LFSR - Linear Feedback Shift Register - генератор псевдорандомных чисел
Pasted image 20250306085859%20darkmode.png
^ *справа написано "степени непривод[имого полинома]"*
Суть - регистр перебирает значения в пределах 2^n сдвигами, при этом возвращая значения всех/некоторых регистров обратно на вход
Аналог простых чисел в полиномах - неприводимые полиномы - которые невозможно разложить на произведение других двух полиномов, можно юзать в этой схеме псевдорандома
Пример:
Полином $x^4 + x + 1$ - неприводимый, $\rightarrow$ q[3] ^ q[0] (смотрим на степени x и минусуем 1 ибо индекс)
При начале со всеми единицами схема переберёт 15 значений, не повторяясь, и вернётся в 1111 // Можешь сам проверить если делать нехуй

```
module lfsr(
    input clk, rstn;
    output reg[167:0] q
);
reg [167:0] next_q;

\\ *long ass tangent about how we'll be the best and we know everything on hardware and software levels and shit*

always @(posedge clk) begin
    if (!rstn) begin
        q <= {168{1'b1}}; // Забить 168 единиц
    end else begin
        next_q = q << 1;
        next_q[0] = q[167] ^ q[165] ^ q[152] ^ q[150] // Ибо такой полином
        q <= next_q
    end
end
endmodule
```

Вспоминаем схемоту 5 сем получается:

- "=" - блокирующее присваивание - означает, что присваивание должно быть произведено незамедлительно, как только обнаружено. Может заместить другое присваивание, с которым может конфликтовать
- "<=" - неблокирующее присваивание - означает, что присваивание произойдёт "как будто триггер принял значение на вход"
  Как подать индексы из полинома? Ибо проблема - значения нужны константные при синтезе

```
module lfsr #(parameter n=4)
localparam integer poly[767:0][3:0] = '{'{1,2,3,4}, '{5,6,7,8}, ... ,'{167, 165}} // Возможно только в system verilog, называ

localparam [767:0][3:0][31:0] poly = ...{1,2,3,4, ...} // Ебейше огромная шина длиной ~400'000 бит
define o(n, idx) poly[4 * 32 * n + idx * 32 +:32]
```

Q: Что сука такое "+:"???
A: Запись по своей сути эквивалентна "+=" из C++. Можно задать начало и сказать синтезатору "вот с этого значения + 32"

Идея конвейера и как его описать

Идея - вместо одного сумматора реализовать тот же сумматор в несколько стадий ака как конвейер
Pasted image 20250306094701%20darkmode.png
Условно S = 4 - количество стадий, W=128 - количество бит на входах
Разбиваем всю сумму на шаги, на каждом из которых будем досуммировать
Например, если входных бит 128, то можно конвейерно "доплюсовывать" по 32 бита 4 стадиями. В первом шаге - первые 32 бита A + B, на втором - вторые 32 бита A + B и плюсануть к результату и т.д.

```
module pipelined_adder #(parameter w = 128, s = 4)(
    input clk, rstn, cin, valid_op1, valid_op2;
    input [w-1:0] op1, op2;
    output reg [w-1:0] res;
    output reg valid
);
localparam [s * 32 - 1:0] stage_width =
    {32'd34, 32'd32, 32'd32, 32'd30} // Размерность стадий, сумма 128
define wth(stage) stage_widths[32 * stage +:32]
```