УДК 658.512.2;004.4:004.9

В.И. Романов, Ю.Ю. Ланкевич

ОРГАНИЗАЦИЯ ГРАФИЧЕСКОЙ ИНФОРМАЦИИ ДЛЯ ПРОСМОТРА МНОГОСЛОЙНОЙ ТОПОЛОГИИ СБИС

Предлагается один из возможных способов реорганизации графической информации, описывающей множество слоев топологии современных СБИС. Этот способ ориентирован на использование в условиях ограниченной по размеру памяти видеокарты. Дополнительный эффект, обеспечивающий высокое быстродействие формирования демонстрируемого многокадрового изображения многослойной топологии современных СБИС, достигается за счет предварительной загрузки требуемых текстур при помощи вспомогательного фонового процесса.

Введение

Топология СБИС может быть представлена множеством слоев, каждый из которых, в свою очередь, состоит из множества кадров. Каждый кадр описывается отдельным файлом в некотором графическом формате. Учитывая доступную разрешающую способность современных микроскопов, а также размеры исследуемых микросхем, нетрудно показать, что для построения полного видеоизображения могут потребоваться десятки гигабайт памяти. Оказывается, что компьютеры, традиционно используемые при решении задачи анализа топологии СБИС, не в состоянии одновременно разместить в памяти видеокарты всю информацию. В силу последнего утверждения требует решения задача информационного обеспечения просмотра топологии в условиях ограничений по размерам доступной памяти видеокарты с одновременным сохранением дружественности интерфейса, выражающейся в высокой скорости предоставления пользователю необходимой порции видеоинформации. Для обеспечения кроссплатформенности разрабатываемого программного обеспечения предлагается использовать инструментарий Qt 5 [1], поддерживающий кроссплатформенную графическую библиотеку OpenGL [2]. Видеоадаптер компьютера, например NVIDIA GeForce GTX 760, должен обеспечивать поддержку стандартов библиотеки OpenGL 4.0 и иметь не менее 2 Гб видеопамяти.

Рассматривая вопрос обеспечения доступа к графической информации, обычно хранящейся на внешней памяти, можно сделать следующие выводы:

1) время доступа будет тем меньше, чем меньший объем данных потребуется прочитать из внешней памяти. На практике из этого следует, что для повышения скорости необходимо воспользоваться графическими форматами, обеспечивающими внутреннее сжатие информации. Применительно к функциональному набору, реализация которого осуществляется создаваемыми программными средствами, это означает, что в условиях «тяжелых» по объему данных необходимо осуществить однократную конвертацию графики, представленной в формате BMP, в формат DDS, поддерживаемый современными видеокартами в качестве встроенного формата, обеспечивающего высокую скорость обработки [3];

2) время доступа будет тем меньше, чем меньшее число файлов будет задействовано при переносе информации из внешней памяти в оперативную с ее последующим переносом в память видеокарты. Практически это означает, что имеет смысл перераспределить графическую информацию по файлам таким образом, чтобы в отдельном файле собиралось несколько графических объектов, одновременно видимых на экране.

1. Использование формата DDS (DirectDraw Surface)

Для всех слоев топологии отображаемые изображения конвертируются в графический формат DDS [3]. В данном формате обеспечивается возможность хранения текстур – растровых изображений, накладываемых на плоскость рисования топологии и загружаемых в память ви-

деокарты для последующей обработки. При организации формата DDS возможно применение MIP-текстурирования (MIP mapping), использующего несколько копий одной текстуры с разной детализацией.

MIP-текстурирование – это метод улучшения качества текстурных изображений при помощи текстур с разным разрешением для различных объектов одного и того же изображения в зависимости от их размера и глубины. Принцип работы метода заключается в создании так называемой MIP-пирамиды (рис. 1) – последовательности текстур с разрешением от максимального до 1×1, например: 1×1, 2×2, 4×4, 8×8, 16×16, 32×32, 64×64, 128×128, 256×256, 512×512 и 1024×1024. Каждая из этих текстур называется MIP-уровнем [4].

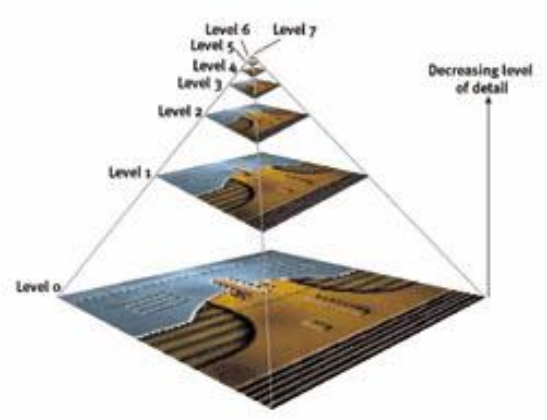


Рис. 1. Пирамидальность MIP-текстуры

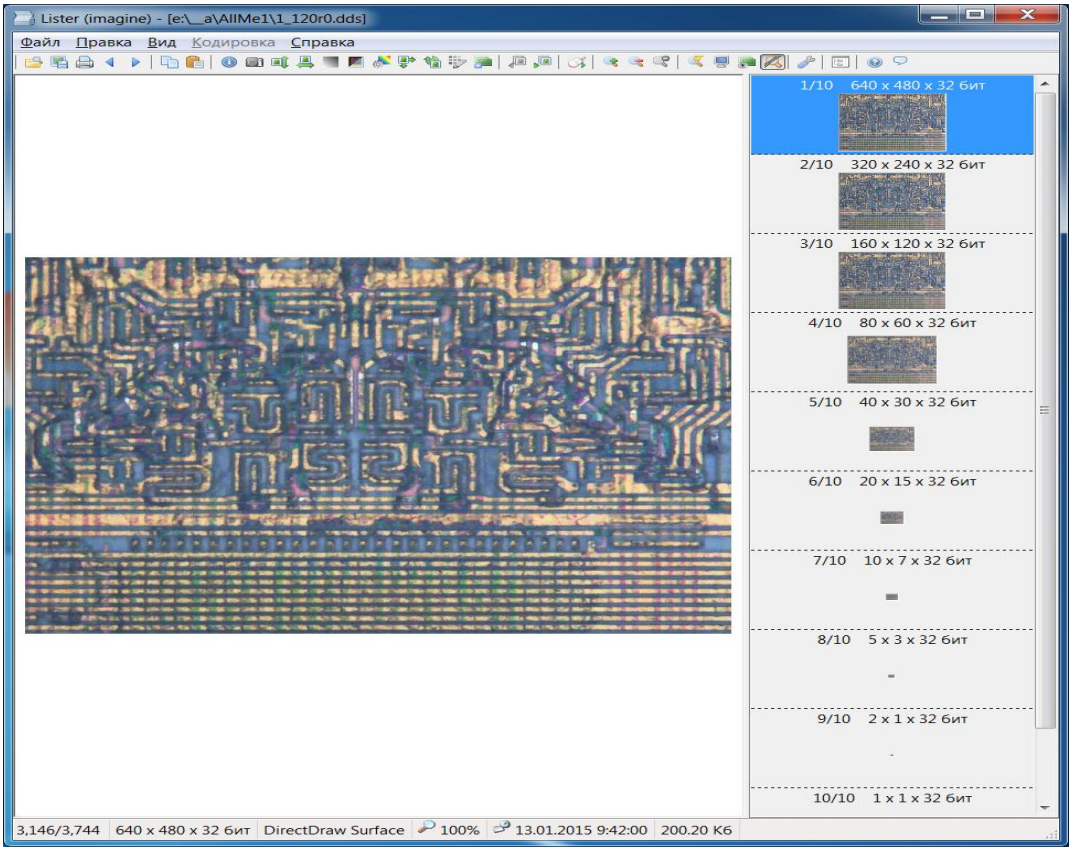


Рис. 2. Демонстрация одного файла топологии формата DDS

На всех текстурах находится одно и то же изображение. В результате расход видеопамяти при MIP-текстурировании увеличивается на треть в соответствии с формулой

$$\sum_{i=0}^{\infty}\left(\frac{1}{4}\right)^{i}=1\frac{1}{3}.$$

При отображении текстур вычисляется расстояние до плоскости рисования и номер (уровень) текстуры находится по формуле

$$\text{miplevel} = \log_2 (d / \text{texsize}\times\text{resolution}) + \text{mipbias},$$

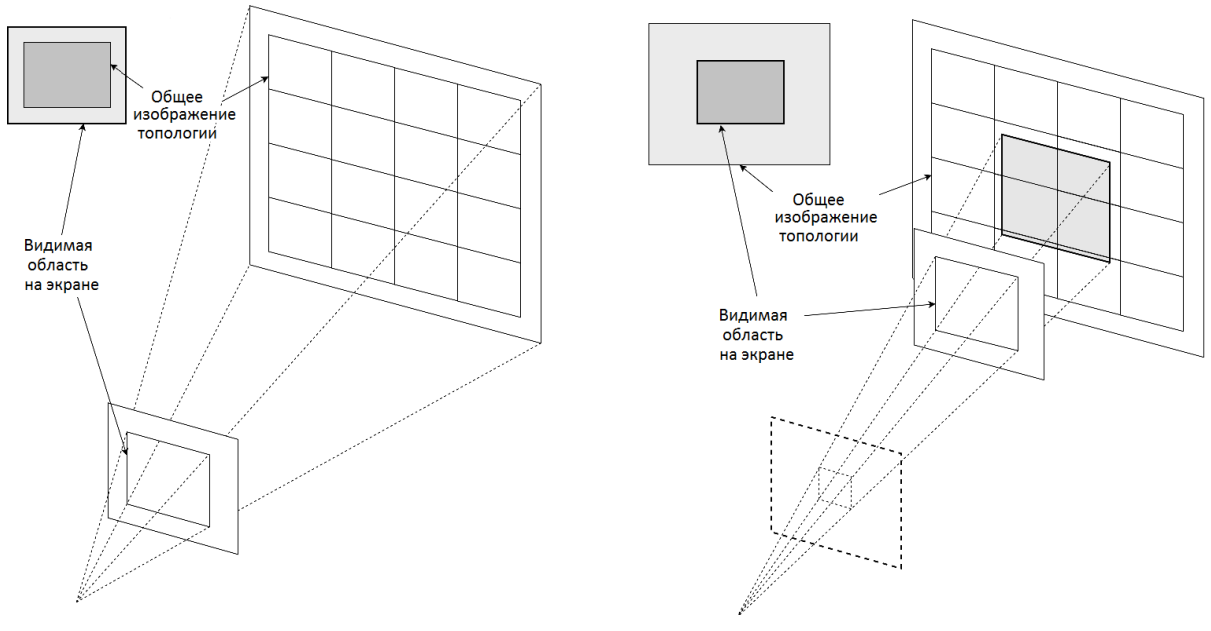
где resolution – разрешение виртуальной камеры (количество пикселов, которое будет в объекте размером в 1 ед., расположенном в 1 ед. от камеры); texsize – размер текселя в единицах трехмерного мира; d – расстояние до объекта в тех же единицах; mipbias – числовая поправка, позволяющая выбирать более или менее детальную текстуру, чем дает формула.

Значение miplevel округляется до целого, и текстура с соответствующим номером (нулевая – самая детальная, первая – вдвое меньшая и т. д.) используется для отображения.

Как уже было сказано, на практике все MIP-уровни текстуры хранятся в одном файле. На рис. 2 показано отображение одного из кадров реальной топологии слоя СБИС. Видно, что правая панель содержит определение всех MIP-уровней демонстрируемого файла формата DDS.

2. Проведение просмотра

При просмотре топологии используется модель, показанная на рис. 3. В ней имеется плоскость изображения, где размещены все кадры отдельного слоя топологии с тем разрешением, которое было использовано при съемках на микроскопе. Между «проектором» и указанной плоскостью изображения размещена параллельная ей дополнительная плоскость, соответствующая экранному окну, которое ограничивает видимый топологический фрагмент. Исходя из позиция «проектора» можно рассчитать масштабирование представленного на экране рисунка.



а) *б)*

Рис. 3. Масштабирование многокадрового изображения: *а)* исходное состояние отображения, при котором коэффициент масштабирования выбирается так, что в окне экрана видна вся схема; *б)* последующая демонстрация топологии, обеспечивающая видимость выбранного пользователем фрагмента

Фактически «сближение» плоскостей экрана и изображения приводит к увеличению масштаба изображения и позволяет пользователю рассмотреть его более детально, но при этом в окне становится видимой только небольшой фрагмент общего изображения топологии.

Многослойность топологии современных СБИС в рамках рассматриваемого подхода учитывается в виде нагрузки на использование ресурсов памяти: чем больше слоев участвует в формировании топологического изображения, тем больше затраты требуемой памяти и времени на ее обслуживание. Именно этот факт делает еще более актуальными предлагаемые средства оптимизации памяти.

Прочие аспекты отображения многих слоев в одном изображении в рамках рассматриваемой модели являются несущественными, поскольку в оперативной памяти компьютера или памяти видеоадаптера всегда представлены все слои одного проекта. Одновременное отображение нескольких слоев обеспечивается за счет возможности использования свойства «прозрачности» изображения слоя. Масштабирование, позиционирование и поворот на плоскости изображений осуществляются пользователем при помощи предлагаемых средств проведения просмотра независимо для каждого отдельного слоя.

Вариативность масштабирования показанного на экране изображения хорошо поддерживается на уровне драйвера видеокарты при условии использования MIP-текстурирования – драйвер самостоятельно выбирает наиболее подходящий из представленных в текстурах уровней и освобождает программу просмотра от решения данной задачи.

Анализ действий пользователя в процессе просмотра топологии показал, что можно условно выделить два режима. Один из них, который можно назвать *навигацией*, связан с решением задачи выбора на всем изображении слоя некоторого фрагмента, представляющего интерес для последующего детального анализа. При этом используется достаточно большой набор кадров, каждый из которых рассматривается без углубления в детали рисунка. Альтернативный навигации режим *анализа* связан с детальным рассмотрением совсем небольшого по количеству задействованных кадров фрагмента топологии.

3. Реорганизация данных

Наличие двух режимов просмотра – навигации и анализа – послужило основой предлагаемой реорганизации графической информации.

Для каждого слоя из всего множества образующих его файлов строится единый так называемый MAP-файл, размер которого устанавливается параметрически в зависимости от количественных оценок ширины и высоты слоя в числе кадров, а также размеров используемой памяти компьютера и установленной на нем видеокарты.

При этом в результирующий файл отбираются не все, а только несколько самых верхних (наименьших по объему данных) MIP-уровней от каждой текстуры. На рис. 4 показано графическое представление построения MAP-файла текстур. Использование единого файла для хранения текстур позволяет достаточно существенно сократить время формирования памяти видеокарты.

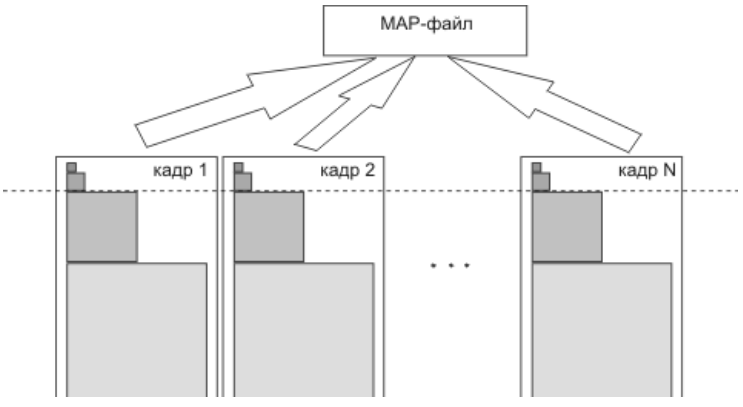


Рис. 4. Принцип отбора в один файл верхних MIP-уровней формата DDS

Отказываясь от нижних (больших по размеру) уровней, разработчик намеренно «огрубляет» изображение: при большом масштабе отображения видеоизображение будет интерполироваться драйвером видеокарты и выглядеть размытым. Однако данный эффект будет заметен только в условиях сильного «приближения» рисунка. Фактические размеры съемки отдельного кадра ограничиваются линейными размерами в несколько сотен точек. Если предположить, что для анализа топологии будут использованы мониторы с высокой разрешающей способностью [5], то и в этом случае общее число MIP-уровней будет не слишком большим: для размера 2560×1440, являющегося статистически заметным (несколько последних лет таким разрешением обладает 1 % от всех используемых дисплеев настольных компьютеров), число уровней равно 12. Учитывая разрешение современных дисплеев, можно заключить, что точное детальное отображение потребуется в условиях видимости на экране только нескольких кадров, во всех остальных случаях представление топологии будет происходить без информационных потерь.

В таблице приведены данные, касающиеся размеров требуемой памяти при использовании MIP-текстурирования. Анализ таблицы показывает, что допустимые размеры MAP-файла текстур (определяемые двумя последними колонками таблицы) ограничиваются восьмым-девятым MIP-уровнями. При небольшом числе кадров, например 30×20, размер MAP-файла даже с включением 10-го уровня составит порядка 2,5 Гб и одновременно видимыми окажутся не более девяти кадров.

Оценки памяти при просмотре топологии слоя с использованием MIP-текстурирования

Номер MIP-уровня	Ширина кадра *w*, пикселы	Высота кадра *h*, пикселы	Представление пиксела, байт	Размер уровня, байт	Размер кадра (текстуры), байт	Объем памяти представления 60×60 кадров, байт	Объем памяти представления 100×100 кадров, байт
11	2048	1536	4	12 582 912	16 777 208	60 397 948 800	167 772 080 000
10	1024	768	4	3 145 728	4 194 300	15 099 480 000	41 943 000 000
9	512	384	4	786 432	1 048 572	**3 774 859 200**	10 485 720 000
8	256	192	4	196 608	262 140	943 704 000	**2 621 400 000**
7	128	96	4	49 152	65 532	235 915 200	655 320 000
6	64	48	4	12 288	16 380	58 968 000	163 800 000
5	32	24	4	3 072	4 092	14 731 200	40 920 000
4	16	12	4	768	1 020	3 672 000	10 200 000
3	8	6	4	192	252	907 200	2 520 000
2	4	3	4	48	60	216 000	600 000
1	2	1	4	8	12	43 200	120 000
0	1	1	4	4	–	–	–

4. Определение набора одновременно видимых кадров

При поддержке просмотра топологии в режиме анализа возникает задача определения контекста ее видимой локальной области и предварительного размещения необходимых текстур в памяти видеокарты.

Вывод кадров изображения осуществляется в некоторой координатной сетке с определенным шагом по горизонтали и по вертикали. Только после привязки кадров к сетке осуществляются такие преобразования выводимого изображения, как поворот, масштабирование и сдвиг, реализуемые при помощи известных функций библиотеки OpenGL [2]. При этом для сопоставления отдельных кадров с глобальными координатами отображения используются:

– величина сдвига по горизонтали (*hShift*) и по вертикали (*vShift*);
– величина шага по горизонтали (*xStep*) и по вертикали (*yStep*);
– масштабные коэффициенты (*hScale* и *vScale*);
– угол поворота (*ang*).

После геометрических преобразований, таких как поворот, масштабирование или сдвиг, глобальные координаты узлов сетки меняются. Поэтому, чтобы получить соответствие между глобальной координатой и конкретным кадром из сетки, следует выполнить обратные координатные преобразования:

– для обратного сдвига

$$x' = x - hShift; \quad (1)$$

$$y' = y + vShift; \quad (2)$$

– для обратного масштабирования

$$x'' = x' / hScale; \quad (3)$$

$$y'' = y' / vScale; \quad (4)$$

– для обратного поворота

$$x''' = x'' * cos(ang) - y'' * sin(ang); \quad (5)$$

$$y''' = x'' * sin(ang) + y'' * cos(ang); \quad (6)$$

– итоговые преобразования, обеспечивающие вычисление позиции кадра по горизонтали и по вертикали,

$$j = \lceil x''' \rceil / xStep; \quad (7)$$

$$i = \lceil y''' \rceil / yStep. \quad (8)$$

Рассмотрим решение следующей задачи. На рис. 5 все текстуры лежат в прямоугольнике *ABCD*, а область отображения (окно видимости) обозначена прямоугольником *A'B'C'D'*. Необходимо выбрать только те текстуры, которые попадают в окно отображения (*A'B'C'D'*) полностью или частично.

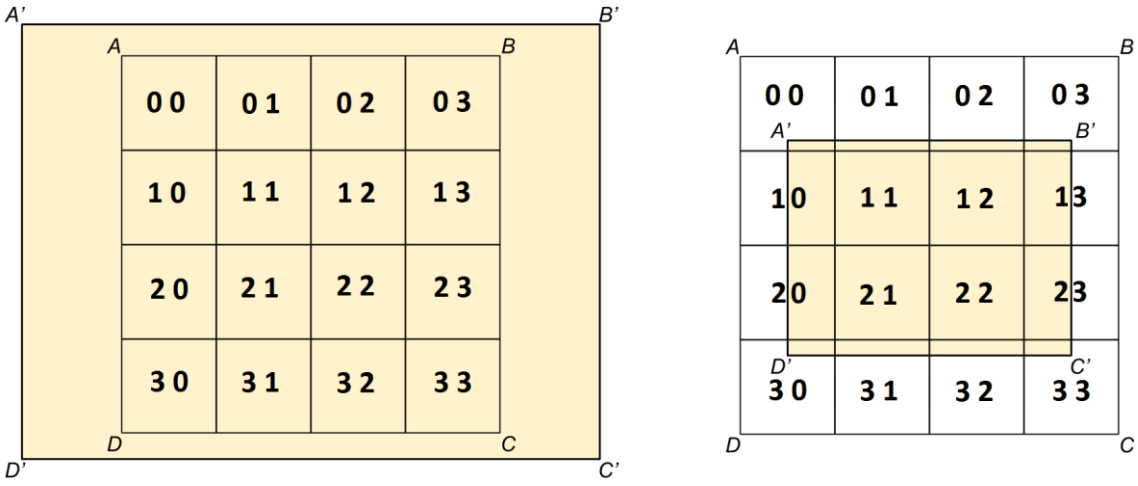


Рис. 5. Попадание всех текстур в область отображения: *а)* полное; *б)* частичное

Алгоритм отбора подмножества текстур, отображаемых в окне видимости, основывается на использовании координат середины окна отображения, его ширины и высоты, масштаба, угла поворота, ширины и высоты текстуры отдельного кадра и реализуется следующим образом:

Шаг 1. Вычисляются координаты вершин *A'*(*x*, *y*), *B'*(*x*, *y*), *C'*(*x*, *y*), *D'*(*x*, *y*) с помощью выражений (1)–(6).

Шаг 2. Вычисляются номера (индексы) текстур, в границах которых лежат угловые точки окна отображения (*A'*, *B'*, *C'*, *D'*), с помощью выражений (7), (8).

Шаг 3. Находятся наименьшие и наибольшие номера строк и столбцов из тех, что были получены на шаге 2.

Предложенный алгоритм подходит и для случая, когда изображение слоя повернуто относительно окна отображения (рис. 6). В данном случае кадры {0 0} и {3 3} не попадают в область отображения *A'B'C'D'*.

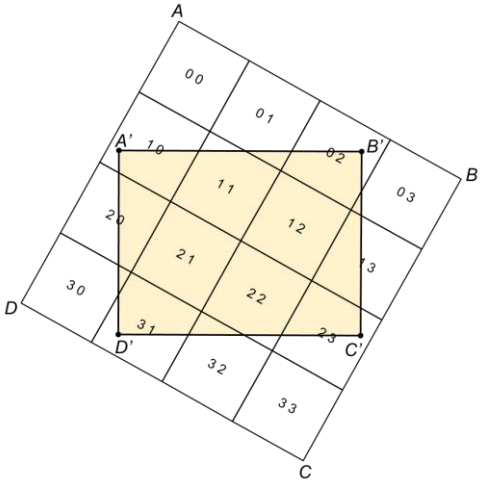


Рис. 6. Частичное попадание текстур в область отображения при их повороте относительно этой области

5. Фоновая загрузка текстур видимой области

На основе предложенного алгоритма можно не только построить подмножество кадров, присутствующих в видимой области, но и организовать их прогнозную загрузку, которая основывается на том факте, что при неизменном масштабе отображения и использовании встроенных в программу средств смещение локального окна видимости без изменения масштаба отображения не приведет более чем к однорядному отклонению.

Уточнение и загрузка набора текстур, участвующих в отображении топологии в текущий момент времени, встраиваются в программу на основе своего рода виртуализации используемых ресурсов, поддерживаемых представленной на рис. 7 архитектурой.

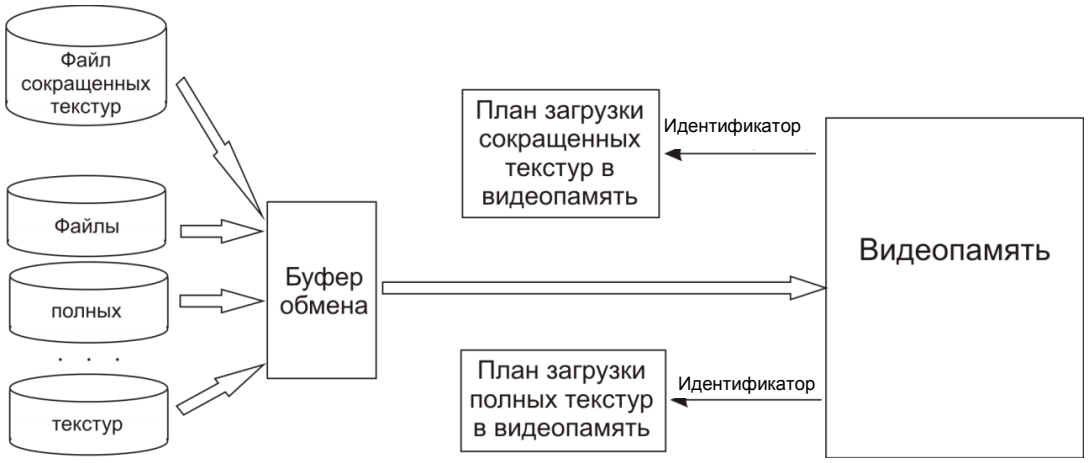


Рис. 7. Архитектура организации памяти

Планы загрузки текстур (как полных, так и сокращенных) представляют собой таблицы кадров слоя, где каждая ячейка предназначена для хранения идентификатора отдельной текстуры, загруженной в память видеокарты. Однако принципы управления планами существенно различаются.

План загрузки сокращенных текстур строится единожды из заранее сформированного MAP-файла при начальной загрузке данных слоя и содержит идентификаторы всех сокращенных текстур слоя. Такие текстуры присутствуют в памяти видеокарты в течение всего сеанса работы над данным проектом.

В отличие от плана сокращенных текстур план полных текстур формируется динамически в процессе сеанса. Полная текстура содержит в себе все уровни MIP-текстурирования и размещается в видеопамяти по специальному запросу управляющей программы просмотра при необходимости построения изображений топологии, требующих использования высокого разрешения.

Загрузчик полных структур представляется фоновым процессом, который реализует виртуализацию памяти на основе обеспеченного в среде Qt [1] сигнально-слотового взаимодействия.

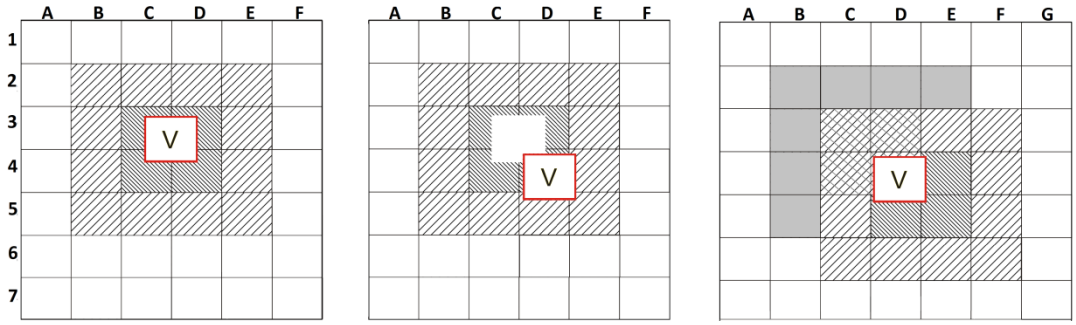
В программе реализации управления просмотром вырабатываются сигналы:

– «запустить управление загрузкой полных текстур»;

– «изменить набор видимых текстур» (при работе с полными текстурами);

– «освободить видеопамять от всех полных текстур» (при выключении режима работы с полными текстурами).

Предположим следующее: после очередного увеличения масштаба изображения управляющая программа обнаружила, что выполняется условие необходимости предъявления пользователю изображения с высоким разрешением и область видимости (отмечена буквой V) размещена так, как показано на рис. 8, *а*. В этом случае вырабатывается сигнал включения режима демонстрации полных текстур. Данный сигнал перехватывается фоновым процессом обработки полных текстур, который, используя доступные ему данные о позиции и размерах окна видимости, определяет множество кадров, чьи текстуры попадают хотя бы частично в область видимости (в примере это текстуры C3, D3, C4, D4, отмеченные густой обратной штриховкой). Если указанных текстур в текущий момент нет в памяти видеокарты, то организуется их загрузка с фиксацией полученных идентификаторов в плане. По завершении подготовки графической информации фоновый процесс информирует управляющую программу о готовности памяти, используемой при построении требуемого изображения. После этого работа управляющей программы и работа фонового процесса протекают параллельно. В то время как управляющая программа занимается прорисовкой топологии, в фоновом процессе осуществляется загрузка в память видеокарты всех текстур, размещенных по соседству с текстурами области видимости и обозначенных прямой штриховкой: B2, C2, D2, E2, B3, E3, B4, E4, B5, C5, D5, E5.



а) *б)* *в)*

Рис. 8. План загрузки полных текстур: *а)* по сигналу «включить режим отображения полных текстур»; *б)* сдвиг видимого фрагмента вправо и вниз; *в)* по сигналу «изменить набор видимых текстур»

Предположим далее, что пользователь сместил область видимости так, как это показано на рис. 8, *б* – вправо и вниз. Возникшая ситуация инициирует посылку сигнала «изменить набор видимых текстур».

Получив указанный сигнал, фоновый процесс оценивает ситуацию на необходимость проведения новой загрузки. На рис. 8, *в* видно, что для реализации отображения в новой позиции не потребуется проведение загрузки новых текстур – вся область видимости будет распо-

лагаться в пределах предварительно загруженных текстур. Следовательно, сигнал о готовности к построению изображения топологии в управляющую программу будет возвращаться практически мгновенно. Вместе с тем фоновый процесс продолжит выполняться параллельно. Как и ранее, реализуется загрузка «прогнозируемых» текстур: F3, F4, F5, C6, D6, E6, F6 – и освобождение занятой памяти видеокарты путем удаления текстур, «вышедших» из области прогноза: B2, C2, D2, E2, B3, B4, B5 (отмеченных на рис. 8, *в* серым цветом).

Заключение

Предлагаемые средства программной организации графической информации, описывающей многослойные и многокадровые изображения топологии СБИС, позволяют увеличивать размерности обрабатываемых схем, обеспечивая при этом дружественность программного интерфейса. Основным результатом их применения является высокое быстродействие отображения на экране фрагментов топологии.

Список литературы

1. Шлее, М. Qt 5.3. Профессиональное программирование на C++ / М. Шлее. – СПб. : БХВ – Петербург, 2015. – 928 с.

2. Райт, Р.С. OpenGL. Суперкнига / Р.С. Райт, Б. Липчак. – М. : Изд. дом «Вильямс», 2006. – 1040 с.

3. Programming Guide for DDS [Электронный ресурс]. – 2015. – Режим доступа : https://msdn.microsoft.com/en-us/enus/library/windows/desktop/ bb943991(v=vs.85).aspx. – Дата доступа : 23.11.2015.

4. MIP-MAP Filtering в процессе выполнения приложения [Электронный ресурс]. – 2015. – Режим доступа : http://www.ixbt.com/video/mip-mapping.html. – Дата доступа : 23.11.2015.

5. Статистика разрешений экрана 2000–2015 гг. [Электронный ресурс]. – 2015. – Режим доступа : http://www.fortress-design.com/display-resolution.– Дата доступа : 23.11.2015.

Поступила 02.06.2016

Объединенный институт проблем
информатики НАН Беларуси,
Минск, Сурганова, 6
e-mail: rom@newman.bas-net.by
yurafreedom18@gmail.com

V.I. Romanov, Y.Y. Lankevich

ORGANIZATION OF GRAPHIC INFORMATION FOR VIEWING THE MULTILAYER VLSI TOPOLOGY

One of the possible ways to reorganize of graphical information describing the set of topology layers of modern VLSI. The method is directed on the use in the conditions of the bounded size of video card memory. An additional effect, providing high performance of forming multi-image layout a multi-layer topology of modern VLSI, is achieved by preloading the required texture by means of auxiliary background process.