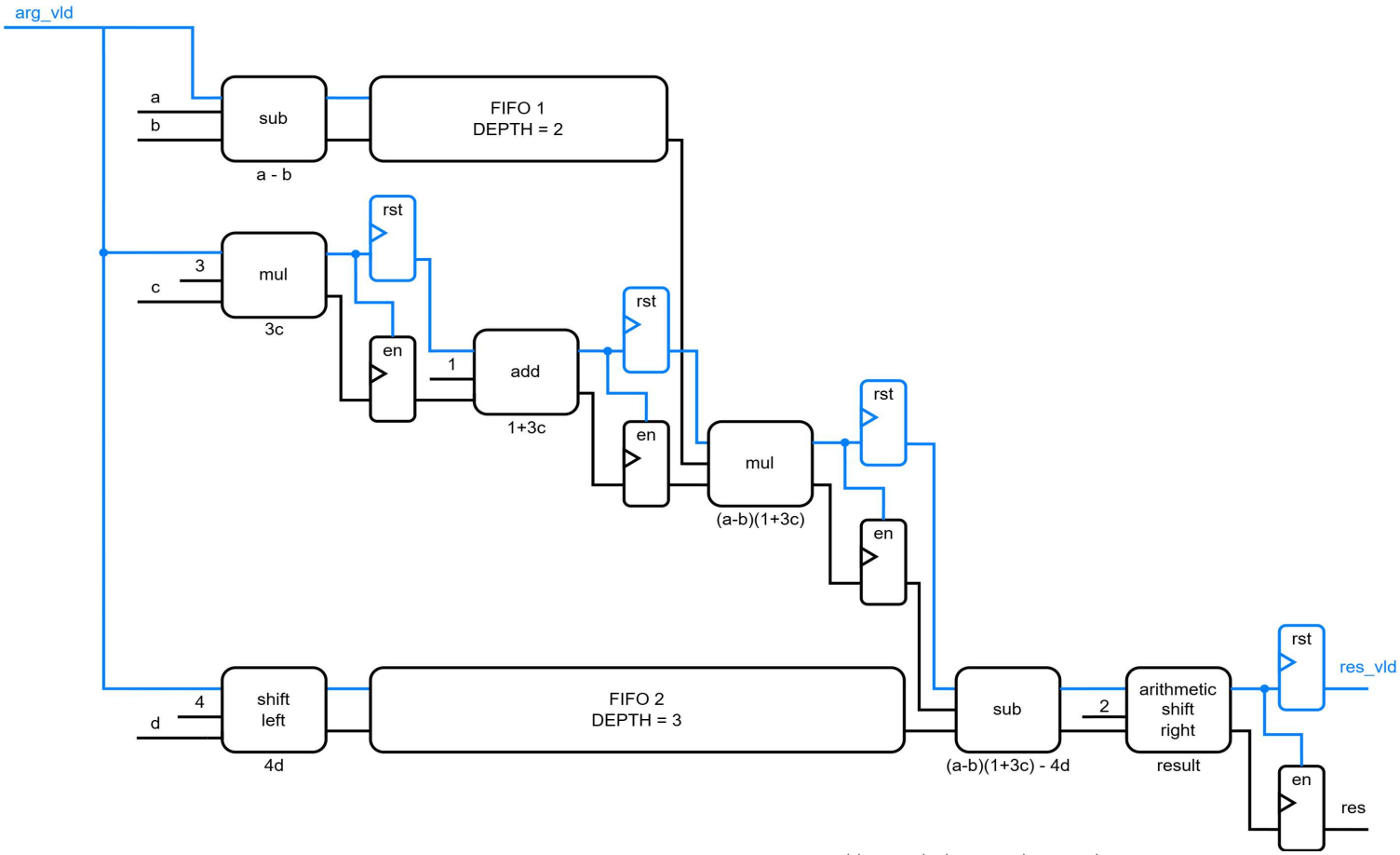

Цифровая схема, реализующая вычисление выражения $Q=((a-b)\cdot(1+3c)-4d)/2$,

где параметры a, b, c, d - целые числа со знаком, разрядность определяется параметром.

Описание архитектурных решений:

Параметр, определяющий ширину входных и выходного сигнала — WIDTH. Так же параметризован и набор ширин промежуточных значений, по умолчанию у всех входных и выходных сигналов ширина равна WIDTH.

Для увеличения максимально возможной частоты тактового сигнала и увеличения пропускной способности модуль конвейеризован.

Вместе с данными по конвейеру идут сигналы валидности, которые служат сигналами разрешения записи для регистров данных, что позволяет добиться экономии энергопотребления и уменьшения тепловыделения, однако это зависит от предполагаемого отношения числа поданных на вход валидных значений к количеству тактов. В предельном случае, когда 100% подаваемых на вход значений валидны эффект получается противоположный, однако нам все равно требуется синал валидности для выходного сигнала.

Очереди использованы для синхронизации данных, на их месте могли бы быть сдвиговые регистры, но очереди предпочтительнее из-за того, что позволяют не менять состояние всех внутренних регистров. Их использование тем более выгодно, чем более глубокий тракт данных предполагается(параметр DEPTH).

В почти каждом промежуточном модуле реализован флаг переполнения, но не использован, для верификации они были полезны, однако нужно будет их использовать, можно будет подключить сигналы к сдвиговому регистру и вместе с флагом res_vld выдавать флаг overflow.

Умножения на степени двойки реализовано как сдвиг влево, деление аналогично — арифметический сдвиг вправо.

Последнее вычитание и сдвиг не разделены регистром, т. к. критический путь, ограничивающий тактовую частоту, будет между входами и выходами модулей умножения(а если было бы деление не на степень двойки, то в модуле деления). Таким образом не ставя регистр между ними мы не увеличим критический путь и уменьшим латентность модуля.

Простейший способ защиты от переполнения — увеличение разрядности. Классический — изменение порядка вычислений. В таком случае нужно будет определять где случилось(или случится) переполнение(флаги уже есть в модулях) и в зависимости от этого вычислять оставшееся выражение:

$$Q=\frac{(a-b)}{2}\cdot(1+3c)-2d, \text{ или}$$

$$Q=(a-b)\frac{1+3c}{2}-2d$$

Для верификации я использовал IcarusVerilog (https://bleyer.org/icarus/), для генерации тестовых векторов — Python. Для простмотра waweform - Surfer (VScode extension или online).
Команды для запуска (win):

```
python .\gen_test_vec.py

iverilog  -g2005-sv *.sv 2>&1 > log.txt
vvp a.out 2>&1 >> log.txt
```

В файле test_vectors.txt каждая строка представляет собой набор входных данных за которыми следует вычисленное в python значение выражения:

a b c d res_expected