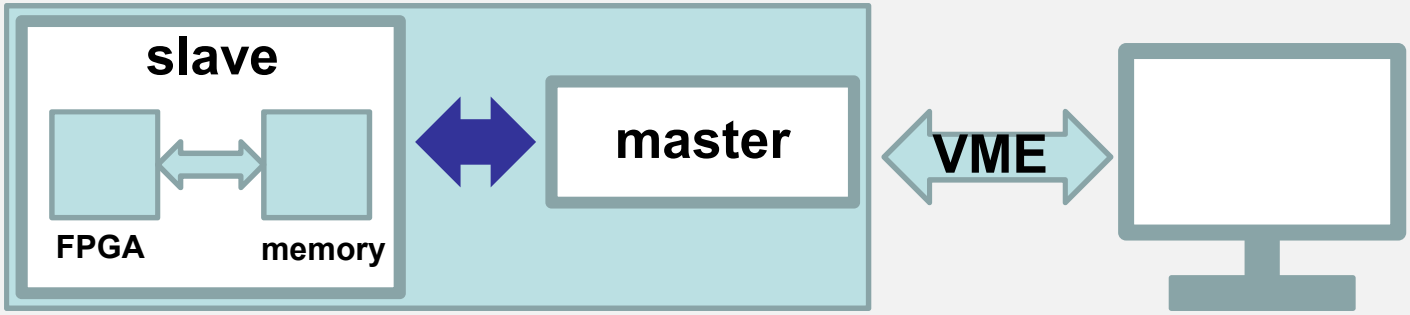
1. 空のfor loopを1000万回処理したときに要する時間

	時間	クロック数	1ループあたり
MicroBlaze (50MHz)	2.6s	1億3000万	13
Pentium4 (2.4GHz)	16.8ms	4000万	4

2. ボード上のメモリへのアクセスと、VME経由でスレイブボード上のメモリアクセス

	時間	クロック数
MicroBlaze	0.94us	47
Pentium4	4.97us	(11,200)

MicroBlazeの方が5倍早く処理が終わっている。これはVMEによるオーバーヘッドも関係している。



CPUコアの性能評価

テストに使用した機器
Spartan6評価ボードSP605。Spartan6 XC6LX45TFGG484、DDR3 SDRAM搭載。

MicroBlazeを用いて行ったテスト
- FIFOバッファからのデータ取り出し
- ボード上のメモリへのアクセス
- デュアルコアを用いた処理

FIFOバッファのデータ取り出し

1. MicroBlazeがデータをプルする。(動作クロック230MHz)

```
XGpio_DiscreteWrite(&flagIO,1,0);

for(i=0;i<100;i++){
   XGpio_DiscreteWrite(&flagIO,1,1);
   data[i] = XGpio_DiscreteRead(&dataIO,1);
   XGpio_DiscreteWrite(&flagIO,1,0);
}
```

1つの関数を使用するのに95クロック必要。
for文を使って100回ループさせたところ12555クロックかかった。従って1回あたり126クロック。
1回に32bitのデータ転送が行えるので、
$230 / 126 \times 4 \times 10^6 = 7.3\text{M}$
従って、この手法では7.3MByte/sで通信できることがわかる。

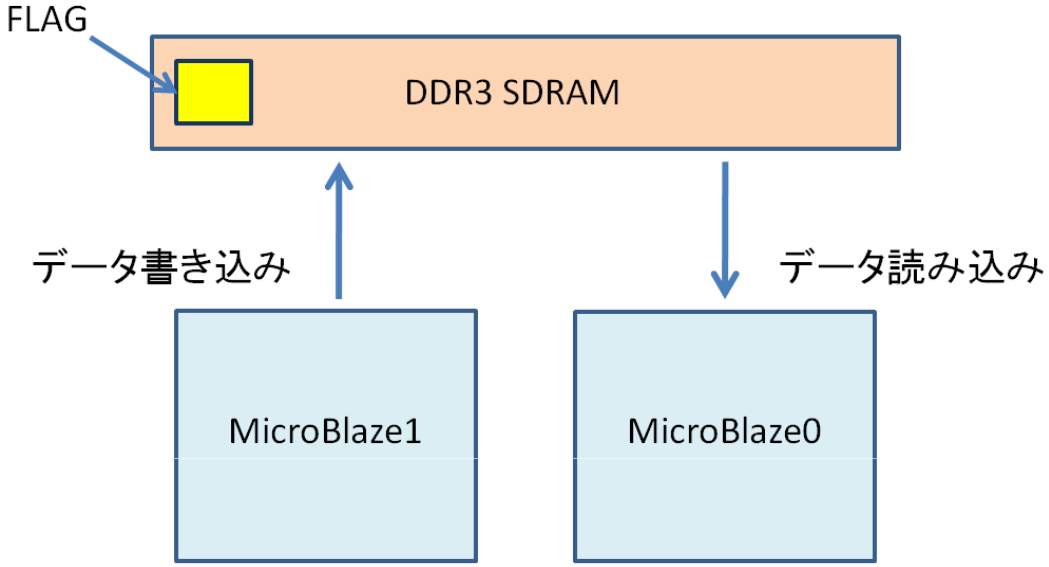
2. HDLデザイン側がデータをプッシュする。(動作クロック230MHz)

ポーリングをしてデータの到着を待つ。
100回繰り返した時のクロック数が17256クロック。同様の計算により、この手法では5.3MByte/sで通信できる。

```
for(i=0;i<100;i++){
   XGpio_DiscreteWrite(&flagIO,1,0);
   while(1){
      flag = XGpio_DiscreteRead(&flagIO,1);
      if(flag == 1){
         XGpio_DiscreteWrite(&flagIO,1,1);
         data[i] = XGpio_DiscreteRead(&dataIO,1);
         break;
      }
   }
}
```

ボード上のメモリへのアクセス

Multi-Port Memory ControllerというIPコアを用いることで簡単に操作できる。

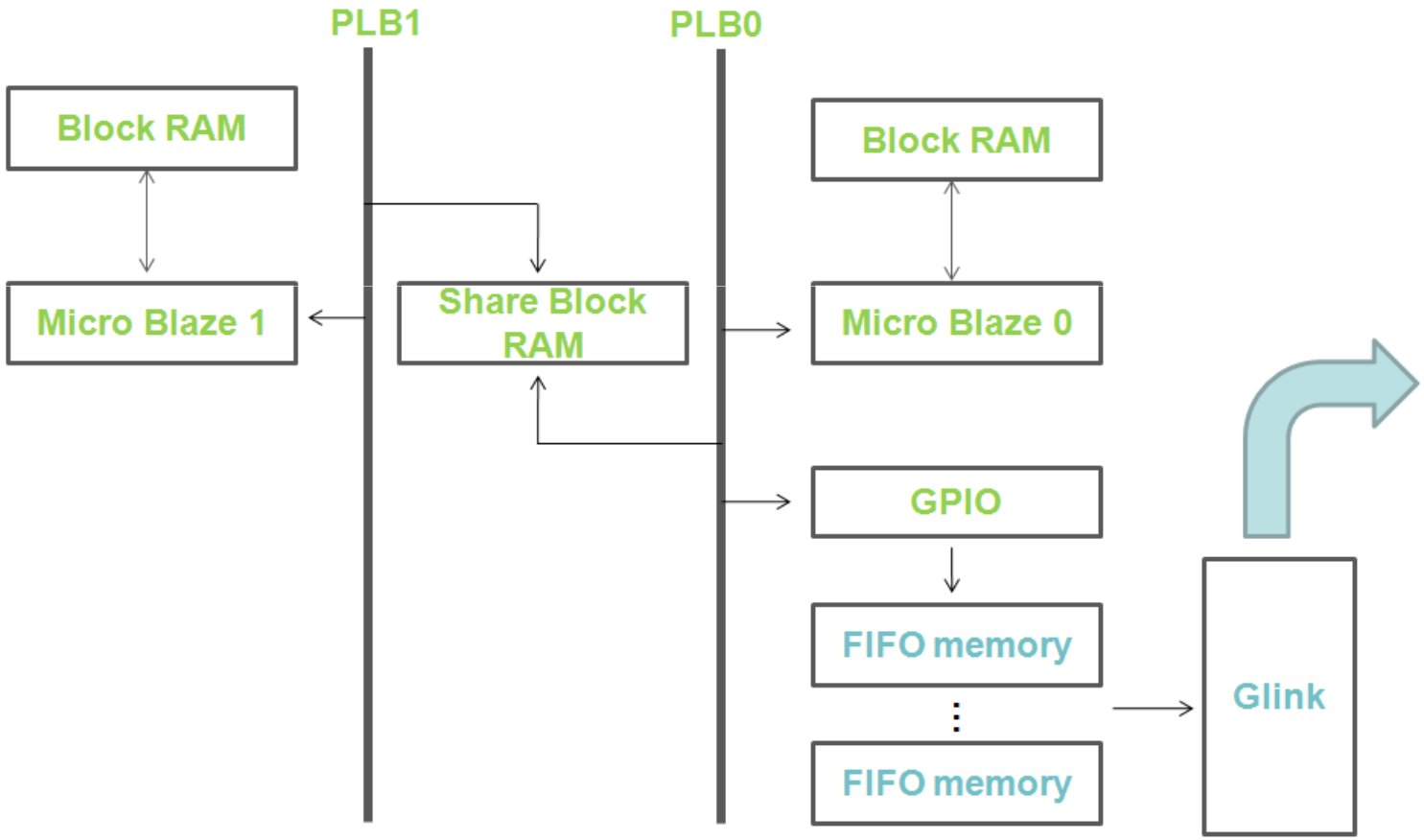


DDR3 SDRAMを2つのMicroBlazeの共用メモリとして使用。
メモリ上にデータの書き込みと読み込みを制御するためにFLAGを設け、2つのMicroBlazeはこの値を参照することで、書き込みと読み込みが同時に行われないようにする。

正しく作動していることを確認

デュアルコアを用いた処理

RODのテストをするために、SSWと同様の働きをするSSWエミュレータを作成。2つのMicroBlazeを使用し、それぞれ役割分担をさせることで処理速度向上を目指した。同様のプログラムをシングルコアで作動させたときの処理速度と比較。



デュアルコアを用いた処理

結果をみると、14%の速度上昇がみられた。このようにMicroBlaze自体の処理速度は低速だが、コア数を増やすことによって処理速度を上げることができる。

	時間	クロック数
single core	22ms	1478万
dual core	19ms (14%up)	1290万

IPコアを使うことで簡単にメモリにアクセスすることができる。ただしSpartan6はSDRAMには対応してないので、注意が必要。
MicroBlazeはソフトプロセッサコアなので、FPGAの容量が許すかぎり、いくつでもMicroBlazeを生成することができる。これにより処理速度を上げることが可能。

- 1つのMicroBlazeを生成するのに約1500LUTsとBRAM1つ使用。
- Spartan6 LX45Tは27288LUTs。
- Spartan6 LX150Tは92152LUTs。

まとめ

データを受け取り、送信するRead Out Driver Logicの枠組みを作ることができた。シミュレーションの結果、内部クロックを120MHz以上にすることで、アップグレード後にも対応できることがわかった。

CPUコアの検証結果では、MicroBlazeを用いることで、設計に幅と柔軟性を持たすことができることがわかった。例えばMicroBlazeとソフトウェアを用いることで、システムの初期化や、エラー発生時のデータの保持、システム診断などを行わせることを考えている。データ通信には5.3MByte/s以上でやりとりできる。
また、実際のRODと同じ環境下を再現するために、テスト用のモジュールを作成した。

これからは具体的なソフトウェアコードを書き、これをFPGAにダウンロードして実際にRODが動いている環境に近い状態でテストを行っていく予定である。

Back-up

OSCについて

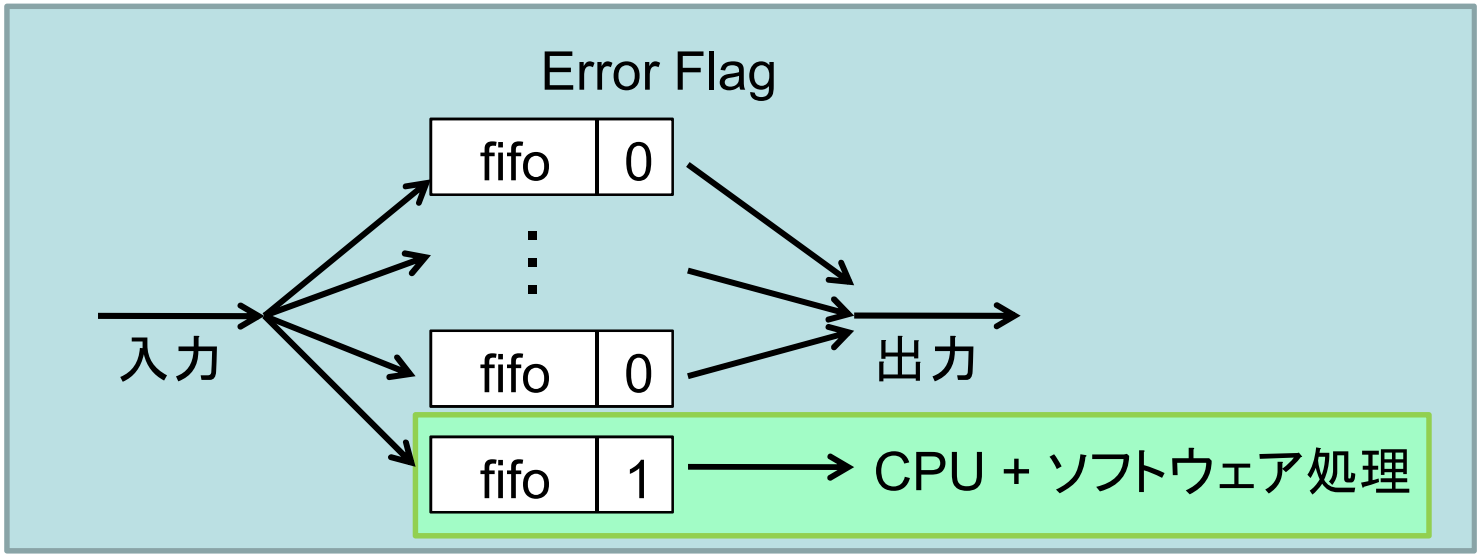
- OSC (Open Source Consortium)とは
  - 計測機システムに必要な技術や知識を
    アカデミック用途のため共有するための組織
  - 本研究開発もこのOSCのプロジェクトの
    1つとして行われています
  - テスト用モジュールの回路図、FPGAの論理回路デザイン、高速
    シリアル通信インターフェースの使用方法についてなど公開予定

http://openit.kek.jp/

ソフトウェアデザインの一例

例：エラーデータを抽出

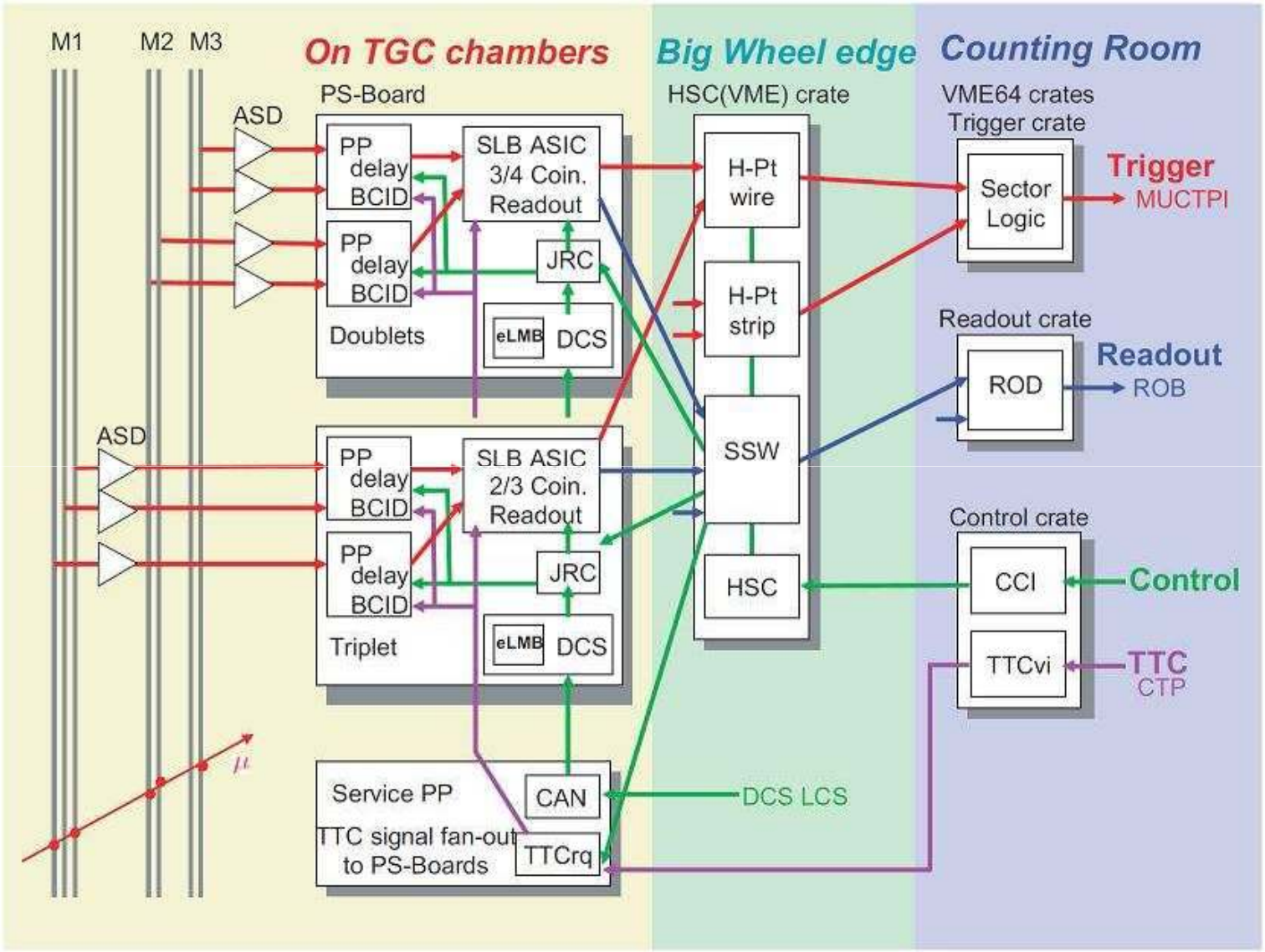
データにエラーがあることがわかったら、データフローから外し、そのデータを抽出する。データは5.3MByte/s以上で通信可能。



XPS Synthesis Summary (estimated values)				
Report	**Generated**	**Flip Flops Used**	**LUTs Used**	**BRAMS Used**
system	水 1 26 01:02:54 2011	2127	2589	4
clock_generator_0_wrapper	水 1 26 01:02:34 2011			
rs232_uart_1_wrapper	水 1 26 00:45:19 2011	146	144	
microblaze_0_wrapper	水 1 26 00:45:10 2011	1281	1573	
dataio_wrapper	火 1 25 21:45:18 2011	270	263	
flagio_wrapper	火 1 25 21:45:08 2011	77	43	
proc_sys_reset_0_wrapper	火 1 25 21:44:58 2011	69	54	
mdm_0_wrapper	火 1 25 21:44:54 2011	126	127	
lmb_bram_wrapper	火 1 25 21:44:33 2011			4
ilmb_cntlr_wrapper	火 1 25 21:44:29 2011	2	6	
dlmb_cntlr_wrapper	火 1 25 21:44:25 2011	2	6	
dlmb_wrapper	火 1 25 21:44:21 2011	1	1	
ilmb_wrapper	火 1 25 21:44:15 2011	1	1	
mb_plb_wrapper	火 1 25 21:44:11 2011	152	371	

Device Utilization Summary (actual values)				
Slice Logic Utilization	**Used**	**Available**	**Utilization**	**N**
Number of Slice Registers	1,688	54,576	3%	
Number used as Flip Flops	1,683			
Number used as Latches	0			
Number used as Latch-thrus	4			
Number used as AND/OR logics	1			
Number of Slice LUTs	1,987	27,288	7%	
Number used as logic	1,777	27,288	6%	
Number using O6 output only	1,478			
Number using O5 output only	47			
Number using O5 and O6	252			

Design Summary


M1
M2
M3
On TGC chambers
Big Wheel edge
Counting Room
PS-Board
HSC(VME) crate
VME64 crates
Trigger crate
ASD
PP delay BCID
SLB ASIC 3/4 Coin. Readout
JRC
eLMB
DCS
Doublets
H-Pt wire
H-Pt strip
Sector Logic
Trigger
MUCTPI
Readout crate
ROD
Readout
ROB
SSW
SLB ASIC 2/3 Coin. Readout
Triplet
Control crate
CCI
Control
HSC
TTCvi
TTC
CTP
Service PP
CAN
DCS LCS
TTC signal fan-out to PS-Boards
TTCrq
μ

Event Header 000 New Record Type=01

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
000			RecType		SSWID				RX mask pattern (1=enabled, 0=disabled)																							

Record Type (RecType) is 01 in this format version, hard-wired in FPGA.
SSWID is arbitrarily set by a dip-switch on each SSW board.

SLB header 010

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
010			SLBID						0	BCmap			Mod Type		0	L1ID															

BCmap shows 3BC data lines taken by RX. 3bit shows (next, current, previous) events. 1=adopted, 0=discarded.
SLBID, Mod Type, L1ID and BCID are all SLB's data. See SLB documents.

SLB header 011 –0

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
011			0	0			RXID				0		RX FIFO status			SLB-OVF											RX-OVF				

RXID is RX identified number from 0 to 22.
RX FIFO status tells what amount of data are stored in RX FIFO then.
SLB-OVF is SLB's data. See SLB documents.
RX-OVF is RX-FIFO overflow counter. This tells the snapshot value when this word is sent from RX to TX.

SLB trailer 001 –1 This word appears after SLB data words only when there is an error

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
001		1	ovf	ssu	LVDSlnk				RX error state						

LVDSlnk=LVDS links status. 2bits are (now,old). 1=Not linked, 0=Linked.
SSU = SLB SSU flag. See SLB documents.
OVF = RX-FIFO overflow flag. If OVF=1, some overflows have happened in this RX data.

SLB data 100, 101, 110

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
100	cell address							cell bitmap							
101	cell address							cell bitmap							
110	cell address							cell bitmap							

In any order:
Cell data for Current BC data
Cell data for Previous BC data
Cell data for Next BC data

PAD word 110

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
110	11111							0							

i.e. 0xDF80

Event Trailer 111

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
111			0x1CA									Glnk	T1C	NRC	T2C	XOR check sum															

Glnk = Glink TX status. "Locked" signal of Glink Tx. 1=Not locked, 0=Locked
T1C = Timeout1_count_flag. Time-out to collect the event fragment from all the enabled input ports.
NRC = Nores_count_flag. No response from RX RFO of "enabled" (not masked) input port.
T2C = Timeout2_count_flag. Time-out to collect the event fragment from each enabled RX FIFO.
These three flags are reset at every event.

The XOR operation includes the first word(1000s) of the event header through the first word of the event trailer. When the result is XOR'ed with the XOR checksum word, the result becomes zero. (the XOR does not include the 0x03C0F or 0x0C0F framing words)

Framing

Each event is proceeded by the 32-bit word 0x0000003F and followed by the 32-bit word 0x03C0F003, both words are sent in Glink control mode.