Структурный анализ и синтез графических изображений на экранах современных средств бортовой индикации на плоских жидкокристаллических панелях

Structure analysis and synthesis of graphics images on screens of air indication on flat liquid-crystal panels

П.П. Парамонов, Ю.А. Ильченко, И.О. Жаринов, П.Ю. Тарасов,
ФГУП «Санкт-Петербургское ОКБ «Электроавтоматика»

Рассматривается актуальная задача синтеза графических изображений на экранах современных средств бортовой индикации на плоских жидкокристаллических панелях. Анализируются основные преимущества и недостатки программного и аппаратного способов реализации вычислительных процедур.

1. Общие сведения о графических изображениях на экране многофункциональных индикаторов МФЦИ

При всех последних достижениях микропроцессорной техники в области структурной организации, повышении тактовой частоты процессоров, применения конвейеризации выполнения команд, дальнейшее существенное повышение эффективности электронных систем связано с отказом от программной реализации алгоритмов вычислений и переходом к их аппаратной реализации на базе СБИС. Такой переход обусловлен, с одной стороны, перспективной тенденцией построения сложных вычислительных систем на основе модульного принципа — систем с «открытой» архитектурой, предполагающей наличие центрального вычислителя и целого спектра высокопроизводительных сопроцессоров, контролеров специального назначения. С другой стороны — уровнем современной полупроводниковой техники с ее высокой степенью интеграции и малой себестоимостью, предоставляющей разработчикам СБИС более широкие возможности для аппаратной реализации сложнейшего математического обеспечения.

Это позволит максимально учитывать специфику вычислительных методов и алгоритмов, применяя по возможности распределение этапов разработки, обеспечивая однородность и регулярность вычислений, простоту состава, структуры и внешнего управления, что в свою очередь составляет основу для разработки базовых операций при выполнении многих типовых вычислительных процедур [2].

Анализ структурных фрагментов растровых кадров изображения (см. рис.1) на экранах современных бортовых многофункциональных индикаторов МФЦИ позволяет говорить о существовании, по-видимому, определенного конечного набора базовых графических структур и графоэлементов, комбинация которых на экране составляет изображения в целом.

Основными структурами изображения современных средств индикации являются:

– индикационный кадр — законченное информационное визуальное сообщение, которое в заданный момент времени может быть целиком размещено на экране средства индикации. Кадр состоит из одного или нескольких фрагментов изображения — модулей изображения.

– модуль изображения — связанные между собой элементы изображения, которые характеризуются одинаковым законом перемещения на экране. Модуль может состоять из одного или нескольких элементов изображения. Модуль может быть статическим или динамическим. Статический модуль — модуль, в котором все элементы в процессе изображения не изменяют ни своего положения на экране, ни своего значения. Динамический модуль — модуль, в котором хотя бы один элемент в процессе формирования изображения от одного кадра к другому изменяет свое положение, либо значение.

– элементы изображения — простейшие составляющие изображения, с помощью и на основе которых может строиться любой модуль и кадр. Элементом изображения являются, например, точка, символ (знак алфавита или цифра).

Учитывая матричную структуру ЖК-экрана, любой графический примитив можно представить как упорядоченный набор отдельных «подсвеченных» пикселей. Тогда визуальному отображению, скажем, элементарной линии на экране будет отвечать последовательность повернутых определенным образом кристаллов, закон выбора которых напрямую зависит от алгоритма перебора соответствующих им ячеек видео ОЗУ при, так называемой, росписи (или формировании) изображения в видеопамяти.

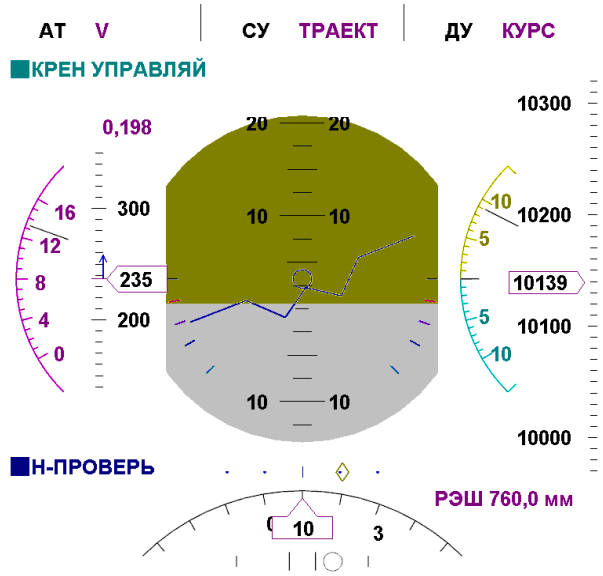


Рис.1. Пример рабочего кадра изображения на экране МФЦИ.

Построению алгоритмов генерации изображения в цифровых электронных устройствах отображения информации и разработке характерных для этого методологических вопросов посвящено немало публикаций как в нашей стране, так и за рубежом. Соответствующая область научного и практического исследования получила название Интерактивной Машинной Графики и сегодня включает в себя не только изучение математических основ синтеза отдельных фрагментов изображения, но и вопросы их аппаратной реализации как на базе современных микропроцессорных платформ, так и на базе специализированных СБИС — так называемых микросхем графических контроллеров.

Графические контроллеры представляют собой «чипы» с высокой степенью интеграции, внутреннее устройство и связи которых подчинены решению определенной задачи — формированию во внешних ячейках памяти (видео ОЗУ) фрагментов изображения в заданной системе координат. Порядок заполнения ячеек памяти видео ОЗУ основан на аппаратной реализации вычислительных процедур при выполнении поступающих в графический контроллер команд «высокого уровня». В частности, известно, что в большинстве языков программирования для того, чтобы «нарисовать», скажем, линию на экране необходимо задать четыре ее координаты (в декартовой системе координат: X-начальное, Y-начальное, X-конечное, Y-конечное) и определить команды задания цвета и построения линии. Например, в популярном языке программирования высокого уровня Си для программиста это эквивалентно заданию и выполнению следующих двух команд:

setcolor (red); // задание красного цвета;

line(Xн, Yн, Xк, Yк) // построение линии с координатами (Xн, Yн)(Xк, Yк).

Однако не всегда оказывается очевидным как это реализуется на аппаратном уровне. На самом деле эти команды раскладываются (в машинном коде системы команд выполняющего программу процессора) в последовательность управляющих слов и команд графического контроллера, который осуществляет роспись ячеек в видео ОЗУ по жестко определенному алгоритму задания цвета, построения линий и т.д.

Такие алгоритмы сегодня существуют для достаточно большого числа различных графических примитивов:

– алгоритмы построения линий различной толщины (в 1 пиксель, в 2, в 3) и атрибутов (пунктирная линия, мигание линий и т.д.);

– алгоритмы построения дуг, окружностей, эллипсов и т.д., основанные на задания точки центра окружности, радиуса, начальных/конечных углов;

– алгоритмы генерации различных шрифтов и символов согласованной конфигурации;

– алгоритмы построения фигур и объектов произвольной формы, их «заливка» (закрашивание или штриховка заданным элементом изображения и т.д.) и в той или иной степени стандартизированы.

Фирмы-разработчики (в частности, Motorolla, Intel, Texes Instruments и т.д.) таких «чипов» стараются унифицировать их аппаратные интерфейсы для обеспечения программной совместимости рабочих индикационных программ. Действующий сегодня стандарт получил название VESA и распространяется на большинство аппаратно-программных средств современных систем отображения информации. Стандарт непрерывно совершенствуется и накладывает ограничения в первую очередь на систему команд VESA-

совместимых графических контроллеров и на форматы данных (что достигается за счет программной поддержки «чипов» библиотеками драйверов), оставляя на усмотрение разработчиков вопросы организации внешнего интерфейса микросхем памяти (видео ОЗУ), управляющей шины и специализированных интерфейсов ЖК-экранов.

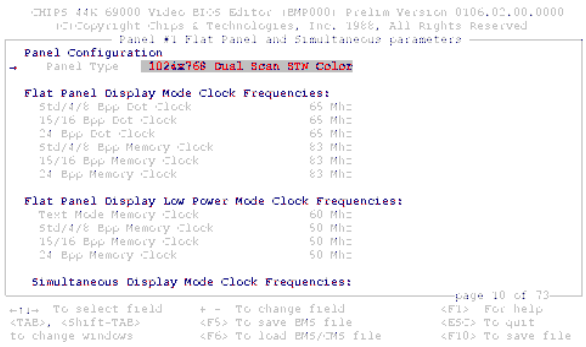


Рис.2.Вид оболочки для разработки новых «прошивок» графических контроллеров фирмы Motorolla.

В стремлении к реализации своей продукции в информационно-управляющих системах некоторые перспективные фирмы-производители микросхем графических контроллеров предлагают инженерам-схемотехникам набор программных инструментов (см. рис.2) для модификации базовых «прошивок» микросхем с целью обеспечение интерфейсного пользовательского взаимодействия с ЖК-экранами различного типа за счет возможности изменения временных соотношений (диаграмм) тактовых CLK, кадровых VSYNC и строчных HSYNC управляющих сигналов в широких пределах.

Типовая структура видео ОЗУ.

Видео ОЗУ, как правило, представляет собой набор микросхем памяти разделенный на два, реже — на три, т.н. банка с общим (в пределах каждого банка) адресным пространством. Такая организация памяти позволяет осуществлять бесконфликтное чтение/запись со стороны графического контроллера и ЖК-экрана. При этом, очевидно, в один банк памяти видео ОЗУ производится запись графической информации со стороны контроллера (аппаратное формирование изображения), из второго банка памяти видео ОЗУ производится считывание информации во временной диаграмме экрана для ее отображения. Третий банк, иногда используемый в 3-х банковом ОЗУ, обычно используется для буферизации (в случае необходимости совмещения изображений: наложение графической информации на телевизионный растр, наложение графической информации на радиолокационный кадр метео РЛС и т.д.) или стирания — обновления информации на экране в каждом цикле программно реализуемой задачи, гарантирующего отсутствие на экране «следов» предыдущего кадра изображения или неточностей графики при осуществлении, в частности, поворота отдельных частей изображения на экране или общего поворота изображения. При отсутствии третьего банка ОЗУ стирание осуществляется перед формированием изображения, что естественно сказывается на общем быстродействии графического контроллера.

Объем (разрядность) видео ОЗУ зависит от разрешающей способности экрана, на который производится вывод изображения. Например, пусть необходимо определить объем памяти ОЗУ для вывода изображения на экран с разрешением 640x480 пикселей с типовым кодированием изображения по 6 бит на каждый из основных цветов: красный, зеленый и синий, т.е. общая разрядность цвета — 18 бит, а количество воспроизводимых цветов и оттенков $2^6 \cdot 2^6 \cdot 2^6 = 2^{18}$.

Так, разрешение 640x·480 пикселей составляет 307200 пикселей, каждому из которых должна соответствовать индивидуальная 18-ти разрядная ячейка памяти (наиболее тривиальный случай без использования средств кодирования или упаковки). Учитывая, что технология современной промышленности выпускает микросхемы памяти с разрядностью адресного пространства, кратному числу 2, в такой, возможно нетипичной для микропроцессорной техники, разрядности данных выпускаются в современной промышленности (см. разработки и каталоги фирм Cypress, Alliance Semiconductor и т.д) микросхемы динамического ОЗУ, которые и используются, как правило, для организации видео ОЗУ.

Использование же микросхем статического ОЗУ осложнено, т.к. они не выпускаются с такой разрядностью и требуется их специальное каскадирование для повышения их общей разрядности по данным. А это, в свою очередь, нежелательно, т.к. возникают дополнительные вопросы габаритов, энергопотребления и т.д.

Система команд графических контроллеров. Общие сведения.

Система команд современного графического контроллера достаточно объемна и разнообразна. В ней содержится набор аппаратно реализуемых команд, каждая из которых имеет свое определенное описание. Например, известно, что в большинстве случаев при

использовании элементарной команды построения линии на экране Line(x1,y1,x2,y2) сама команда может быть разложена на последовательность команд ввода/вывода от процессора до графического контроллера (например, на языке Си):

```
outport(Adr, Dn); /* Dn - код команды Линия, Adr - адрес регистра команд графического контроллера */
outport(Adr, x1); /* x1 - значение начальной X-координаты */
outport(Adr, y1); /* y1 - значение начальной Y-координаты */
outport(Adr, x2); /* x2 - значение конечной X-координаты */
outport(Adr, y2); /* y2 - значение конечной Y-координаты */.
```

Очевидно, что при запасе общей производительности аппаратных средств индикатора, содержащего такой графический контроллер, ту же функцию может выполнит и сам процессор напрямую (в режиме прямого доступа к памяти видео ОЗУ) осуществляя роспись изображения. Только количество операций ввода/вывода в этом случае определяется уже не фиксированными 5-ью командами, а пропорционально длине линии, т.к. росписи подлежит каждый пиксель.

На принципе аппаратной поддержки работают современные интеллектуальные графические контроллеры и графоускорители. Интеллект же определяется размерностью аппаратно поддерживаемой системы команд.

Наибольшее распространение получили алгоритмы построения: линии, дуг, окружностей и секторов, прямоугольников, закрашенных областей на экране определенной формы и т.п.

Так, например, окружность, как известно, задается общей командой: circle(x, y, r), где x, y - координаты центра окружности радиуса r; дуга задается командой arc(x, y, r, α1, α2), где x, y - координаты центра окружности радиуса r, из которой формируется дуга с начальными и конечными углами α1, α2. Аналогично соответствующими командами и параметрами-переменными могут быть заданы другие интеллектуальные команды.

Для примера целесообразно рассмотреть взаимосвязь математических выражений алгоритмов и их аппаратно реализуемых аналогов, в частности, на базовом алгоритме машинной графики — алгоритме рисования элементарной линии.

Алгоритм построения линии. Математические выражения [1].

Назначение генератора линий — соединение двух точек изображения на экране отрезком прямой.

При построении элементарных отрезков на экранах современных средств индикации предъявляются определенные требования к качеству их изображения:

– конечные координаты отрезков в системе координат экрана должны находиться в заданных точках (пикселях);

– отрезки прямых должны визуально наблюдаться на экранах как прямые линии,

– яркость (насыщенность) цвета вдоль линии должна быть постоянной и не зависеть от длины и наклона линии.

Разумеется в общем случае ни одно из этих условий не может быть абсолютно точно выполнено при построении изображения на экранах с матричной структурой из пикселей конечных размеров, т.к.:

– окончание отрезков в общем случае располагается в пикселях, лишь наиболее близких к требуемым позициям и только в частных случаях координаты концов отрезка точно совпадают с координатами пикселей;

– отрезок аппроксимируется набором пикселей и лишь в частных случаях — вертикальных, горизонтальных и отрезков под 45° — они будут выглядеть гладкими прямыми, причем гладкими прямыми без ступенек (см. рис.3);

– яркость для различных отрезков и даже вдоль одного отрезка в общем случае различна, так как, например, расстояние между центрами пикселей для вертикального отрезка и отрезка под 45° различно (см. рис.3).

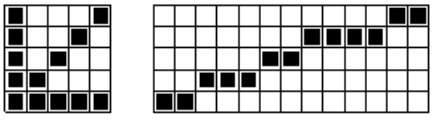


Рис.3. Растровое представление линий с различными углами наклона.

Объективное улучшение качества изображения достигается за счет увеличения разрешения отображающего экрана, но в силу существенных технологических проблем изготовления этот путь не бесконечен.

Субъективное улучшение качества аппроксимации изображения на экране основано на психофизиологических особенностях восприятия человека и, в частности, может достигаться за счет уменьшения размеров экрана. Другие способы субъективного улучшения

качества аппроксимации основаны на различных программно-программных ухищрениях по «размыванию» резких границ изображения на экране.

Алгоритм Брезенхема [1]. В конце 20-ого века исследователь Брезенхем предложил алгоритм формирования (генерации) изображения на экране устройств отображения обеспечивающий минимизацию отклонения искусственной линии от истинного отрезка, заданного 4-мя координатами: Xн - начальная координата X, Yн - начальная координата Y, Xк - конечная координата X, Yк - конечная координата Y.

Основная идея алгоритма состоит в анализе коэффициента углового наклона желаемой прямой: если угловой коэффициент прямой < 1/2, то точку, следующую за точкой с координатами (0,0), необходимо поставить в позицию с координатами (1,0) (см. рис.4, а), а если угловой коэффициент > 1/2, то — в позицию с координатами (1,1) (см. рис.4, б).

Решающее правило о порядке заполнения пикселей вдоль формируемого отрезка, очевидно, основано на анализе величина отклонения E точной позиции от середины между двумя возможными растровыми точками в направлении наименьшей относительной координаты. Знак E используется как критерий для выбора ближайшей растровой точки.

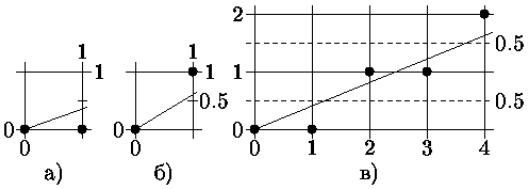


Рис.4. Алгоритм Брезенхема генерации отрезков.

Тогда, если величина E < 0, то точное Y-значение округляется до последнего меньшего целочисленного значения Y, т.е. Y-координата не меняется по сравнению с предыдущей точкой. В противном случае Y увеличивается на 1.

Для вычисления величины E без ограничения общности можно полагать, что рассматриваемый вектор начинается в точке с координатами (0,0) и проходит через точку с координатами (4, 1.5) (см. рис.4, в), т.е. имеет положительный наклон, меньший 1. Из рис.4, в видно, что отклонение для первого шага:

$$E_1 = Py/Px - 1/2 < 0,$$

поэтому для занесения пикселя выбирается точка с координатами (1,0). Отклонение для второго шага вычисляется добавлением приращения Y-координаты для следующей X-позиции (см. рис.4, в):

$$E_2 = E_1 + Py/Px > 0,$$

поэтому для занесения пикселя выбирается точка с координатами (2,1). Так как отклонение отсчитывается от Y-координаты, которая теперь увеличилась на 1, то из накопленного отклонения для вычисления последующих отклонений надо вычесть 1:

$$E_2 = E_2 - 1.$$

Отклонение для третьего шага:

$$E_3 = E_2 + Py/Px < 0,$$

поэтому для занесения пикселя выбирается точка с координатами (3,1).

Суммируя и обозначая большими буквами растровые точки, а маленькими — точки вектора, получаем:

$$E_1 = y_1 - 1/2 = dY/dX - 1/2.$$

При этом возможны следующие случаи:

$E_1 > 0$	$E_1 \le 0$
ближайшая точка:	
$X_1 = X_0 + 1;$	$X_1 = X_0 + 1;$
$Y_1 = Y_0 + 1;$	$Y_1 = Y_0;$
$E_2 = E_1 + Py/Px - 1;$	$E_2 = E_1 + Py/Px.$

Так как интересует только знак величины E, то можно избавиться от неудобные частных умножением E на 2×Px:

$$E_1 = 2\times Py - Px$$

Для $E_1 > 0$: $E_2 = E_1 + 2\times(Py - Px)$

Для $E_1 \le 0$: $E_2 = E_1 + 2\times Py$

Таким образом, получается алгоритм, в котором используются только целые числа, сложение, вычитание и сдвиг.

Программное построение линии поточечно в видео ОЗУ	Программное построение линии с аппаратной поддержкой функции

X= x1; Y= y1;
Px= x2 – x1;
Py= y2 – y1;
E= 2Py – Px;
i= Px;
PutPixel(X, Y); //
Первая точка линии
while (i= i – 1 ≤ 0)
{
if (E ≤ 0) {
X= X + 1; Y= Y + 1;
E= E + 2(Py – Px);
} else X= X + 1;
E= E + 2Py;
PutPixel(X, Y); //
Очередная точка линии }
	Line(x1,y1,x2,y2);

Этот алгоритм пригоден для случая 0≤dY≤dX. Для других случаев алгоритм строится аналогичным образом. На рис.5 приведен пример генерации линии по алгоритму Брезенхема.

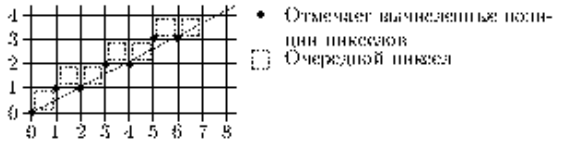


Рис.5. Генерация отрезка по алгоритму Брезенхема.

Аппаратная поддержка функции построения линии в видео ОЗУ.

Задача: необходимо разработать алгоритм аппаратной поддержки построения на экране элементарной линии в декартовой системе координат с любым наклоном произвольной длины и в произвольном месте экрана с фиксированным разрешением, не превышающим, скажем, 640x480 пикселей.

Такую задачу, очевидно, необходимо решать в несколько этапов (тактов):

– принять и дешифрировать команду построения на экране элементарной линии: line;

– принять, выделить и зафиксировать начальные и конечные значения координат линий: X-начальное, Y-начальное, X-конечное, Y-конечное;

– определить значения и знак приращений Δ, характеризующих наклон и длину линий, для реализации последующей процедуры построения линии: Δx=Xк-Xн, Δy=Yк-Yн;

– сравнить значения Δx и Δy между собой оценив максимальное число итерационных сложений/вычитаний при переборе текущих XY-координат, составляющих на экране линии в целом;

– произвести итерационный расчет текущих XY-координат с их фиксацией для каждого пикселя, составляющего линию.

В результате аппаратной реализации такого алгоритма во внешнем видео ОЗУ графическим контроллером формируется образ линии с соответствующим заполнением ячеек памяти. По окончании аппаратного построения растр из видео ОЗУ считывается на ЖК-экран и формируется визуально наблюдаемое изображение линии.

Такт 1: принятие и дешифрация команды построения на экране элементарной линии.

Как уже отмечалось ранее, каждая команда интеллектуального графического контроллера может быть разложена на последовательность элементарных команд ввода/вывода, содержащих информацию о коде команды и о данных параметрах-переменных команды.

Учитывая, что в системе команд графического контроллера каждой команде соответствует свой индивидуальный код, этот код может быть дешифрирован, обеспечивая, тем самым, аппаратное разделение разных команд друг от друга.

Прием команды и данных параметров-переменных осуществляется в МФЦИ по межмодульному интерфейсу, а собственно сам прием реализуется аппаратными средствами, выполняющими функции ввода/вывода в форматах временных диаграмм шины. Принятая таким образом и запомненная в регистр команда попадает на дешифратор, с одного из выходов которого формируется битовый сигнал: принята команда Линия.

Такт 2: принятие, выделение и фиксация начальных и конечных значений координат линии.

Каждой команде в системе команд интеллектуального графического контроллера соответствует фиксированное число слов и последовательность их передачи по шине данных. В частности, команде Линия соответствует 4-ре параметра, поступающих в графический контроллер в строго определенной последовательности. Зная это, и, приняв с дешифратора команд бит Линия, 4-ре последующих слова последовательно записываются в 4-ре персональных регистра D1-D4, фиксирующих значения начальных и конечных координат прямой. По окончании принятия четвертого слова данных можно считать, что передача команды и параметров от процессора в графический контроллер состоялась.

На рис.6 приведен пример организации регистров записи 10-ти разрядных данных, что в свою очередь достаточно для задания линии в координатах (0,0) — (1024,1024) и более чем достаточно для задания линии на экране с размерами 640x480 пикселей.

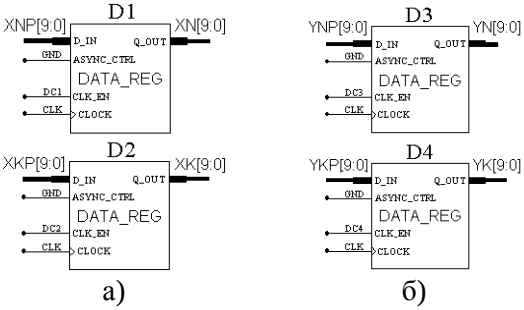


Рис.6. Регистры хранения начальных и конечных координат прямой.

Такт 3: Определение значения и знака приращений Δx и Δy, характеризующих наклон и длину линии.

При известных значениях начальных и конечных координат прямой определение приращения производится как Δx=Xк-Xн, Δy=Yк-Yн, а знак определяется соотношение значений Xк>/<Xн, Yк>/<Yн между собой.

Операция вычитания, как известно, реализуется в общем случае на сумматоре, а операция сравнения — на компараторе.

Тогда (см. рис.7), если подать на первый вход компаратора D5 значение Xн, а на второй — Xк, то на его выходе образуется логический сигнал ZnX, уровень которого определяет отношение Xк, Xн между собой и характеризует знак будущего приращения Δx. Собственно само приращение рассчитывается на сумматоре D11 как разница данных, поступающих с выхода регистров D9 и D10. Элементы D6-D8 определяют какое из двух слагаемых должно быть переведено в дополнительный код при реализации функции сложения на D11. Таким образом с выхода схемы на рис.7,а получается сигнал ZnX, характеризующий знак приращения (наклон прямой) и код DX, характеризующий значение приращения. Одновременно аналогичные действия производятся по каналу Y-координаты для вычисления приращения Δy (см. рис.7,б): на компараторе D12 определяется знак ZnY, элементы D13-D15 формируют дополнительный код, записываемый в регистры D16 и D17, а на сумматоре D18 определяется код значения приращения DY.

Приращения DX и DY, как уже отмечалось, определяют разницу абсцисс и ординат начальных и конечных координат линии. Сигналы ZnX и ZnY — направление (сложение или вычитание) при последующем расчете промежуточных координат пикселей, образующих на экране линию в целом.

а)

б)

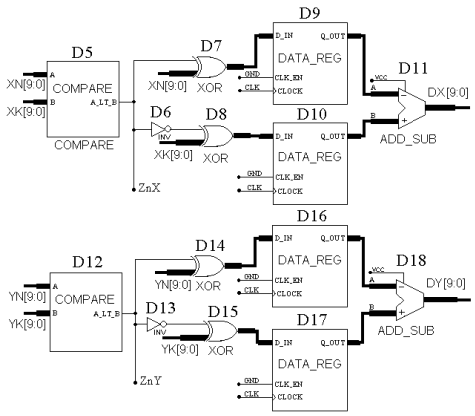


Рис.7. Аппаратное вычисление приращений при построении линии

Такт 4: сравнение значений Δx и Δy между собой для оценки общего числа итерационных сложений/вычитаний при переборе текущих XY-координат.

Как таковое сравнение значений величин кодов DX и DY можно было бы произвести на обычном компараторе, однако интерес представляет не само их соотношение, а, в том числе и абсолютное значение, т.к. оно определяет общее число итерационных процедур сложения/вычитания при выборе текущих XY-координат. В математических выражениях алгоритма построения линии, основу построения процедуры составляют операции приращений/вычитаний (инкремента/декремента) начальных или текущих координат для получения значений последующих. Так формируется линия.

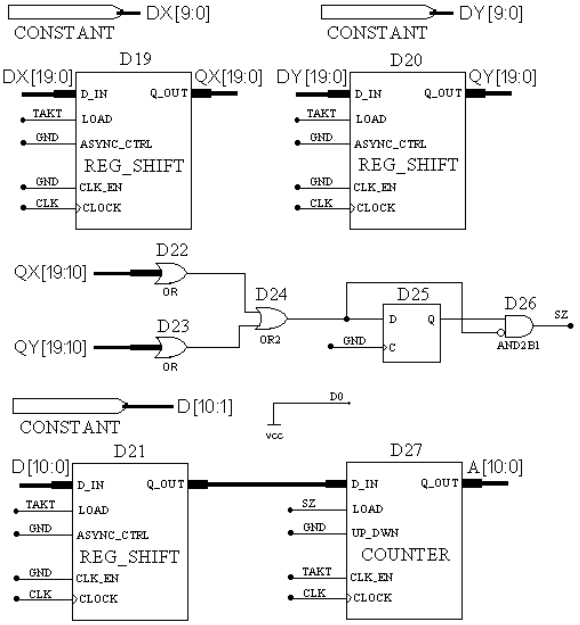


Рис.8. Организация расчета числа операций сложения/вычитания при аппаратном построении линии.

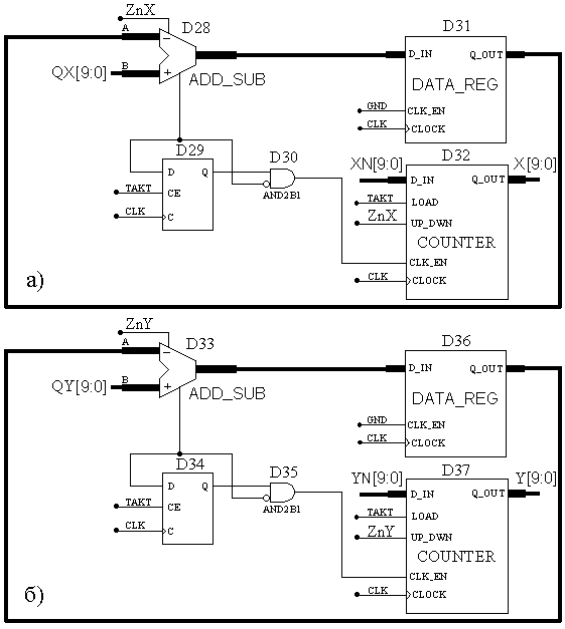
Для реализации такой задачи используется следующее оригинальное техническое решение (см. рис.8). Значения приращений Δx и Δy заносятся в старшие 10 разрядов данных регистров сдвига D19 и D20 соответственно. Младшие же 10 входных разрядов (вводятся дополнительно) представляют собой константы, равные нулю. Параллельно с загрузкой 20-ти разрядных данных в D19 и D20 производится загрузка 10-ти разрядного числа 1 в регистр сдвига D21. По окончании общей записи начинается сдвиг данных регистров D19-D21 по сигналам CLK.

Количество операций (длительность) сдвига определяется состоянием сигнала SZ, который формируется при достижении нулевых значениях старших 10-ти разрядов данных с выхода регистров D19 и D20. Появление всех нулей, контролируемых элементами D22-D26, в этих разрядах свидетельствует о том, что информативная часть (значимые разряды) кодов DX и DY закончились, что, в свою очередь, характеризует общее число последующих операций сложения/вычитания при расчете текущих XY-координат линии.

Тем временем на выходе регистра D21 логическая 1 из младшего разряда сдвинется в разряд, определяющий общее число сложений/вычитаний. Это число заносится в счетчик D27.

Такт 5: итерационный расчет текущих XY-координат с их фиксацией для каждого пикселя, составляющего линию на экране в целом.

Приращения DX и DY, сдвинутые из старших 10-ти разрядов на регистрах D19 и D20 (см. рис.9) в младшие 10 разрядов поступают на сумматоры D28 и D33, осуществляющие рекуррентное сложение/вычитание (определяется сигналом ZnX по X-каналу и ZnY по Y-каналу) текущей или начальной координаты со значением единичного приращения, формируя тем самым по сигналам переполнения на элементах D29, D30, D34, D35 управляющие сигналы на счетчики координат D32, D37. Их выходные значения определяют текущие значения XY-координат при построении линии. Итерации производятся число раз, определяемое числом в счетчике D27.



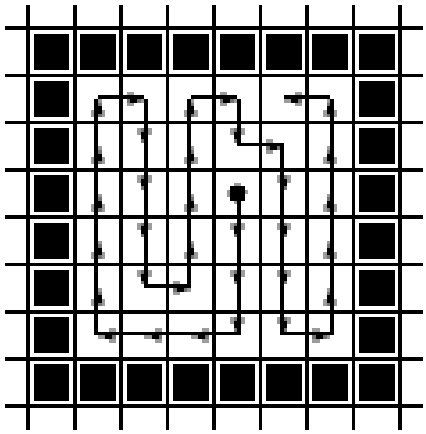


Рис.10. Пример построения на экране структуры изображения, состоящего из элементарных линий.

Счетчик D27 осуществляет счет в обратную сторону (реверсивный счетчик) пропорционально числу сложений/вычитаний, по сигналу переполнения которого осуществляется общая остановка алгоритма. При этом процедура аппаратного построения линии считается состоявшейся.

Можно показать [1], что аппаратная реализация построения окружности, прямоугольника или других графических примитивов может быть сведена (см., в частности, рис.10) к использованию аналогичных узлов и схем, последовательно и аппаратно выполняющих функции сложения/вычитания и сдвига при реализации сравнительно несложных цифровых операций интерактивной машинной графики. Кроме того, представленные алгоритмы (математический и аппаратный) не являются единственными, а предмет их оптимизации — научный вопрос, заслуживающий отдельного исследования в соответствующей предметной области.

ЛИТЕРАТУРА

1. Вельтмандер П.В. Основные алгоритмы компьютерной графики. Машинная графика: Учеб. пособие / НГТУ. Новосибирск. 1997. кн.2.

2. Информационно-управляющие системы для подвижных объектов. Семинары ASK LAB 2001 / Сборник статей // Под общ. ред. М.Б. Сергеева — СПб.: Политехника, 2002. — 185с.