

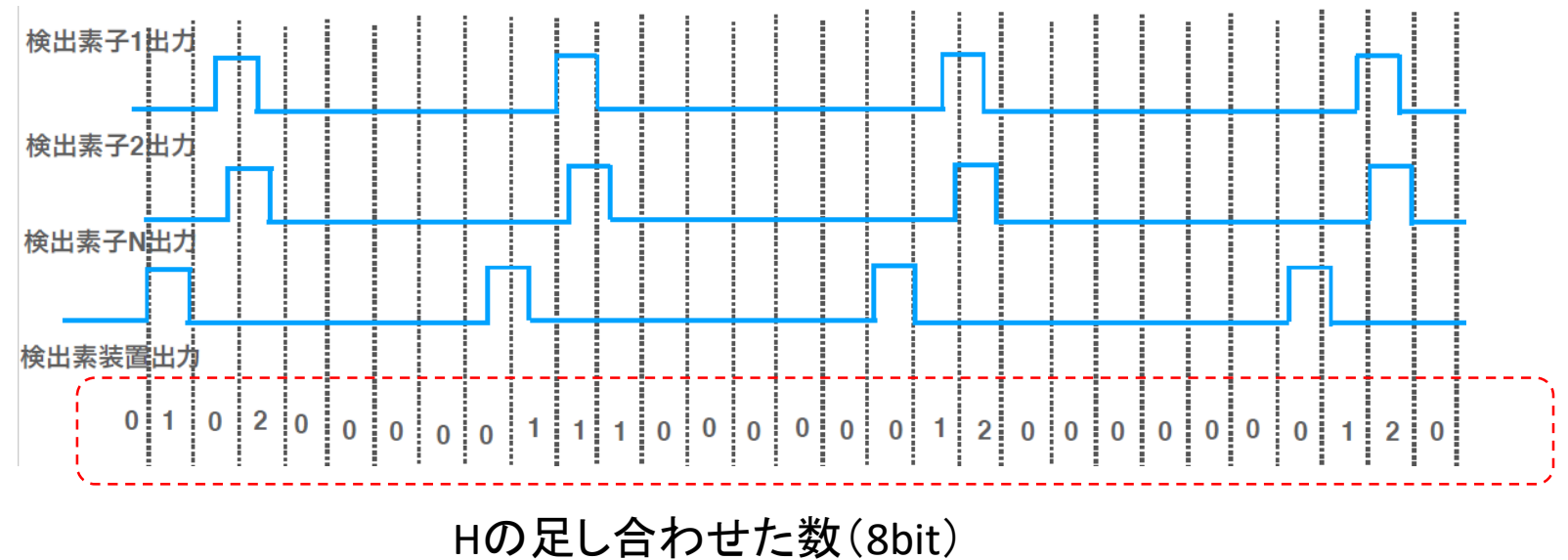
中性子センサ FPGAエミュレータ 仕様書

2020/11/18

仕様概要

- LまたはHの1bitの信号を255個の素子から受信する

N=255
と仮定



- 周期的に0から255までの信号(Hの足し合わせた数)を作成し、TCP/Ethernetでその信号を出力する
- 周期は最大10MHzで、スローコントロールにより設定変更を可能とさせる

使用するボード

Xilinx Evaluation Board KC705

- FPGA

Xilinx Kintex-7 XC7K325T

- ネットワーク
1Gbps Ethernet

- Pin

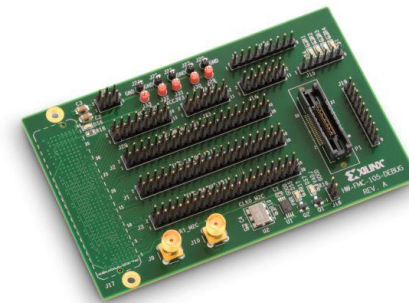
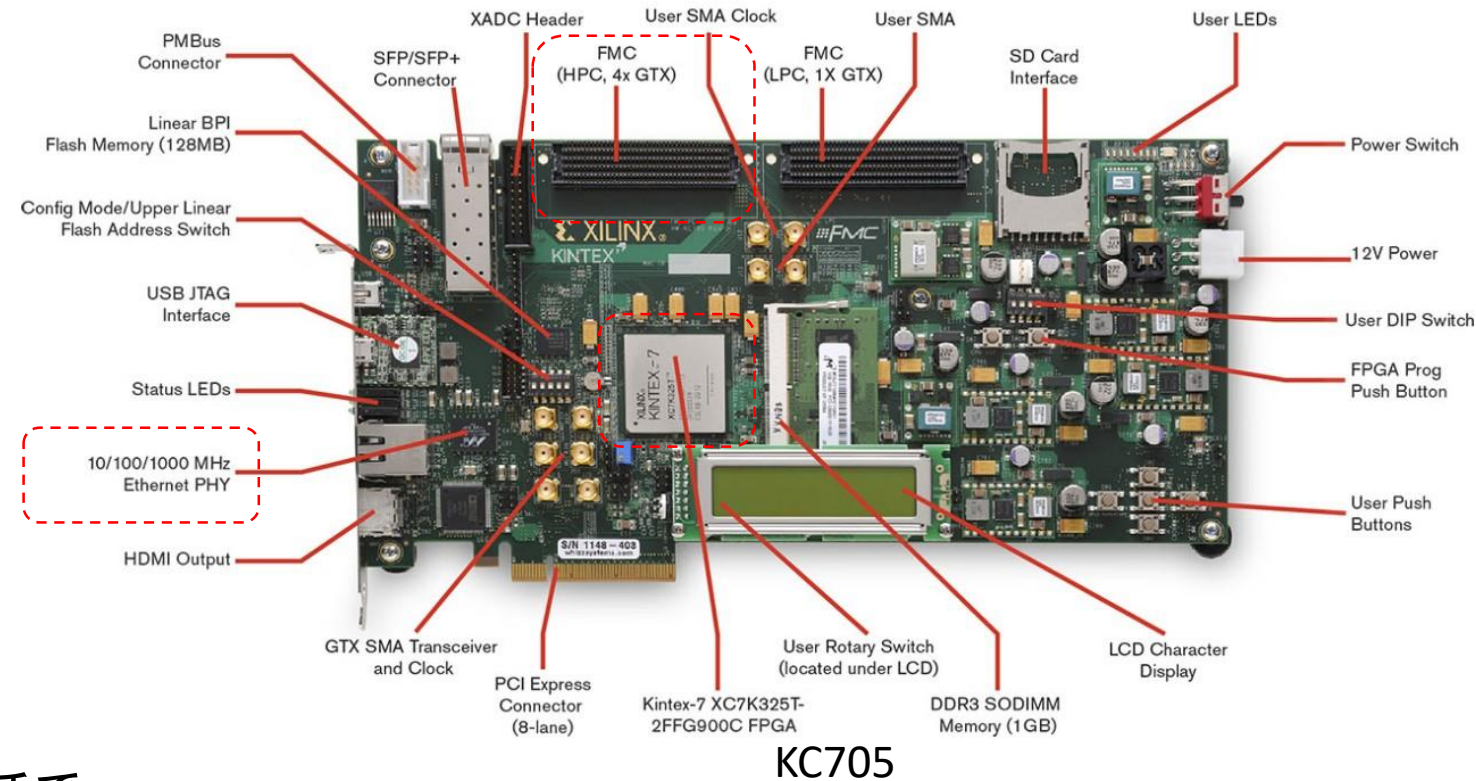
FMCを搭載

拡張ボード(FMC XM105)を利用すれば2.54ピッチで30ペアのLVDS信号の入出力が可能

電源電圧は1.8V, 2.5V, 3.3Vから選択が可能
(デフォルトは2.5V)

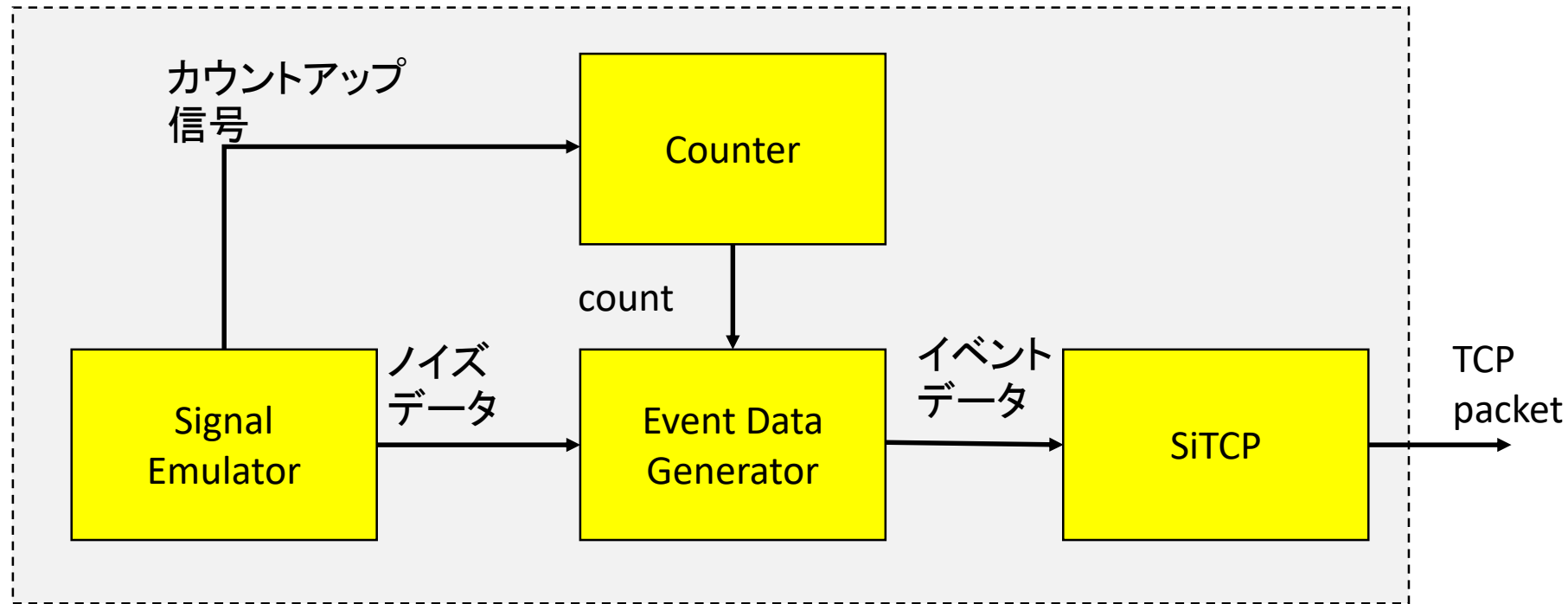
- 価格

\$1,695 (xilinx公式ホームページより)



拡張ボード(FMC XM105)

FPGAファームウェア ブロック図



- Signal Emulator: 最大周波数10MHzで8bit (0から255まで) のノイズデータをEvent Data Generatorに送る。また、ノイズデータを生成するたびに、カウントアップ信号をCounterに送信。
- Counter: 最大周波数10MHzのカウンタ。カウントアップ信号を受信するたびに、カウンタが+1される。また、Count(7bit)の値をEvent Data Generatorに送る。
- Event Data Generator: Count(7bit)及びsignal dataを受信、次ページで示すイベントデータを作成し、SiTCPへ送る。

データフォーマット

Event Data Generatorで作成するイベントデータのフォーマットを右図に示す。

- Flag(1bit)

前のサンプルデータと連続している場合は0、そうでない場合は1とする

- Count(7bit)

1サンプルごとに+1されていく。127になると0に戻る

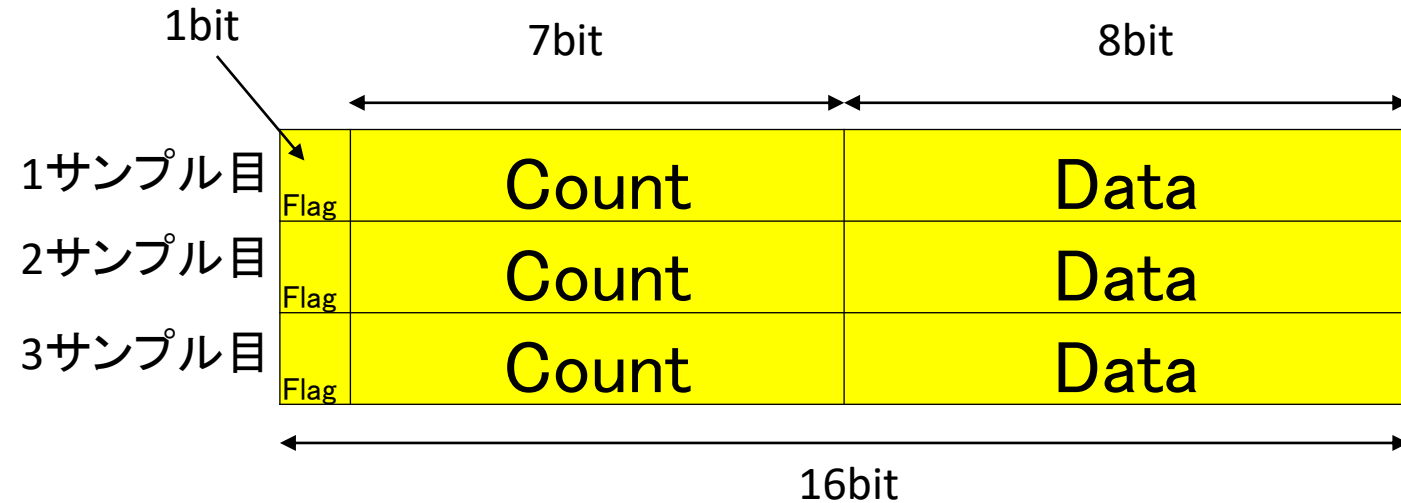
- Data(8bit)

ノイズデータ

※16bitのデータを常にこのイベントデータを送り続ける。

周波数を10MHzに設定した場合、データレートは160Mbps(= 16bit × 10MHz)

※もし、ネットワーク上の問題で、一部のデータを送ることができない場合はFlagの値を見ることによって、そのことが分かるようにする。



Signal Emulator

- Hになる信号はポアソン分布に従った乱数を用いて、出力される
- 発生頻度(ポアソン分布の λ の値)はスローコントロールで設定が可能(全ch共通)

1chあたりのHになる信号の発生頻度(ポアソン分布の λ の値)の設定範囲

- Max: 0.97 [個/100ns]
- Min: 0.022 [個/100ns]

レジスタ設定

SiTCPのスローコントロールを用いて、FPGA内のレジスタ値の設定を行う。

- frequency: 2BYTE

周波数設定用のレジスタ

$10\text{MHz} \div (\text{frequency} + 1)$ の値で周波数を決定させる

- DataEnable : 1bit

TCPでコネクションしている時に、この値がHになることでデータが送信されるようになる

- lamda : 8bit
- random seed: 8BYTE


APPENDIX

Signal Emulator: FPGAにおける乱数の発生方法

Xorshiftを採用

wikipedia(<https://ja.wikipedia.org/wiki/Xorshift>)より

Xorshiftは疑似乱数列生成法の1つである。George Marsagliaが2003年に提案した。演算が排他的論理和とビットシフトのみであるため高速である、などの特徴がある。

```
uint32_t xor64(void) {  
    static uint64_t x = 88172645463325252ULL;  初期値  
    x = x ^ (x << 13); x = x ^ (x >> 7);  
    return x = x ^ (x << 17);  
}
```

周期は $2^{64} - 1 \rightarrow$ 11年以上測定しないと、元の値に戻らない

この乱数生成方法を使用し、8bit(0から255)の乱数を作成する。

Signal Emulator: ポアソン分布と指数乱数

ポアソン分布

$$P(X = k) = \frac{\lambda^k e^{-\lambda}}{k!}$$

指数乱数 $t = -\frac{1}{\lambda} \log(1 - r_n)$ r_n は0~1の乱数

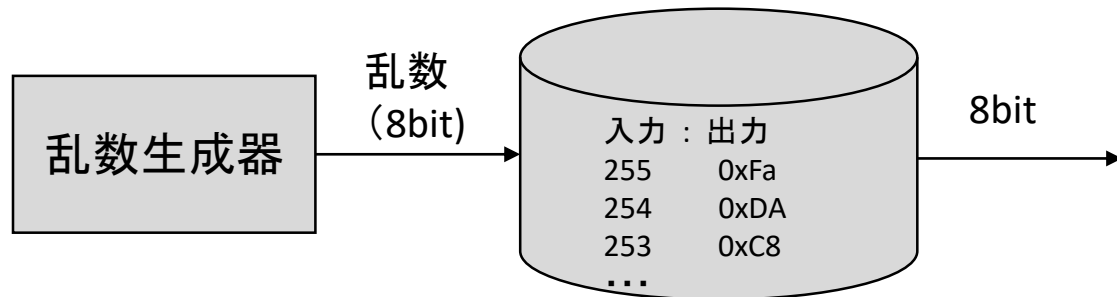
式を変形

$$t = -\frac{45}{\lambda} \log\left(\frac{256 - r_n}{256}\right)$$

r_n は0~255の整数の乱数
45 × 100nsの間に信号が来る期待値

$$t = \underbrace{((-45) \times \log\left(\frac{256 - r_n}{256}\right))}_{\text{ROMに保存}} \div \lambda$$

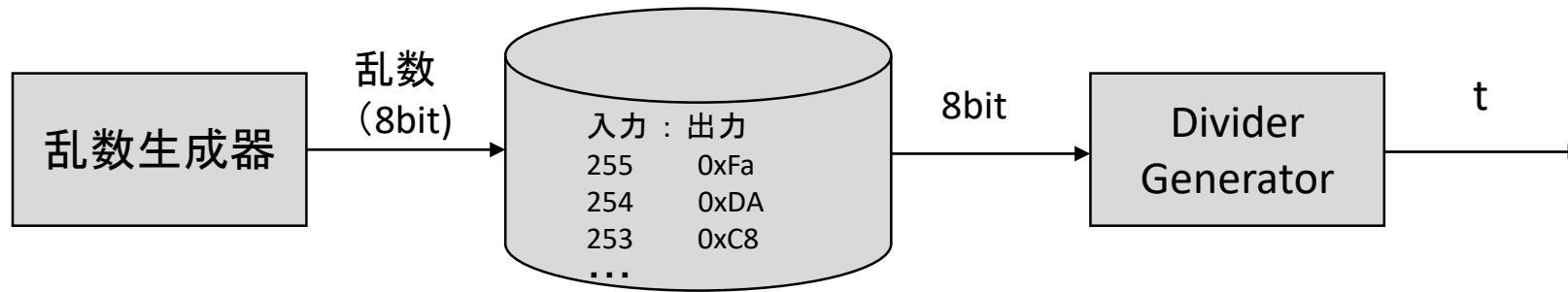
ソフトウェアであらかじめ計算し、ROMにその値を保存



Signal Emulator: 割り算

割り算の計算には、Divider Generator (xilinxのIPの1つ)を利用

$$t = ((-45) \times \log\left(\frac{256 - r_n}{256}\right)) \div \lambda$$



xilinx公式HPより

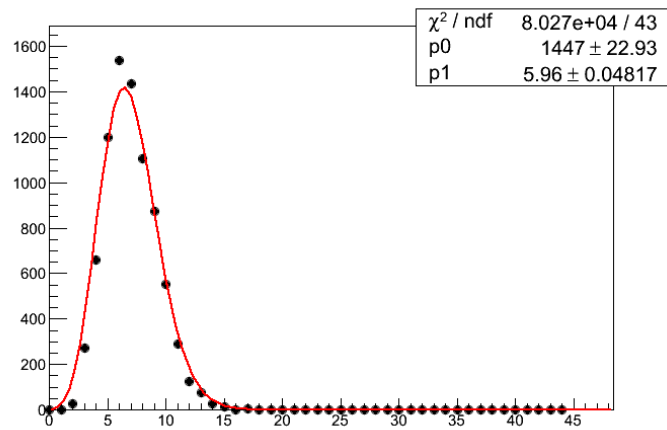
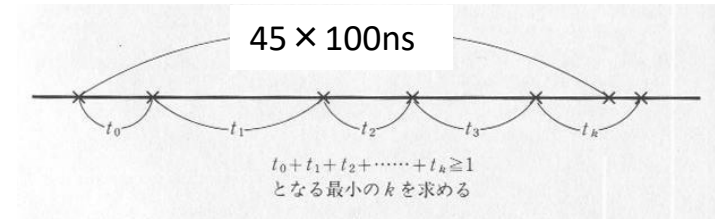
製品説明

Divider Generator LogiCORE™ IP は、リソース効率が高く高性能な整数除算機能をインプリメントできる画期的なソリューションです。整数の除算は、Radix-2 引き放し (non-restoring) 除算またはブリスケーリング除算アルゴリズムで High-Radix 除算を使ってインプリメントできます。

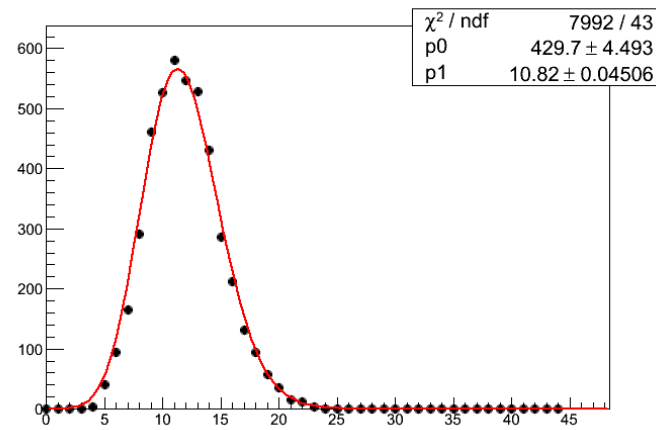
このIPを利用し、小数第一位を四捨五入してtを計算

Signal Emulator: シミュレーション結果

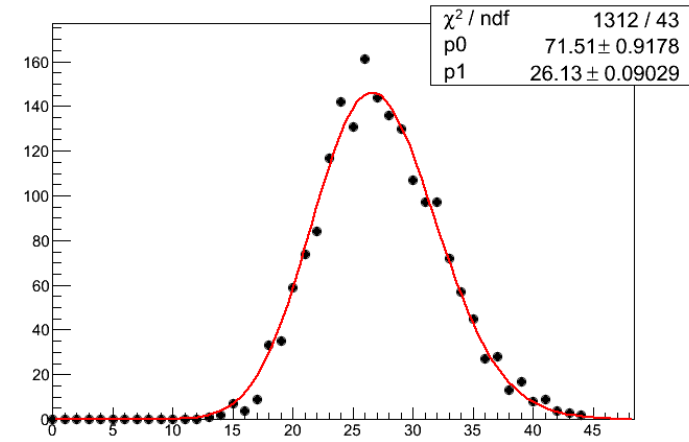
45 × 100ns内に発生する信号数をvivadoでシミュレーション



$\lambda = 5$



$\lambda = 10$



$\lambda = 10$

※黒い点がシミュレーション結果、赤い線がポアソン分布によるフィッティングの結果