УДК 004.932

ВАРИАЦИОННЫЙ МЕТОД ВЫЧИСЛЕНИЯ ОПТИЧЕСКОГО ПОТОКА В СИСТЕМЕ-НА-КРИСТАЛЛЕ

Беляков П.В., *аспирант кафедры ЭВМ Рязанского государственного радиотехнического университета, belyakov.p.v@evm.rsreu.ru;*
Никифоров М.Б., *к.т.н., доцент кафедры ЭВМ Рязанского государственного радиотехнического университета, nikiforov.m.b@evm.rsreu.ru.*

SYSTEM-ON-CHIP VARIATIONAL OPTICAL FLOW COMPUTATION

Belyakov P.V., Nikiforov M.B.

Variational methods of the optical flow computation are widely known and common used methods for motion detection, object tracking in various areas of image processing, 3D reconstruction and autonomous robot navigation. The variational nonlinear method of the optical flow computation is the most accurate, but at the same time the most computationally intensive. Its implementation on a system-on-chip (SoC) is a trade-off between the design difficulty and high performance hardware implementation. The article is devoted to the specific SoC-based solution of the variational optical flow computation method which has been implemented using Verilog hardware description language. The proposed solution is applicable for a dense variational nonlinear optical flow real time computation and can act as a hardware SoC accelerator for the optical flow computation in various image processing tasks.

Key words: optical flow, variational methods, FPGA, system-on-chip.

Ключевые слова: оптический поток, вариационный метод, ПЛИС, система-на-кристалле.

Вариационные методы вычисления оптического потока - широко известные и общепринятые в различных областях обработки изображений для обнаружения движения, сопровождения объектов, 3D-реконструкции и автономной навигации роботов. Нелинейная модель вычисления оптического потока вариационным методом является наиболее точной, но вместе с тем наиболее вычислительно сложной. Ее реализация на системе-на-кристалле (SoC – System on Chip) является компромиссом между трудностью проектирования и высокой производительностью аппаратной реализации. Статья посвящена реализации на SoC вариационного метода вычисления оптического потока с использованием языка аппаратного дизайна Verilog. Предлагаемое решение способно производить вычисление плотного нелинейного оптического потока в реальном масштабе времени и может выступать в качестве аппаратного ускорителя на SoC вычисления оптического потока в различных задачах обработки изображений.

Введение

Оптический поток – это изображение видимого движения, представляющее собой сдвиг каждой точки между двумя изображениями [1]. Это векторное поле, где каждый вектор имеет две компоненты, чтобы показать смещение между точками при их перемещении от первого изображения ко второму [2]. По сути, оптический поток представляет собой поле скоростей (т.к. сдвиг эквивалентен мгновенной скорости) и для каждой точки первого изображения $\mathrm{I}_1(\mathrm{x},\mathrm{y})$ находится такой сдвиг $(\mathrm{dx},\mathrm{dy})$, чтобы исходной точке соответствовала точка на втором изображении $I_2(x+dx, y+dy)$ [3].

Метод, который вычисляет оптический поток для всех пикселей в изображении, называется плотным, тогда как методы, обрабатывающие некоторые отдельные пиксели, называются разреженные [4].

Предлагаемая работа основана на вариационном методе [5], который относится к плотным методам и основывается на нескольких основных предположениях для вычисления оптического потока: предположении о постоянстве значения яркости изображения, предположении о постоянстве значения градиента яркости и предположении гладкости искомого вектора оптического потока. Минимизация функционала, содержащего указанные предположения, основана на решении уравнений в частных производных численными методами.

Использование минимизации функционала с регуляризацией является одним из эффективных методов вычисления оптического потока. Этот подход основан на минимизации заданного функционала путем решения системы линейных уравнений. Для решения такой системы уравнений можно использовать один из итерационных численных методов решения системы линейных алгебраических уравнений, например, метод ослабления Гаусса-Зейделя (метод релаксации SOR – Successive-Over Relaxation), обладающий повышенной сходимостью.

Дополнительным инструментом эффективного вычисления оптического потока является применение подхода масштабирования изображений, когда поиск решения начинается на самом грубом уровне пирамиды сглаженных масштабированных изображений и проходит через всю последовательность пирамиды изображений разного масштаба для поиска глобального минимума функционала.

В течение долгого времени главным инструментом для вычисления оптического потока в реальном масштабе времени из-за его сложности оставались графические процессоры. Относительно недавно и SoC стали подходить для выполнения аналогичных задач. Хорошо известно, что SoC поддерживают разный уровень па-

раллелизма, что, в свою очередь, приводит к повышению производительности с точки зрения вычислительной эффективности [6]. Целью этой работы является разработка соответствующей аппаратной архитектуры для вычисления оптического потока. Для наиболее эффективной аппаратной реализации использовался язык аппаратного дизайна Verilog. Таким образом, в этой статье представлена высокопроизводительная оптимизированная аппаратная архитектура SoC для вычисления оптического потока в реальном времени.

Работа показывает возможность реализации алгоритма вариационного плотного оптического потока на ПЛИС на языке описания аппаратуры Verilog для создания параллельной высокопроизводительной конвейерной архитектуры, обеспечивающей максимальную производительность [7]. Аппаратное ядро оптического потока реализовано с помощью Xilinx Vivado IDE. Система разработки и отладки вариационного оптического потока, состоящая из программного приложения и аппаратной части, реализована на платформе Xilinx Zynq-7000 SoC отладочной платы ZC706.

Вариационный метод вычисления оптического потока

Одной из наиболее точной, но вместе с тем наиболее вычислительно сложно, является постановка задачи вычисления оптического потока в вариационной формулировке и ее решение через минимизацию функционала, описываемого выражением

$$E(u,v)=\int(\psi((I(\boldsymbol{x}+\boldsymbol{w})-I(\boldsymbol{x}))^2+\gamma\,|\nabla I(\boldsymbol{x}+\boldsymbol{w})-\\-\nabla I(\boldsymbol{x})|^2+\alpha\psi(|\nabla u|^2+|\nabla u|^2))\boldsymbol{dx}. \quad (1)$$

Здесь I – изображение, $\boldsymbol{x}=(x,y,t)^T$ – координаты пикселя на изображении, $\boldsymbol{w}=(u,v,1)^T$ – искомый вектор смещения между пикселями двух изображений в момент времени t и момент времени $t+1$. $\nabla\coloneqq(\partial_x,\partial_y)^T$ пространственный градиент, γ – коэффициент между яркостью и градиентом яркости, $\alpha>0$ параметр регуляризации, описывает то, насколько важно требование гладкости полученного вектора смещений. Функция регуляризации $\psi(s^2)=\sqrt{s^2+\varepsilon^2}$ определяет устойчивость минимизации функционала к изменениям яркости изображения и гладкости поля с параметром $\varepsilon=0,001$.

Уравнения Эйлера-Лагранжа

Минимизация функционала (1) сводится к решению уравнений Эйлера-Лагранжа:

$$\psi'(I_z^2+\gamma(I_{xz}^2+I_{yz}^2))\cdot(I_xI_z+\gamma(I_{xx}I_{xz}+I_{xy}I_{yz}))-\\-\alpha\,div(\psi'(|\nabla u|^2+|\nabla v|^2)\nabla u)=0,\\\psi'(I_z^2+\gamma(I_{xz}^2+I_{yz}^2))\times(I_yI_z+\gamma(I_{yy}I_{yz}+I_{xy}I_{xz}))-\\-\alpha\,div(\psi'(|\nabla u|^2+|\nabla v|^2)\nabla v)=0. \quad (2)$$

Здесь использование индекса z вместо t означает аппроксимацию операции дифференцирования по времени разностью.

$$I_x\coloneqq\partial_xI(\boldsymbol{x}+\boldsymbol{w}),$$
$$I_y\coloneqq\partial_yI(\boldsymbol{x}+\boldsymbol{w}),$$
$$I_z\coloneqq I(\boldsymbol{x}+\boldsymbol{w})-I(\boldsymbol{x}),$$
$$I_{xx}\coloneqq\partial_{xx}I(\boldsymbol{x}+\boldsymbol{w}),$$
$$I_{xy}\coloneqq\partial_{xy}I(\boldsymbol{x}+\boldsymbol{w}),$$
$$I_{yy}\coloneqq\partial_{yy}I(\boldsymbol{x}+\boldsymbol{w}),$$
$$I_{xz}\coloneqq\partial_xI(\boldsymbol{x}+\boldsymbol{w})-\partial_xI(\boldsymbol{x}),$$
$$I_{yz}\coloneqq\partial_yI(\boldsymbol{x}+\boldsymbol{w})-\partial_yI(\boldsymbol{x}),$$
$$|\nabla u|^2=u_x^2+u_y^2,\quad|\nabla v|^2=v_x^2+v_y^2,$$
$$u_x=\frac{\partial u}{\partial x},\ u_y=\frac{\partial u}{\partial y},\ v_x=\frac{\partial v}{\partial x},\ v_y=\frac{\partial v}{\partial y}.$$

Минимизация функционала

Для устранения нелинейности в выражении (2) для $\boldsymbol{w}=(u,v,1)$ первоначально применяется метод последовательных приближений [8]. Пусть k означает итерацию последовательно приближения, тогда I_*^k обозначает первые и вторые производные в выражении (2) применительно к неизвестному вектору $\boldsymbol{w}^k$ на k -ой итерации вместо $\boldsymbol{w}$. Тогда $\boldsymbol{w}^{k+1}$ будет очередным приближением решения уравнения

$$\psi'((I_z^{k+1})^2+\gamma((I_{xz}^{k+1})^2+(I_{yz}^{k+1})^2))\times\\\times(I_x^kI_z^{k+1}+\gamma(I_{xx}^kI_{xz}^{k+1}+I_{xy}^kI_{yz}^{k+1}))-\\-\alpha\,div(\psi'(|\nabla u^{k+1}|^2+|\nabla v^{k+1}|^2)\nabla u^{k+1})=0, \quad (3)$$

Второе выражение в уравнении Эйлера-Лагранжа может быть записано аналогичным образом.

Далее, для устранения нелинейности в выражениях I_*^{k+1}, применяется разложение в ряд Тейлора:

$$I_z^{k+1}\approx I_z^k+I_x^kdu^k+I_y^kdv^k,$$
$$I_{xz}^{k+1}\approx I_{xz}^k+I_{xx}^kdu^k+I_{xy}^kdv^k,$$
$$I_{yz}^{k+1}\approx I_{xz}^k+I_{xy}^kdu^k+I_{yy}^kdv^k,$$

где $u^{k+1}=u^k+du^k$ и $v^{k+1}=v^k+dv^k$. Искомые $u^{(k+1)},v^{(k+1)}$ разделяются на решение с предыдущей итерации u^k,v^k и неизвестное приращение вектора du^k,dv^k.

Обозначим

$$(\psi_D')^k=\psi'((I_z^k+I_x^kdu^k+I_y^kdv^k)^2+\\+\gamma((I_{xz}^k+I_{xx}^kdu^k+I_{xy}^kdv^k)^2+\\+(I_{yz}^k+I_{xy}^kdu^k+I_{yy}^kdv^k)^2)),$$
$$(\psi_s')^k=\psi'(|\nabla(u^k+du^k)|^2+|\nabla(v^k+dv^k)|^2),$$

где ψ_D' – устойчивость к изменению яркости, and ψ_s' – требование гладкости вектора, тогда выражение (3) может быть записано как

$$(\psi_D')^k\cdot(I_x^k(I_z^k+I_x^kdu^k+I_y^kdv^k))+\\+\gamma(\psi_D')^k(I_{xx}^k(I_{xz}^k+I_{xx}^kdu^k+I_{xy}^kdv^k)+\\+I_{xy}^k(I_{yz}^k+I_{xy}^kdu^k+I_{yy}^kdv^k))-\\-\alpha\,div((\psi_s')^{k,l}\nabla(u^k+du^k))=0. \quad (4)$$

Аналогично и для второго выражения уравнения Эйлера-Лагранжа.

Для устранения остающейся нелинейности в приращениях du^k, dv^k и в выражениях для ψ', применяется второй внутренний цикл последовательных приближений с индексом l. Окончательно выражение для вычисления $du^{k,l+1}$ записывается

$$\begin{aligned}&(\psi_D')^{k,l}\cdot(I_y^k(I_z^k+I_x^k du^{k,l+1}+I_y^k dv^{k,l+1})+\\&+\gamma I_{xy}^k(I_{xz}^k+I_{xx}^k du^{k,l+1}+I_{xy}^k dv^{k,l+1})+\\&+\gamma I_{xy}^k(I_{yz}^k+I_{xy}^k du^{k,l+1}+I_{yy}^k dv^{k,l+1}))-\\&-\alpha\, div((\psi_s')^{k,l}\nabla(u^k+du^{k,l+1}))=0.\end{aligned}\quad(5)$$

Аналогично для $du^{k,l+1}$.

Интерполяция изображения $I(\boldsymbol{x}+\boldsymbol{w}^k)$ может быть осуществлена билинейной интерполяцией .

Численная аппроксимация

Численные методы оказывают определяющие влияние как на точность вычислений, так и на производительность [9].

Аппроксимация дифференциальных уравнений в частных производных (5) операцией конечной разности для искомых приращений вектора смещения du^k на -ой итерации последовательного приближения записывается в виде

$$\begin{aligned}&(\psi_D')^{k,l}\cdot(I_x^k(I_z^k+I_x^k du^{k,l+1}+I_y^k dv^{k,l+1})+\\&+\gamma I_{xx}^k(I_{xz}^k+I_{xx}^k du^{k,l+1}+I_{xy}^k dv^{k,l+1})+\\&+\gamma I_{xy}^k(I_{yz}^k+I_{xy}^k du^{k,l+1}+I_{yy}^k dv^{k,l+1}))-\\&-\alpha\sum_{n=x,y}\sum_{j\in N_n(i)}\frac{(\psi_{si}')^{k,l}+(\psi_{si}')^{k,l}}{2}\cdot\frac{u_j^k+du_j^{k,l+1}-u_i^k-du_i^{k,l+1}}{(h_n^k)^2}=0,\end{aligned}$$

для сетки $h_x^k\times h_y^k$. Аналогичным образом и для приращения dv^k.

Таким образом линеаризация нелинейных уравнений приводит к системе линейных алгебраических уравнений (СЛАУ), которые могут быть численно решены итерационным методом ослабления Гаусса-Зейделя (методом релаксации SOR – ***S**uccessive-**O**ver **R**elaxation*), обладающим повышенной сходимостью.

$$Ax=b,$$
$$A=L+D+U,$$
$$Dx^{m+1}=(1-\omega)Dx^m-\omega(Lx^{m+1}+Ux^m)+\omega b.$$

L – нижняя треугольная матрица, U – верхняя треугольная матрица, D – диагональная матрица и ω – коэффициент релаксации.

В окончательном виде выражение для итерационного решения системы алгебраических уравнений для приращения du^k запишется:

$$\begin{aligned}&du_i^{k,l,m+1}=(1-\omega)du_i^{k,l,m}+\omega\frac{F^-+F^+-F}{R}-\\&-\omega\frac{\frac{(\psi_D')_i^{k,l}}{\alpha}(L+\gamma G)}{R},\end{aligned}\quad(6)$$

где $F^-=\sum_{j\in N^-(i)}(\psi_s')_{i\sim j}^{k,l}(u_j^k+du_j^{k,l,m+1}),$

$$F^+=\sum_{j\in N^+(i)}(\psi_s')_{i\sim j}^{k,l}(u_j^k+du_j^{k,l,m}),$$

$$F=\sum_{j\in N^-(i)\cup N^+(i)}(\psi_s')_{i\sim j}^{k,l}u_i^k,$$

$$R=\sum_{j\in N^-(u)\cup N^+(u)}(\psi_s')_{i\sim j}^{k,l}+$$

$$f=\frac{(\psi_D')_i^{k,l}}{\alpha}(((I_x)_i^k)^2+((I_{xy})_i^k)^2+((I_{xx})_i^k)^2),$$

$$+\frac{(\psi_D')_i^{k,l}}{\alpha}(((I_x)_i^k)^2+((I_{xy})_i^k)^2+((I_{xx})_i^k)^2),$$

$$L=(I_x)_i^k((I_y)_i^k dv_i^{k,l,m}+(I_z)_i^k),$$

$$G=(I_{xx})_i^k((I_{xy})_i^k dv_i^{k,l,m}+(I_{xz})_i^k)$$

$$+(I_{xy})_i^k((I_{yy})_i^k dv_i^{k,l,m}+(I_{yz})_i^k).$$

Аналогично для dv^k.

Здесь m – индекс итерации SOR, $N^-:=\{j\in N_n(i)\mid j<i\}$ означает пикселы, которые только будут обработаны в текущей итерации вычисления SOR, $N^+:=\{j\in N_n(i)\mid j>i\}$ означает пикселы, которые были вычислены на предыдущей итерации. $(\psi_s')_{i\sim j}^{k,l}$ коэффициент гладкости вектора между пикселями i и j.

Аппаратная реализация

Декомпозиция задачи вычисления оптического потока на программную (SW-software) и аппаратную (HW-hardware) части является естественным для реализации в системе-на-кристалле, содержащую процессорную систему (ARM-процессор) и перепрограммируемую логику (ПЛИС), что вместе образует гибкую вычислительную платформу. На рис. 1 показано распределение задач между программными (SW) и аппаратными (HW) компонентами системы вычисления оптического потока.

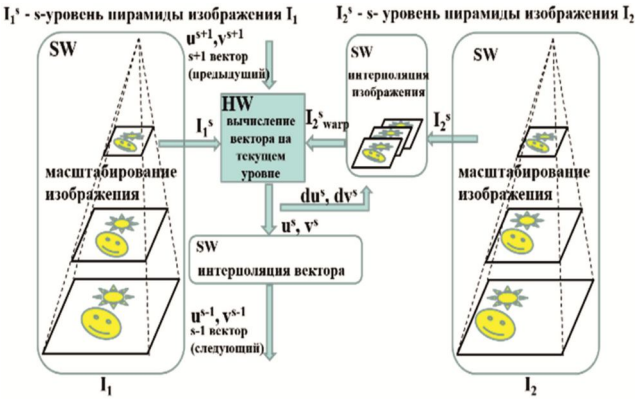

Рис. 1. Схема разбиения SoC на программную и аппаратную подсистемы

Основная часть вычисления вариационного оптического потока реализована на ПЛИС как аппаратное ядро для вычисления вектора [10, 11] на текущем уровне пирамиды изображений. Построение пирамид изображений, интерполяция изображения и вектора возлагается на дополнительные программные модули (на ARM-процессор) из-за дефицита перепрограммируемых логических ресурсов ПЛИС.

Псевдокода обобщенного алгоритма вычисления вариационного оптического потока:

```
for s = nScales − 1, s > 0, s = s − 1 do
      // масштабирование изображений
         Pyramid_Images = I^s_{1,2} * scaleFactor;
      while k ≤ outer_num_Iterations //итерации
по интерполяции изображения
           //вычисление производных для текущего
уровня пирамиды изображения
          Ix, Ixx, Iyy, Ixy, Iz, Izx, Izy // для
I^s_1 и I^s_2
          while l ≤ inner_num_Iterations
            foreach col, row in Images [0, nRows −
1] × [0, nCols − 1] do
            // вычисление градиента вектора оп-
тического потока
                u_x, u_y, v_x, v_y
            // функции штрафов для яркости изоб-
ражений и гладкости потока
                ψ_data, ψ_smoth
            // формирование тензоров движения
                J(col, row) = ∇3I1,2∇3I1,2^T(col, row);
            // формирование тензоров потока
                dx2, dxy, dtdx, dy2, dtdy
            // Successive Over-Relaxation
       While m ≤ sor_num_Iterations AND error de-
crease ≥ MIN ERROR do
              relax system for  du, dv
          // интерполирование I2 в соответствии с
полученными смещениями потока (du, dv).
            I2warp = biline-
ar_interpolation_warp(I2, du, dv);
            // обновление значения вектора потока
              u+du,v+dv
      // интерполирование вектор потока на следу-
ющий уровень пирамиды изображений
        u^{s−1},v^{s−1} = bilinear_interpolation(u^s,v^s,
scaleFactor)
```

Разработанная аппаратная реализация алгоритма вычисления вариационного оптического потока организована как конвейерная схема с различными этапами обработки [12], которые изображены на рис. 2.

• На этапе построения разномасштабной пирамиды изображений выполняется построение пирамид исходных изображений с одновременным сглаживанием масштабируемых изображений для подавления шума.

• На этапе вычисление первых и вторых производных исходных изображений происходит вычисление пространственных производных изображений (I_x, I_y, I_{xx}, I_{xy}, I_x, I_{yy}, I_{xz}, I_{yz}) как компонент функционала оптического потока.

• На этапе вычисление производных вектора оптического потока происходит вычисление скорости изменения (градиента) вектора оптического потока по направлениям (u_x, u_y, v_x, v_y).

• Вычисление производных функций отклонения от постоянных значений ψ' для яркостной и векторной составляющей функционала оптического потока.

• На этапах построения тензоров изменения яркости и градиента (**J11**, **J12**, **J13**, **J22**, **J23**) и построения тензоров движения оптического потока (**dx2**, **dxy**, **dy2**, **dtdx**, **dtdy**) происходит формирование матриц линейного уравнения (5) вида Ax = b, для вычисления неизвестного смещения оптического потока.

• Этап итерационного решения СЛАУ (5) методом последовательного ослабления Гаусса- Зейделя (метод релаксации SOR).

Таким образом нелинейный вариационный метод реализуется на основе внутреннего и внешнего циклов последовательного приближения, в свою очередь содержащих итерации метода SOR, переопределение ψ' и обновление тензоров движения и потока соответственно.

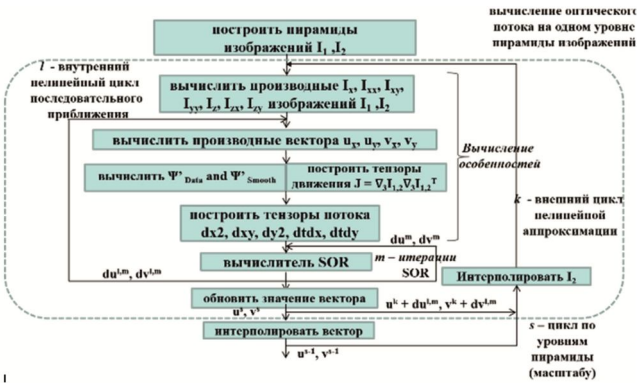

Рис. 2. Схема алгоритма вычисления оптического потока

На рис. 3 изображена разработанная функциональная схема параллельного вычисления производных, определение функций штрафа и построения тензоров оптического потока. Процессы не зависимы по данным и могут быть вычислены независимо и параллельно. Схема отражает возможность конвейерной организации последовательных этапов вычисления производных и построения тензоров, вплоть до этапа итерационного решения СЛАУ SOR.

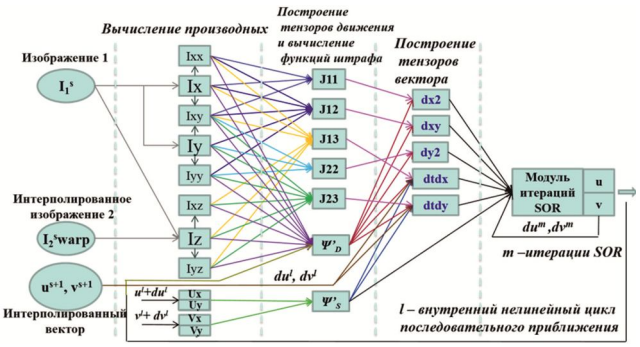

Рис. 3. Функциональная схема аппаратной реализации алгоритма вычисления оптического потока на текущем уровне пирамиды изображений

Исходное выражение для вычисления уравнения (6) непосредственно не подходит для аппаратной реализации из-за зависимости данных от каждого du and dv . Последовательная схема вычисления (SOR, рис. 4 слева) приводит к зависимости по данным, когда для вычисления значения в нечетной позиции, например, 7 (красный элемент в кружке) требуется обновленные в текущей итерации значения двух соседних элементов 2, 6 (красные элементы в кружке) и сохраненные с предыдущей итерации значения двух соседних элементов 8, 12 (черные элементы).

Для возможности параллельной аппаратной реализации модуля решения СЛАУ, применяется представленная на рисунке 4 модификация метода релаксации SOR – метод релаксации «красное-черное» SOR-Red Black (SOR-RB), при котором устраняется зависимость по данным. В схеме параллельного вычислительного процесса (SOR-RB, рис. 4 справа) в текущей итерации для обновления значения в четной позиции, например, 4 (красный элемент в кружке) требуются значения че-

тырех соседних элементов 11, 13, 14, 16 (черные элементы) вычисленных на предыдущей итерации. Т.о. реализуется независимость по данным за счет двухпроходной конвейерной реализации, когда сначала обновляются, например, все красные элементы, а затем все черные [13].

В контексте системы $Ax = b$ задача переупорядочения сводится к вопросу о перестановках строк и столбцов матрицы A.

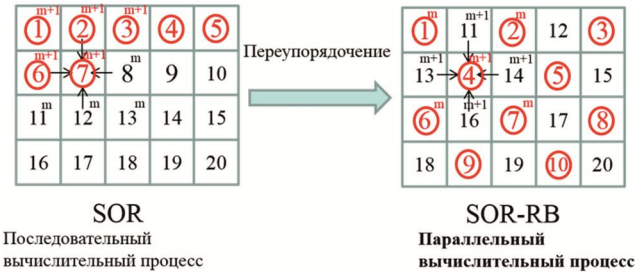

Рис. 4. Избавление от зависимости по данным в методе SOR

Эффективная реализация алгоритмов обработки изображений на ПЛИС требует правильного стиля кодирования на языке аппаратного дизайна HDL. При программной реализации алгоритмов обработки изображений изображения представляют собой 2D-массивы, которые находятся в памяти, при аппаратной же реализации на ПЛИС, изображения представляют собой поток пикселей, который не может быть полностью сохранен во внутренней памяти ПЛИС. Как правило, только несколько линий изображения сохраняются в линейных буферах, которые задерживают входящий поток пропорционально ширине линии. В ПЛИС эти линейные буферы отображаются как FIFO, реализованные как двухпортовые блочные ОЗУ (один блок ОЗУ может быть сконфигурирован для хранения 1024 пикселей по 18 бит на пиксель). Для организации двумерного сверточного фильтра с размером ядра 3x3 требуется два линейных буфера (третья строка – это входящий поток) и скользящее окно размером 3x3. Скользящее окно соотносится с окрестностью входных пикселей, которая взвешивается коэффициентами ядра фильтра и суммируется для генерации каждого выходного пикселя. Кроме того, блочные ОЗУ полностью удовлетворяют требованиям линейного буфера к одновременному доступу для чтения и записи. И наоборот, скользящее окно реализуется как параллельные регистры. Такой подход реализуется как в вычислительном модуле SOR-RB, так и в модуле вычисления признаков изображений, а также для синхронизации потоков данных. На рис. 5 показана функциональная схема модуля SOR-RB.

Аппаратная реализация, основанная на аппаратных примитивах и библиотеках элементов современных ПЛИС, позволяет относительно быстро реализовать математические операции в формате с плавающей точкой. Например выражение $\sqrt{s^2 + \varepsilon^2}$ относительно легко реализуется с помощью функционального генератора операций с плавающей точкой Float Point Arithmetic IP-Core Generator, создающего аппаратное ядро операций с плавающей точкой на базе ядра DSP IP [14] На рис. 6 показана функциональная схема модуля выделения признаков.

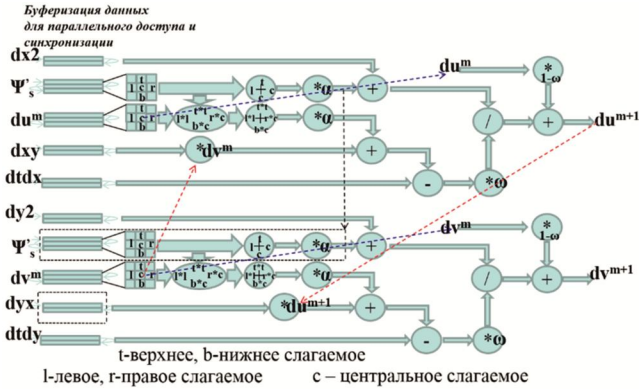

Рис. 5. Функциональная схема аппаратной реализации модуля итерационного решения СЛАУ SOR-RB

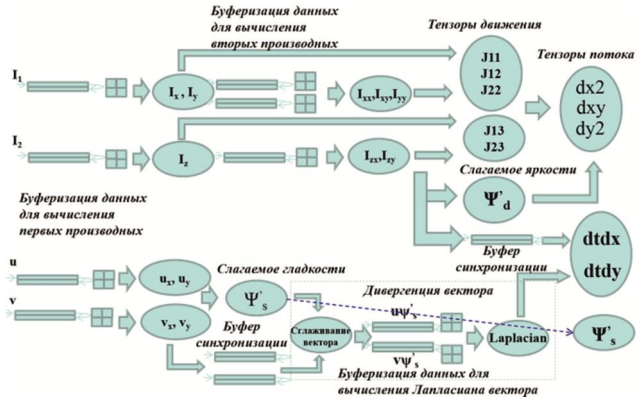

Рис. 6. Функциональная схема аппаратной реализации модуля вычисления признаков

Таким образом, как показано на рис. 7, аппаратные модули для вычисления оптического потока обеспечивают конвейерную и параллельного организацию вычислений на ПЛИС. Основным преимуществом такой конвейерной многоступенчатой организации вычислений является уменьшение количества перемещения данных во внешнюю память и их объема, т.к. не требуется сохранения промежуточных результатов в памяти в связи с тем, что такие данные каждый раз вычисляются «на лету» без сохранения в память. Таким образом, на каждой итерации вычисление тензоров, функций штрафов и производных будет производиться «на лету» и это время вычислений будет намного меньше, чем время доступа к памяти, требующей большую ширину линии данных для чтения и записи промежуточных результатов.

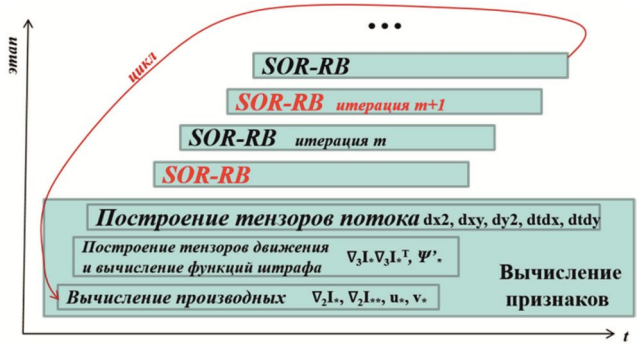

Рис. 7. Функциональная схема аппаратной организации последовательности вычисления оптического потока

Помимо аппаратной оптимизации вычислительного ядра оптического потока требуется организация работы с внешней памятью. Для итерационного вычисления оптического потока на используемой в данной работе архитектуре SoC Zynq [15] предлагается двойная буферизация памяти, которая представлена на рис. 8.

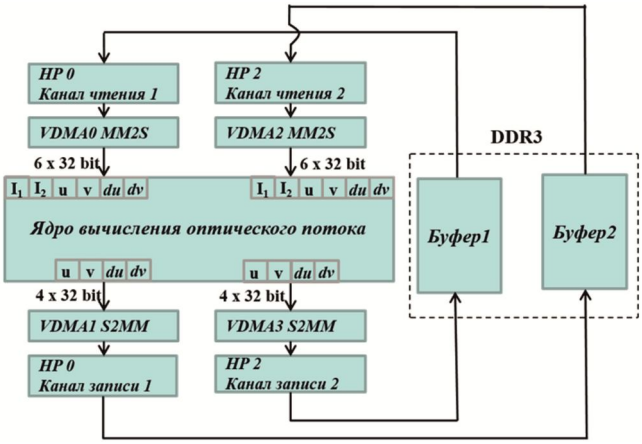

Рис. 8. Организация двойной буферизации памяти

HP0, HP2 – специализированные высокопроизводительные порты доступа к DDR3 памяти системы на кристалле SoC Zynq фирмы Xilinx (HP-high performance port), позволяющие производить обмен данными (входные изображения I_1, I_2 и вектор оптического потока u, v с приращениями du, dv посредством прямого доступа к памяти VDMA (Video Direct Memory Access).

Вся внутренняя и внешняя передача данных основана на шине AXI4 (Advanced eXtensible Interface), что позволяет организовать универсальный обмен между различными модулями.

На рис. 9 показана отладочная платформа Xilinx SoC и структурная схема реализации вычисления оптического потока.

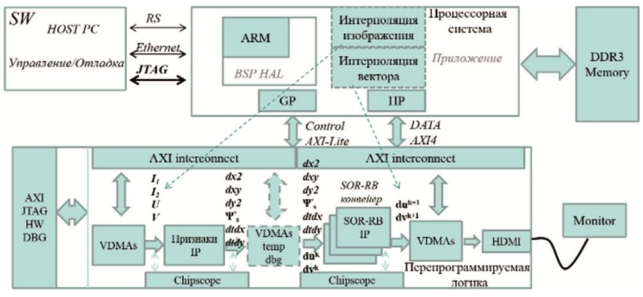

Рис. 9. Прототипирование алгоритма на система-на-кристалле SoC Zynq

Отладочная платформа Xilinx ZYNQ7045 SoC на плате ZC706, сочетает двухъядерный процессор ARM Cortex-A9 с программируемой логикой и делает разработку и отладку (в том числе с помощью тестового приложения Chipscope) более гибкой [16].

Результаты

Потребление ресурсов ПЛИС для аппаратной части вычисления вариационного оптического потока, состоящей из модуля выделения признаков и 8 объединенных в конвейер модулей SOR-RB, показано на рис. 10.

Время выполнения было измерено для одной итерации вычисления приращения вектора du и dv. Оценка времени вычислений проводилась без учета времени исполнения программной части, т.е. без учета времени интерполяции изображения, интерполяции вектора и построения пирамид изображений.

Resource	Utilization	Available	Utilization %
LUT	133664	218600	61.15
LUTRAM	29974	70400	42.58
FF	196131	437200	44.86
BRAM	194	545	35.60
DSP	720	900	80.00
IO	24	362	6.63
BUFG	5	32	15.63
MMCM	1	8	12.50

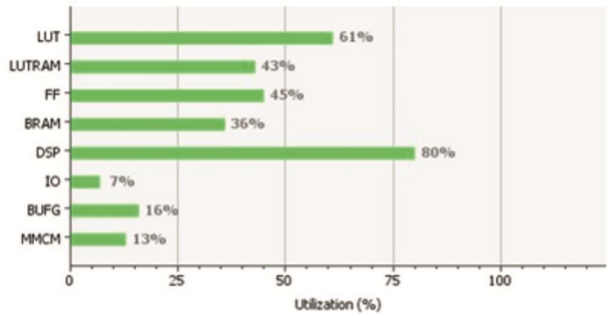

Рис. 10. Потребление ресурсов ПЛИС

Время выполнения было измерено для отладочной платы с пропускной способностью памяти DDR3 533 МГц x 32 бит x 2 ≈ 4,2 ГБ/с и 150 МГц внутренним тактовым сигналом ПЛИС.

Для наибольшего уровня пирамиды изображения – 685x494 пикселей (изображения I_1, I_{2warp}) объем считываемых из DDR3-памяти данных I_1, I_{2warp}, u, v, du^m, dv^m – 6x685x494x32 бит ≈ 8 МБ;

Объем записываемых данных в DDR3-память du^{m+1}, dv^{m+1} – 2x685x494x32 bit ≈ 2 МВ;

Т.о. время вычисления вектора du^{m+1}, dv^{m+1} составляло ≈ 6 мс.

Для наименьшего уровня пирамиды изображения – 33x24 пикселя (изображения I_1, I_{2warp}) объем с считываемых из DDR3-памяти данных I_1, I_{2warp}, u, v, du^m, dv^m - 6x33x24x32 бит ≈ 19 КБ;

Объем записываемых данных в DDR3-память – 2x33x24x32 бит ≈ 6 КВ;

Т.о. время вычисления вектора du^{m+1}, dv^{m+1} составляло ≈ 20 us.

В целом время вычисления оптического потока зависит от количества итераций SOR и уровней пирамиды изображения и может варьироваться в зависимости от задачи, но приведенные оценки свидетельствуют о возможности использования разработанной аппаратной реализации для вычисления оптического потока в реальном времени.

Заключение

Предложенная аппаратная архитектура нелинейного вариационного алгоритма оптического потока была реализована на языке аппаратного дизайна Verilog для обеспечения высокоэффективного параллелизма вычислений и конвейерной обработки. Использование хо-

рошо развитой специализированной библиотеки компонентов (Xilinx) для организации вычислений в формате с плавающей точкой, позволило реализовать все преимущества нелинейного подхода для вычисления вариационного оптического потока в реальном времени.

Экспериментальные результаты показывают, что высокопроизводительная аппаратно-оптимизированная архитектура для высокоэффективного вычисления вариационного оптического потока может быть реализована как ускоритель на SoC. Для выполнения интерполяции изображения и вектора в реальном времени эти программные модули должны быть также реализованы на аппаратном уровне.

Тем не менее предлагаемое решение из-за высокой производительности и точности в сочетании с энергоэффективностью применимо для вычисления смещения изображений и определения глубины сцены для 3D-реконструкции в задачах автономной навигации роботов (алгоритмы SLAM) [17] когда не требуется высоких скоростей перемещения.

Литература

1. B. Lucas and T. Kanade. An iterative image registration technique with an application in stereo vision. In Proc. IEEE Int. Joint Conf., Artificial Intelligence, 1981, pp. 674–679.

2. Елесина С.И., Никифоров М.Б., Логинов А.А., Костяшкин Л.Н. Монография под ред. Л.Н. Костяшкина, М.Б. Никифорова. Совмещение изображений в корреляционно-экстремальных навигационных системах. М.: Радиотехника, 2015. 208 с.

3. B. K. P. Horn and B. G. Schunck. Determining optical flow, Artificial Intelligence, 17:185–203, 1981.

4. Абдухаликов А.А., Беляков П.В., Никифоров М.Б. Реализация на ПЛИС алгоритма поиска ключевых точек на изображении. Международная научно-техническая и научно-методическая конференция «Современные технологии в науке и образовании» СТНО-2016, 2016, с. 103-108.

5. T. Brox, A. Bruhn, N. Papenberg, and J. Weickert. High accuracy optical flow estimation based on a theory for warping. In Proc. European Conf., Computer Vision, volume 4, 2004, pp. 25–36.

6. Обработка изображений в авиационных системах технического зрения / Под ред. Л.Н. Костяшкина, М.Б. Никифорова. М.: ФИЗМАТЛИТ, 2016, с. 28-32

7. D. Ustukov, Y. Muratov, M. Nikiforov, V. Gurov. Implementing one of stereovision algorithms on FPGA. Mediterranean Conference on Embedded Computing, Jun 2016.

8. A. Bruhn and J. Weicker. Towards ultimate motion estimation: combing highest accuracy with real-time performance. In Proc. 10th IEEE Int.Conf., Computer Vision, 2005, pp. 749–755.

9. A. Bruhn, J. Weickert, and C. Schnorr. Lucas/Kanade meets Horn/Schunck: Combining local and global optic flow methods. Int. J. Computer Vision, 2005, 61:211–231.

10. M. Kunz, A. Ostrowski, P. Zipf. An FPGA-optimized architecture of Horn and Schunck optical flow algorithm for real-time applications. Field Programmable Logic and Applications (FPL), 2014 24th International Conference.

11. J.L. Martin, A. Zuloaga, C. Cuadrado, J. Lazaro, and U. Bidarte. Hardware implementation of optical flow constraint equation using fpgas. Computer Vision and Image Understanding, 2005, pp 462–490.

12. Z. Chai, H. Zhou, †Z. Wang and ‡D. Wu Using C to implement high-efficient computation of dense optical flow on FPGA-accelerated heterogeneous platforms. IEEE 14 International Conference on Field-Programmable Technology (FPT), 2014.

13. Ortega, James M. Introduction to Parallel and Vector Solution of Linear Systems, 1988.

14. Xilinx. Zynq-7000 SoC. https://www.xilinx.com/support/documentation/user_guides/ug479_7Series_DSP48E1.pdf.

15. Xilinx. Zynq-7000 SoC. http://www.xilinx.com/products/silicon-devices/soc/zynq-7000/index.htm.

16. Xilinx. Vivado Design Suite. http://www.xilinx.com/products/design-tools/vivado.

17. Ларкин Е.В. Моделирование процесса дистанционного управления роботом. Извести ТулГУ. Технические науки, 2016, Вып. 12. Ч. 4, с. 202-214.