きわめる。拓く。創り出す。

長崎総合科学大学

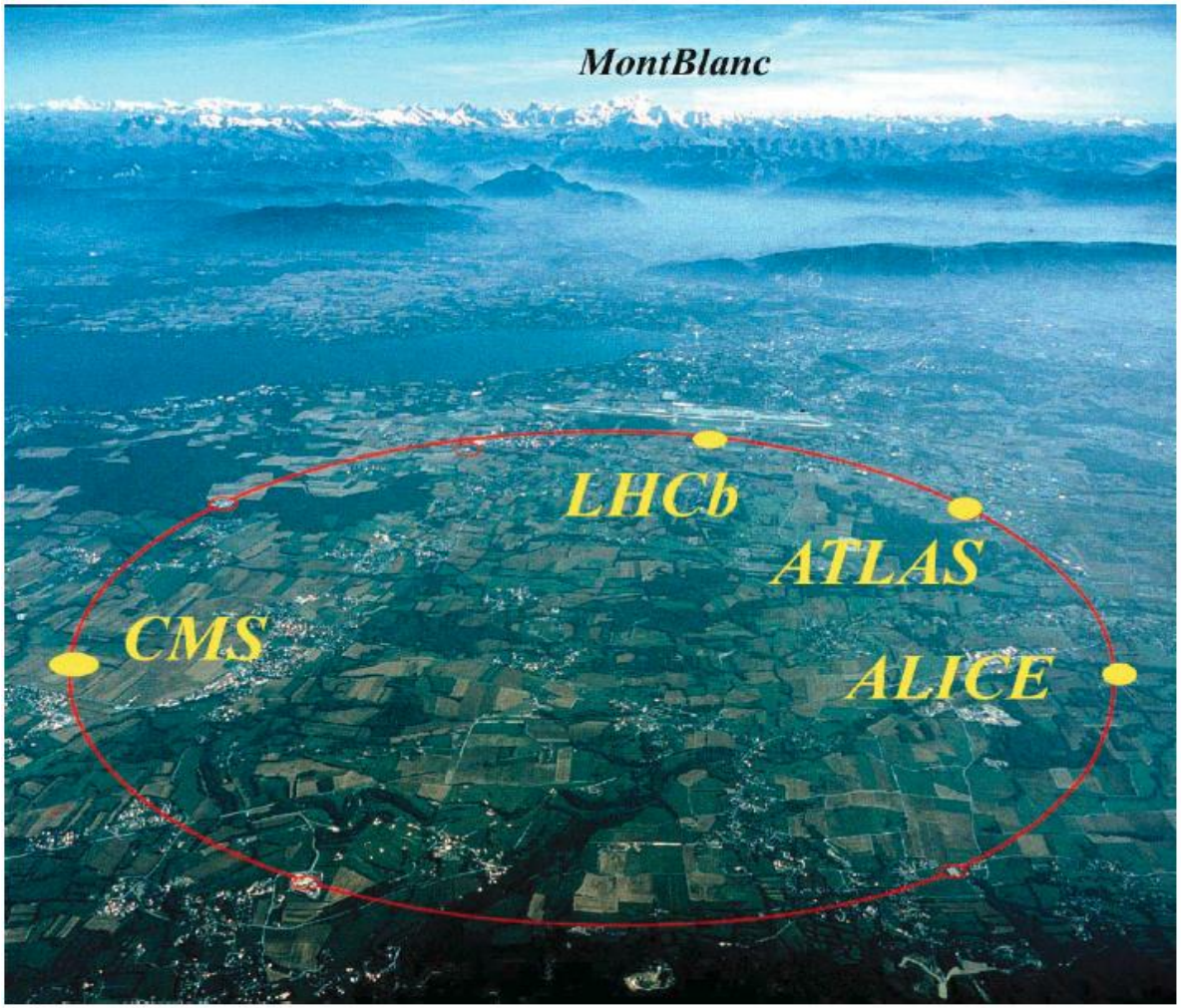
Nagasaki Institute of Applied Science

電気電子工学コース　中間発表

FPGAで動作する中央値を用いたフィルタ作成

田中・大山　研究室所属
電気電子工学コース
1815013　松山裕輝

CERNのLHC



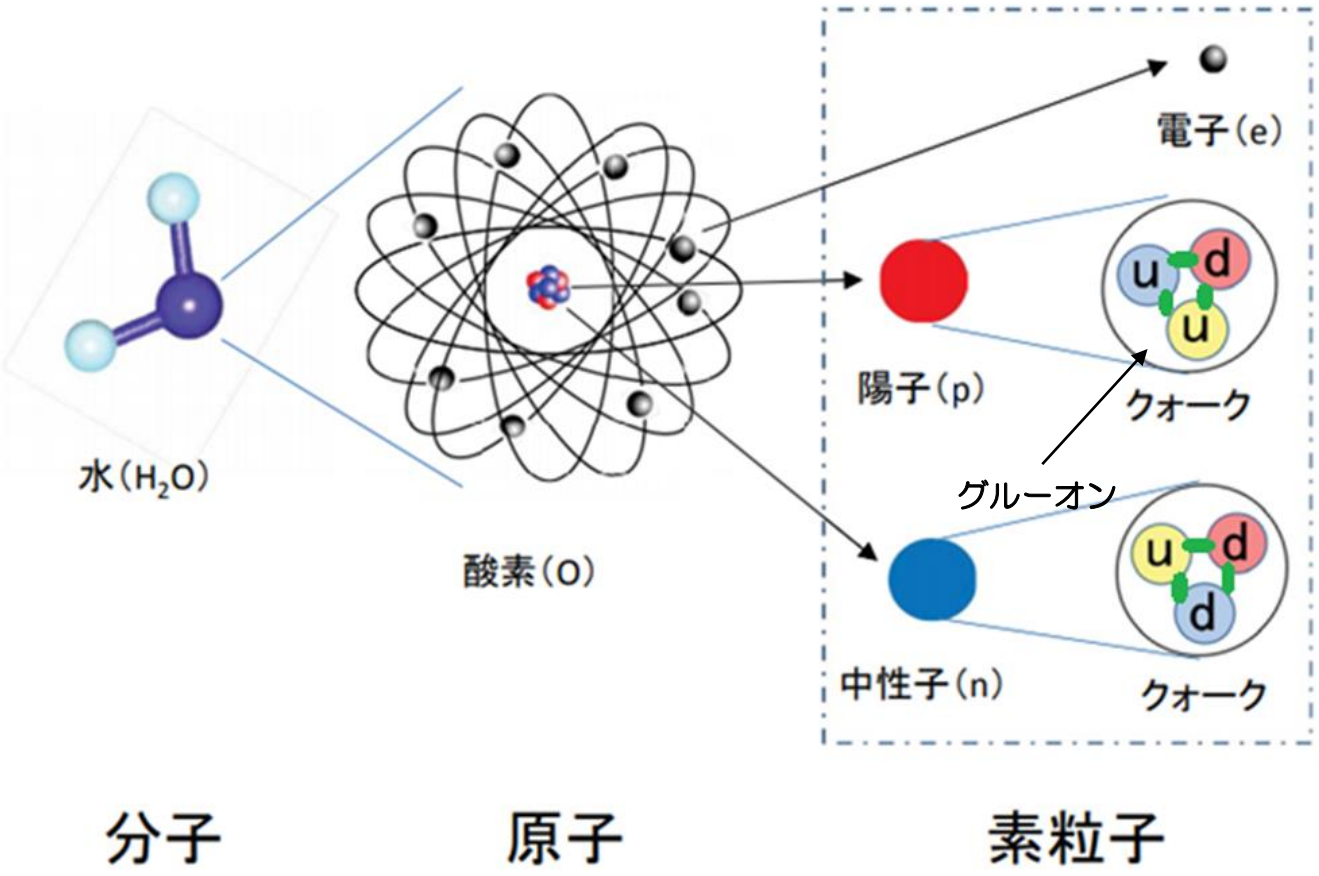
周長　26.6 km

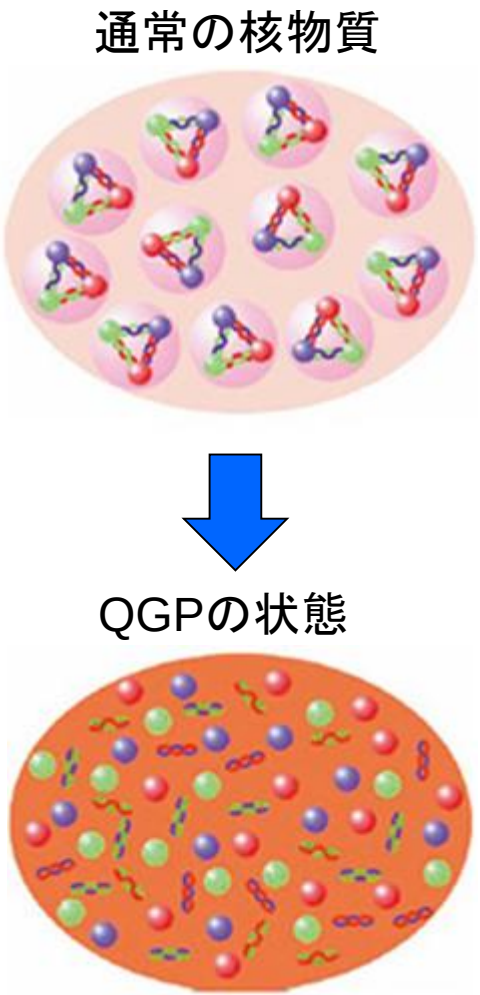
主な実験装置
- ALICE
- ATLAS
- CMS
- LHCb

計画承認　1994
建設完成　2008

http://atlas.kek.jp/public/Visitor5g_novideo.pdf

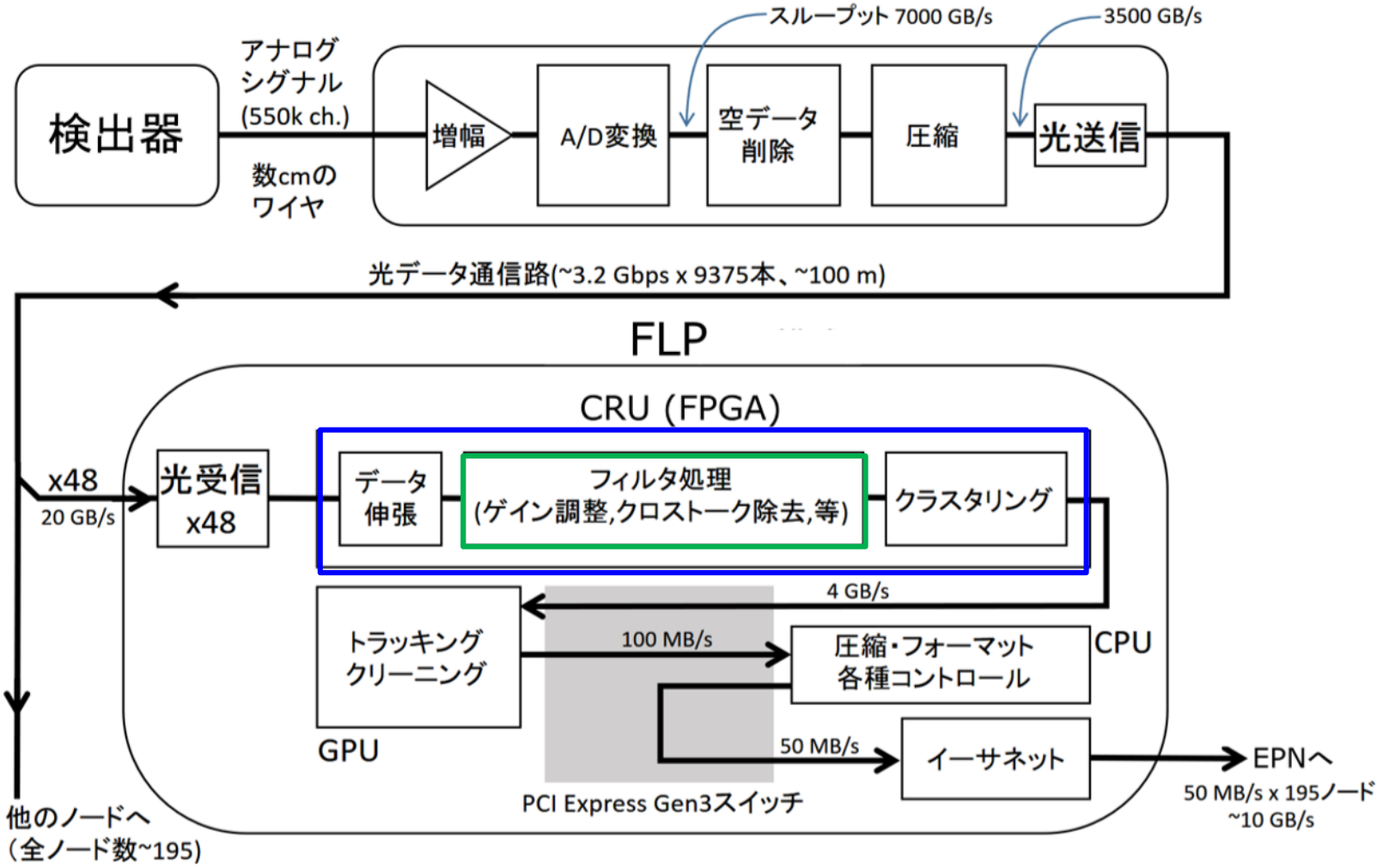
ALICE実験におけるQGPの生成



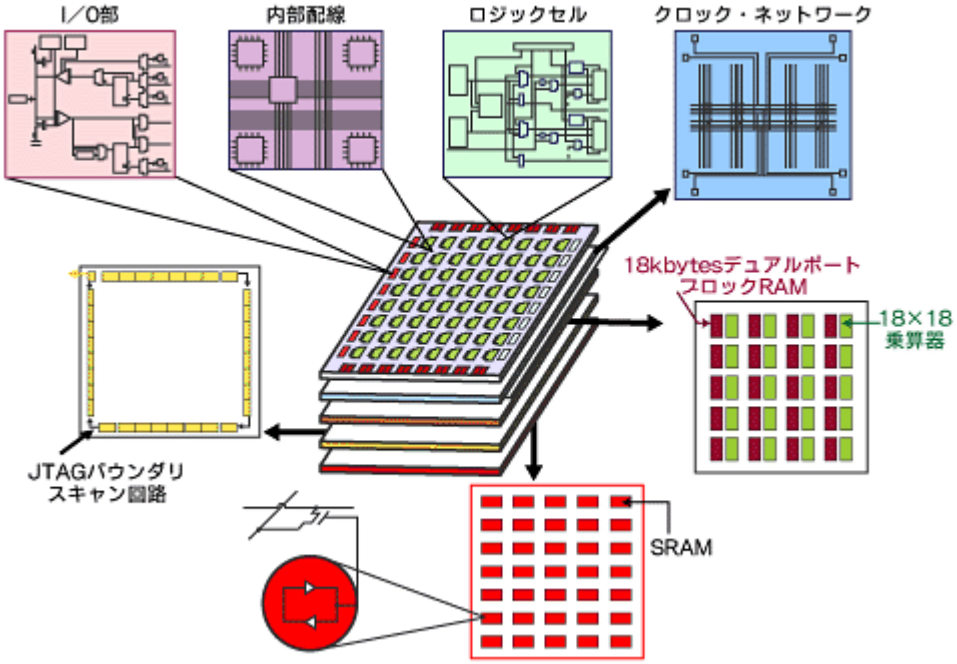


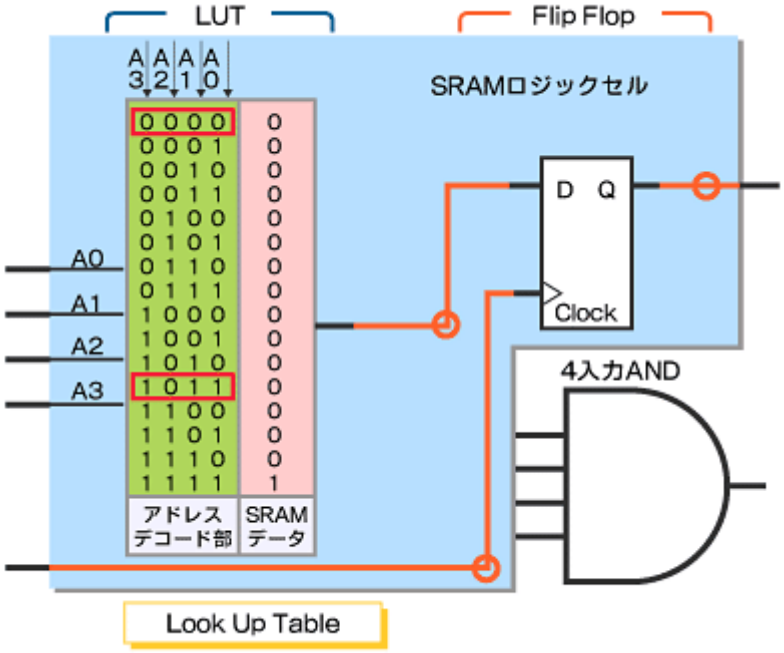
https://www.bnl.gov/riken/research/QGP.php

検出器からのデータフロー



FPGAとは



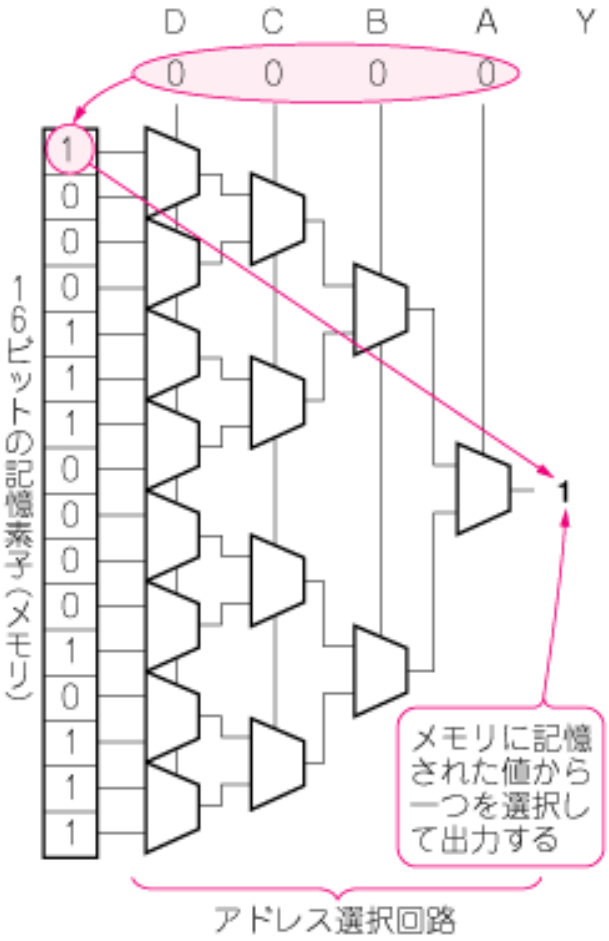


http://monoist.atmarkit.co.jp/mn/articles/0609/20/news118_2.html

FPGAのLUT

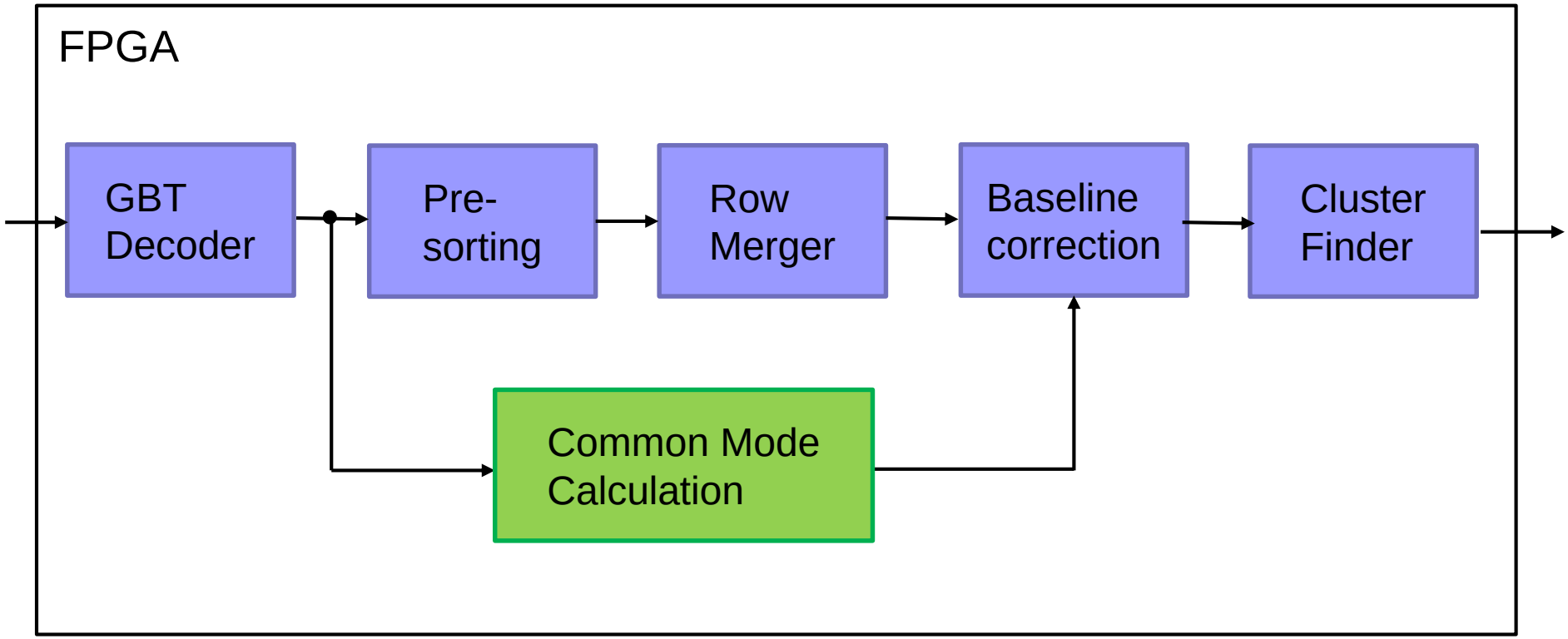
入力の組み合わせに対応して出力が決まる

入力				出力
A	B	C	D	Y
0	0	0	0	1
0	0	0	1	0
0	0	1	0	0
0	0	1	1	0
0	1	0	0	1
0	1	0	1	1
0	1	1	0	1
0	1	1	1	0
1	0	0	0	0
1	0	0	1	0
1	0	1	0	0
1	0	1	1	1
1	1	0	0	0
1	1	0	1	1
1	1	1	0	1
1	1	1	1	1



http://www.kumikomi.net/archives/2009/04/fpgalsifpga.php

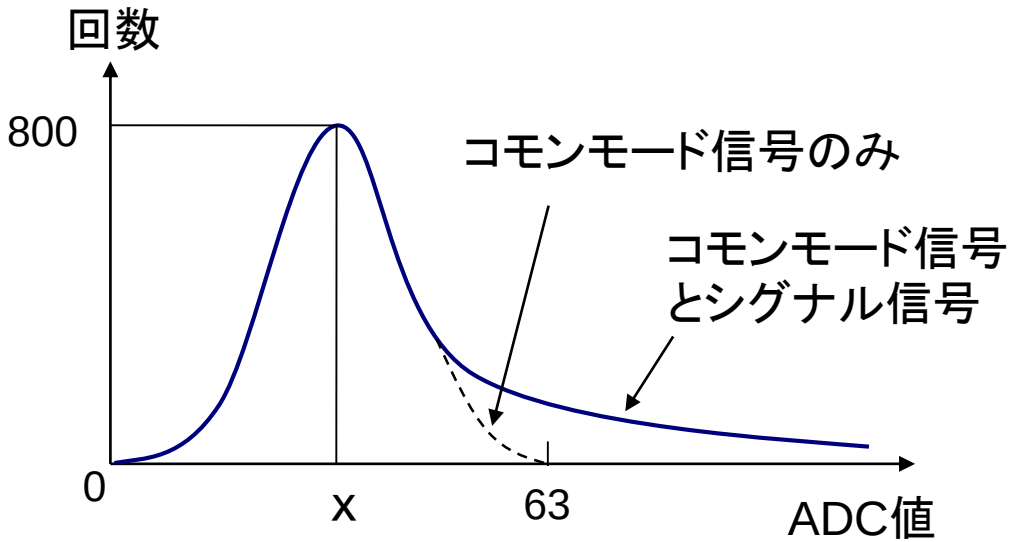
研究で用いるFPGAの処理



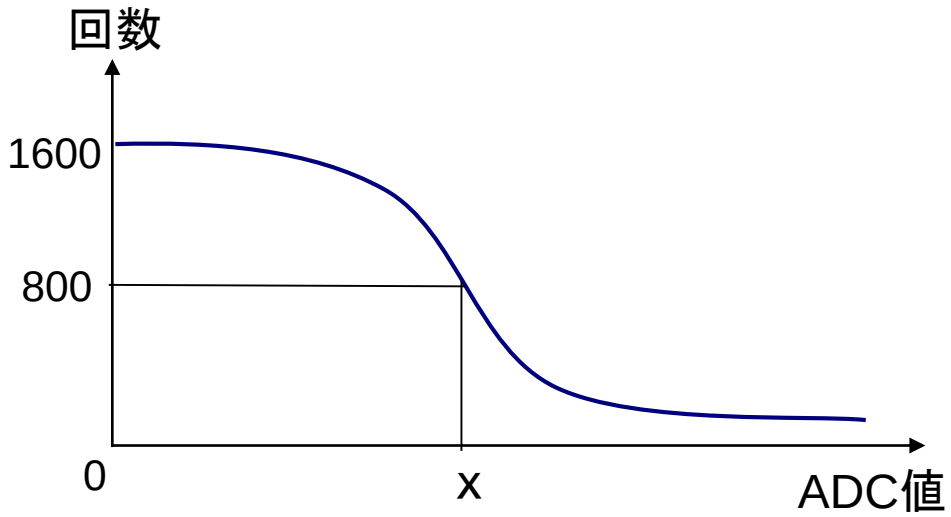
フィルタの種類と特徴

	平均値を用いたフィルタ	中央値を用いたフィルタ
仕組み	ノイズの平均値を算出してADC値から減算を行う	ノイズの中央値を算出してADC値から減算を行う
特徴	全チャンネルの内、閾値を超えた場合をシグナルとして判断する	全チャンネルを算出に利用する
メリット	回路が小さい	正確な値を算出できる
デメリット	正確な値を算出できない恐れがある	回路が大きい

中央値フィルタの仕組み

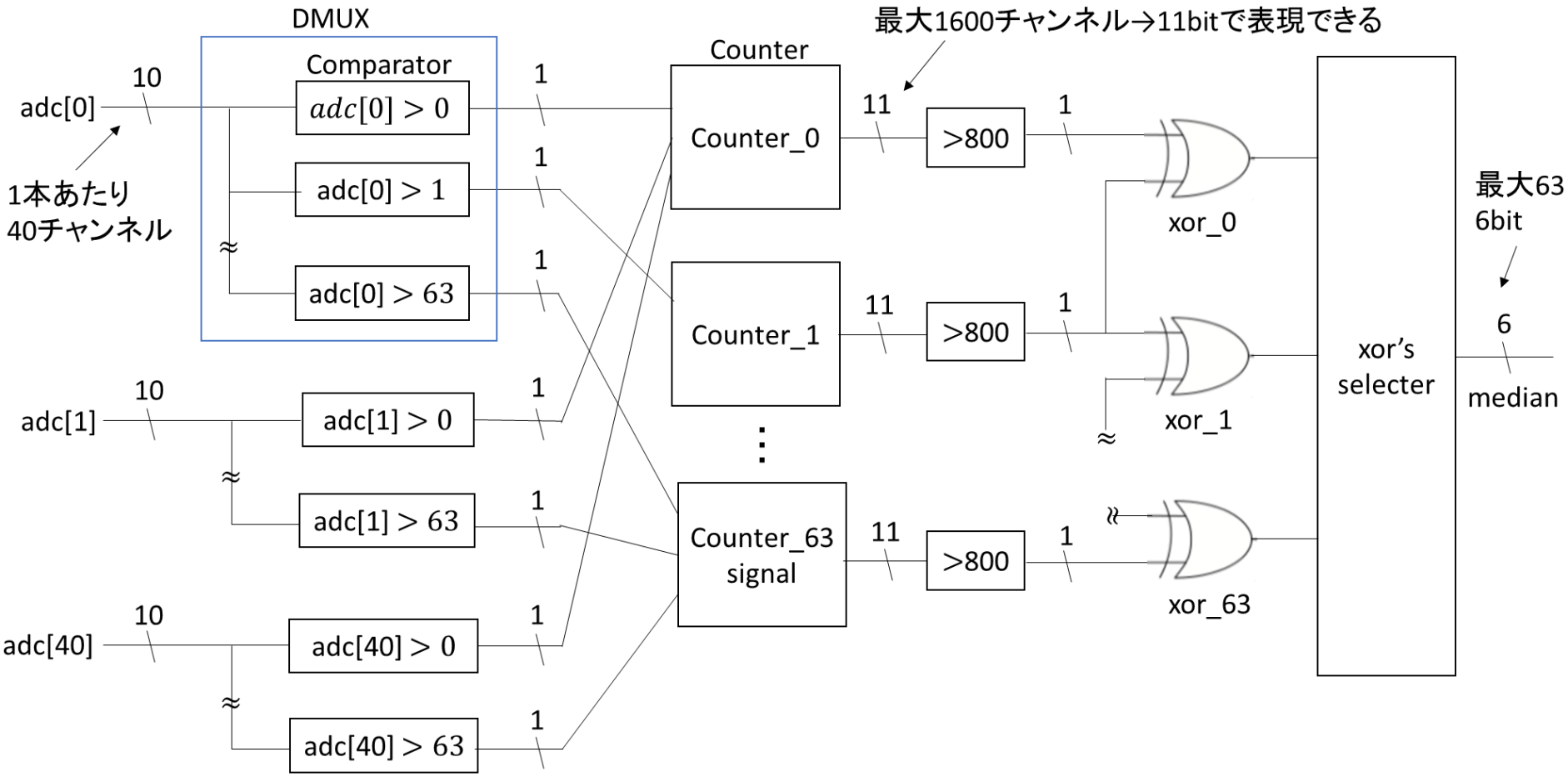


確率密度関数

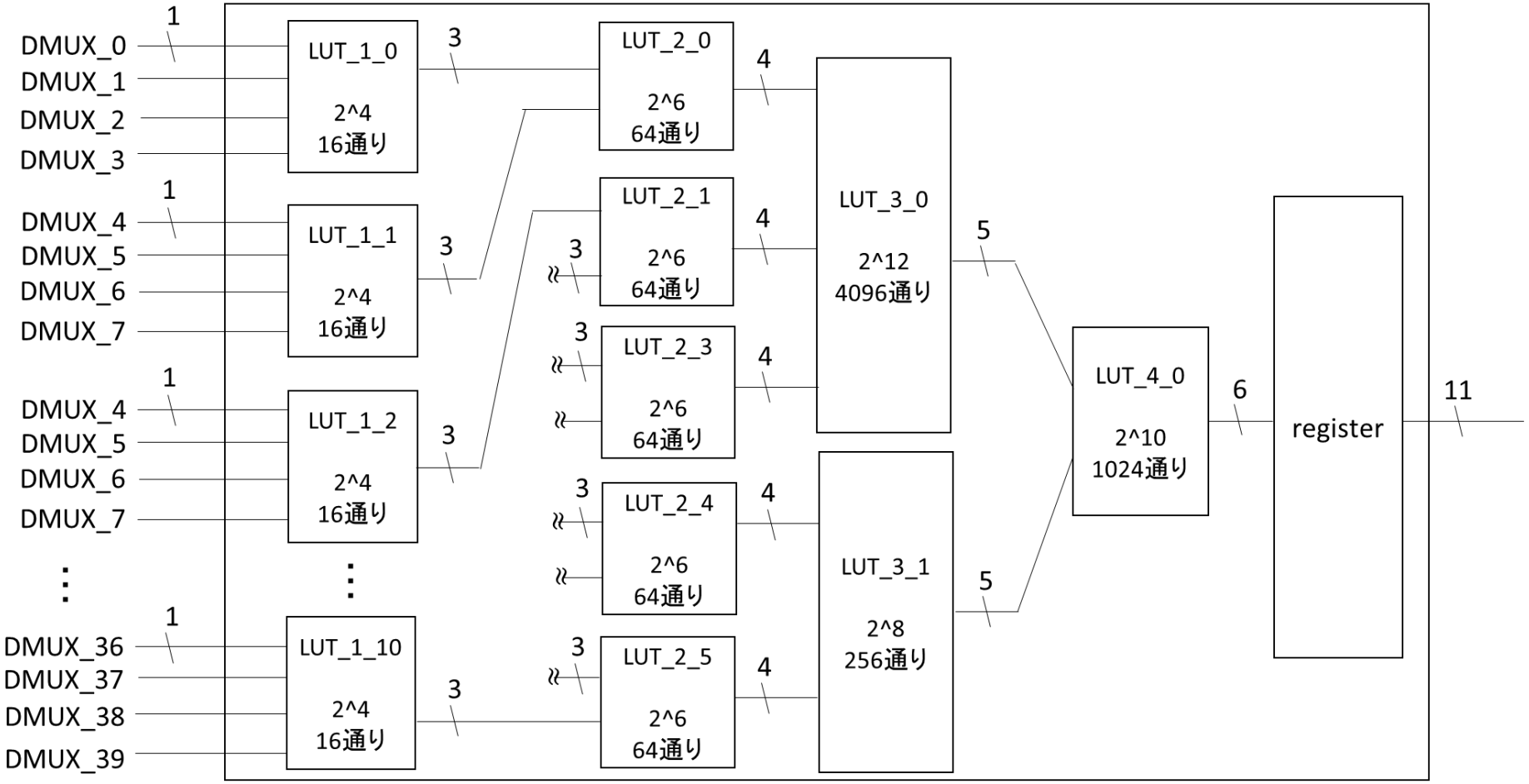


累積分布関数

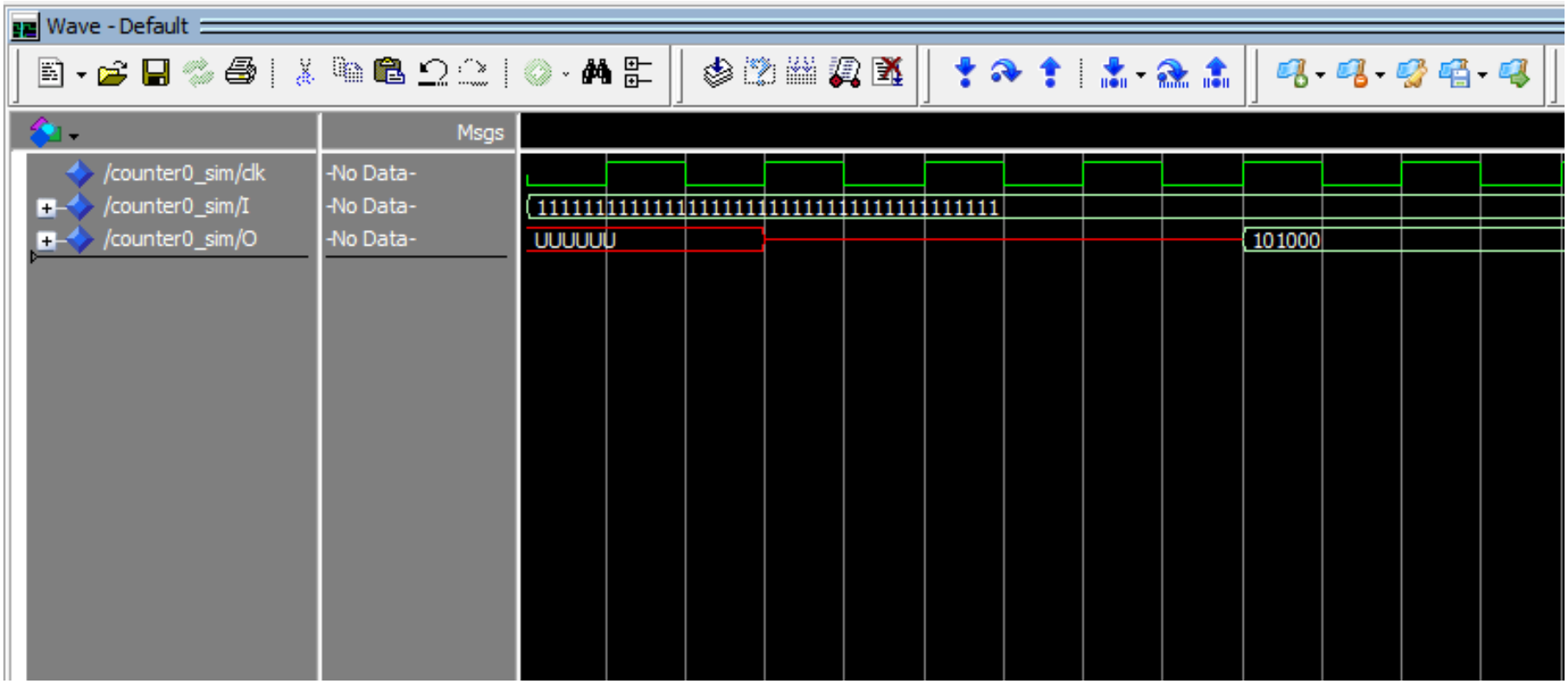
中央値フィルタのブロック図



カウンタのブロック図


DMUX_0
DMUX_1
DMUX_2
DMUX_3
DMUX_4
DMUX_5
DMUX_6
DMUX_7
DMUX_4
DMUX_5
DMUX_6
DMUX_7
DMUX_36
DMUX_37
DMUX_38
DMUX_39
1
LUT_1_0
2^4
16通り
LUT_1_1
2^4
16通り
LUT_1_2
2^4
16通り
LUT_1_10
2^4
16通り
3
LUT_2_0
2^6
64通り
LUT_2_1
2^6
64通り
LUT_2_3
2^6
64通り
LUT_2_4
2^6
64通り
LUT_2_5
2^6
64通り
4
LUT_3_0
2^12
4096通り
LUT_3_1
2^8
256通り
5
LUT_4_0
2^10
1024通り
6
register
11

カウンタの機能シミュレーション



まとめ

- 中央値を用いたフィルタの理論を練り、ブロック図を作成した
- カウンタ部分のブロック図の作成を行い、VHDLでカウンタのコードを作成した
- カウンタの機能シミュレーションを行った

今後の課題

- 中央値を用いたフィルタを作成し、シミュレーションを行う
- 平均値を用いたフィルタと中央値を用いたフィルタの比較を行う

ご清聴ありがとうございました

Counterのコード1

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

library work;
use work.cmc1.all;

entity counter0 is
    port(
        --clock
        clk : std_logic;

        --input
        input : in std_logic_vector(39 downto 0);

        --output
        output : out std_logic_vector(5 downto 0)

        );
end counter0;

architecture rtl of counter0 is

    --signal
    signal counter1 : counter_1(9 downto 0);
    signal counter2 : counter_2(4 downto 0);
    signal counter3 : counter_3(2 downto 0);
    signal counter4 : unsigned(5 downto 0);
```

```
begin
    process(clk) is
    begin
        if rising_edge(clk) then

            --The first step counter
            counter1(0) <= ("00" & input(0)) + ("00" & input(1)) + ("00" & input(2)) + ("00" & input(3));
            counter1(1) <= ("00" & input(4)) + ("00" & input(5)) + ("00" & input(6)) + ("00" & input(7));
            counter1(2) <= ("00" & input(8)) + ("00" & input(9)) + ("00" & input(10)) + ("00" & input(11));
            counter1(3) <= ("00" & input(12)) + ("00" & input(13)) + ("00" & input(14)) + ("00" & input(15));
            counter1(4) <= ("00" & input(16)) + ("00" & input(17)) + ("00" & input(18)) + ("00" & input(19));
            counter1(5) <= ("00" & input(20)) + ("00" & input(21)) + ("00" & input(22)) + ("00" & input(23));
            counter1(6) <= ("00" & input(24)) + ("00" & input(25)) + ("00" & input(26)) + ("00" & input(27));
            counter1(7) <= ("00" & input(28)) + ("00" & input(29)) + ("00" & input(30)) + ("00" & input(31));
            counter1(8) <= ("00" & input(32)) + ("00" & input(33)) + ("00" & input(34)) + ("00" & input(35));
            counter1(9) <= ("00" & input(36)) + ("00" & input(37)) + ("00" & input(38)) + ("00" & input(39));

            --The second step counter
            counter2(0) <= ('0' & counter1(0)) + ('0' & counter1(1));
            counter2(1) <= ('0' & counter1(2)) + ('0' & counter1(3));
            counter2(2) <= ('0' & counter1(4)) + ('0' & counter1(5));
            counter2(3) <= ('0' & counter1(6)) + ('0' & counter1(7));
            counter2(4) <= ('0' & counter1(8)) + ('0' & counter1(9));

            --The third step counter
            counter3(0) <= ('0' & counter2(0)) + ('0' & counter2(1)) + ('0' & counter2(2));
            counter3(1) <= ('0' & counter2(3)) + ('0' & counter2(4));

            --The fourth step counter
            counter4 <= ('0' & counter3(0)) + ('0' & counter3(1));

            --end counter
            output <= std_logic_vector(counter4);
        end if;
    end process;
end rtl;
```

Counterのコード2

```
1   library ieee;
2   use ieee.std_logic_1164.all;
3   use ieee.numeric_std.all;
4
5   package cmc1 is
6
7      type counter_1 is array (natural range <>) of unsigned(2 downto 0);
8      type counter_2 is array (natural range <>) of unsigned(3 downto 0);
9      type counter_3 is array (natural range <>) of unsigned(4 downto 0);
10     type counter_4 is array (natural range <>) of unsigned(5 downto 0);
11
12  end cmc1;
```

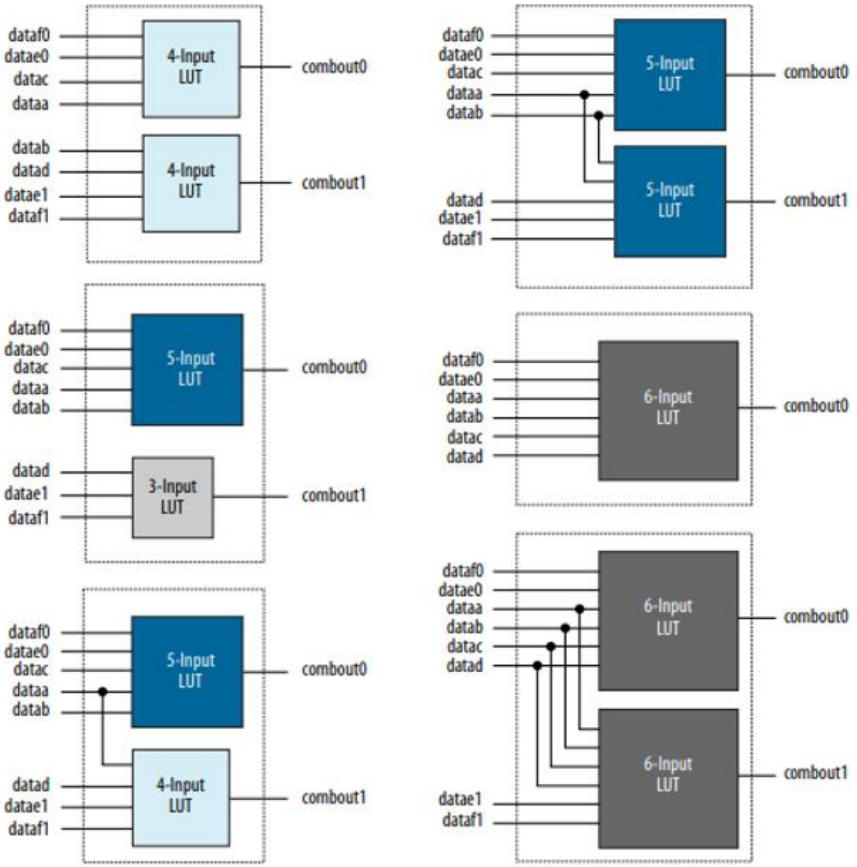
Counterのシミュレーションコード

```
1   library ieee;
2   use ieee.std_logic_1164.all;
3   use ieee.numeric_std.all;
4
5   library work;
6   use work.cmc1.all;
7
8   entity counter0_sim is
9   end counter0_sim;
10
11  architecture SIM of counter0_sim is
12     component counter0
13     port(clk : in std_logic;
14        input : in std_logic_vector(39 downto 0);
15        output : out std_logic_vector(5 downto 0));
16     end component;
17
18  signal clk : std_logic;
19  signal I : std_logic_vector(39 downto 0);
20  signal O : std_logic_vector(5 downto 0);
21
22  begin
23     U01 : counter0 port map(clk => clk, input => I, output => O);
24
25     process begin
26        clk <= '0' ; wait for 5 ns;
27        clk <= '1' ; wait for 5 ns;
28     end process;
29
30     process begin
31        --I <= "0000000000000000000000000000000000000000"; wait for 5 ns;
32        --I <= "0000000000000010000001000000000000000000"; wait for 5 ns;
33        I <= "1111111111111111111111111111111111111111"; wait for 100 ns;
34
35        assert false;
36        report "good" severity Failure;
37     end process;
38  end SIM;
```

Arria10 ALM

Figure 8. ALM in Normal Mode

Combinations of functions with fewer inputs than those shown are also supported. For example, combinations of functions with the following number of inputs are supported: four and three, three and three, three and two, and five and two.



https://www.intel.co.jp/content/dam/altera-www/global/ja_JP/pdfs/literature/hb/arria-10/a10_handbook_j.pdf