NB. Исходные проектные файлы находятся в директории lab1.

Цель лабораторной работы получить базовые навыки работы с пакетом моделирования ModelSim, уяснить понятия проекта ModelSim, получить знания о способах управления входными сигналами. Настроить и выполнить моделирование HDL кода в ModelSim-Altera.

Для получения данных базовых навыков необходимо будет исследовать следующие процедуры пакеты ModelSim:

•Создание проекта в ModelSim;

•Создание рабочих библиотек в ModelSim;

•Компиляция HDL файлов в рабочую библиотеку;

•Исполнение функционального моделирования в ModelSim при помощи команды force и при помощи HDL тестбенчей.

Проект в пакете ModelSim носит те-же свойства, что и проект в любой среде автоматизации проектирования. Проект хранит, список исходных файлов, порядок компиляции исходных файлов, настройки компилятора. Проект позволяет упростить совместное использование библиотек (каталоги скомпилированных устройств проекта) без копирования файлов в локальную директорию. Также проектный .**mpf** файл содержит все настройки находящиеся в глобальном **modelsim.ini** файле.

В лабораторной работе обучающемуся предстоит изучить весь маршрут создания и анализа проекта в пакете ModelSim:

1.Создание проекта (включая **work** библиотеку);

2.Добавление проектных и testbench файлов;

3.Создание библиотек ModelSim (необязательный этап, если библиотеки не используются);

4.Назначение существующих библиотек на физические директории (необязательный этап, если библиотеки не используются);

5.Компиляция проектных файлов;

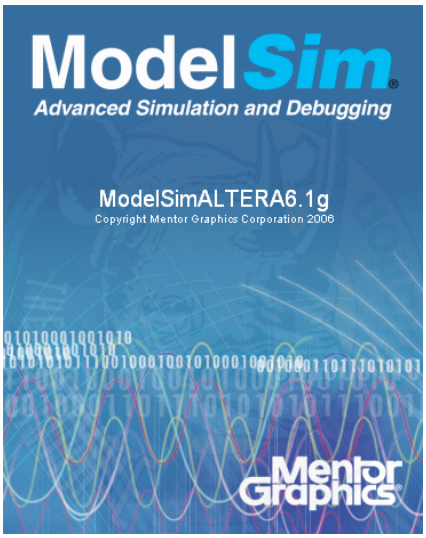
6.Запуск моделирования;

7.Создание конфигурации моделирования (необязательный этап)

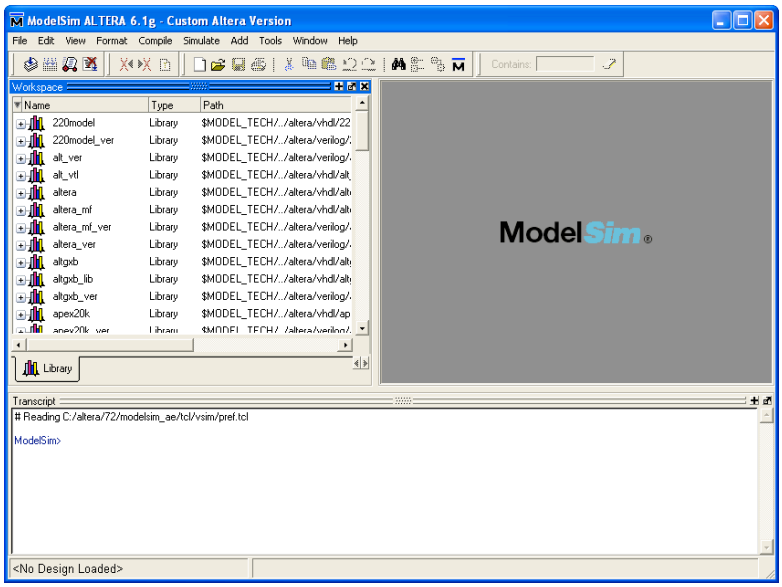
8.Собственно моделирование и отладка поведения анализируемого устройства.

1:Создание проекта и добавление исходных файлов

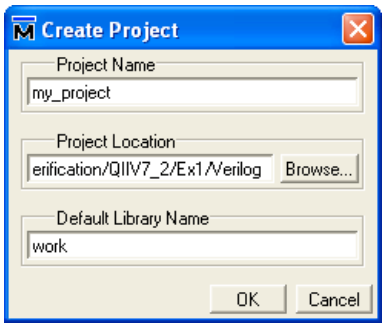
1. Запустите пакет **ModelSim**.



2. Закройте (кнопка **Close)** если всплыло окно **IMPORTANT Information**. Запустится основное окно пакета моделирования.



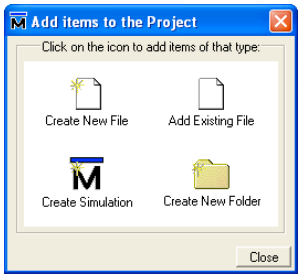
3. Создайте новый проект ModelSim. Из меню **File**, выберите **New** затем **Project**. Это вызовет диалоговое окно создания нового проекта **Create Project**.



4. Введите имя проекта (**my_project в примере)**.

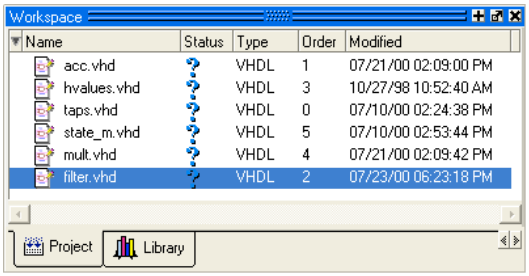
5. Укажите рабочую директорию **Project Location**. Для нового проекта рекомендуется создать новую пустую директорию. Для примера будем использовать директорию lab1

6. Имя библиотеки по умолчанию **Default Library Name** следует оставить так, как оно заведено - **work**. Нажмите **OK**. Появится диалоговое окно добавления файлов в проект **Add items to the Project**.



7. Добавьте исходные файлы в проект. Щелкните **Add Existing File**. Появится диалоговое окно **Add file to Project**.

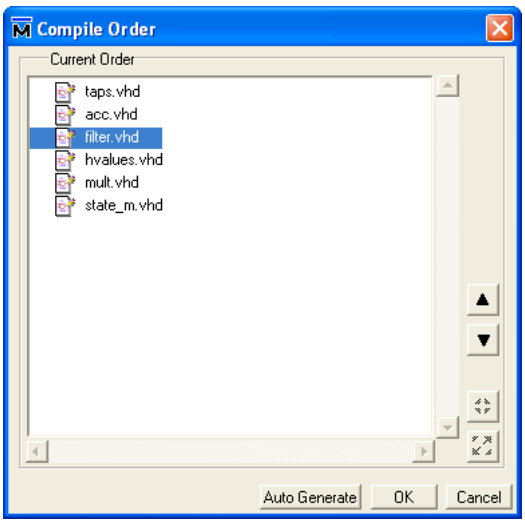
8. Щелкните кнопку **Browse** и выберите все проектные файлы в директории (**.vhd** файлы). Щелкните **Open**, затем **OK** для того чтобы добавить исходные файлы в проект. Закройте **(Close)** диалогове окно **Add items to the Project**.



*Проектные файлы будут перечислены в закладке **Project** окна **Workspace** помеченные знаком (?) в колонке **Status** потому-что они пока не откомпилированы.*

2: Компиляция исходных файлов

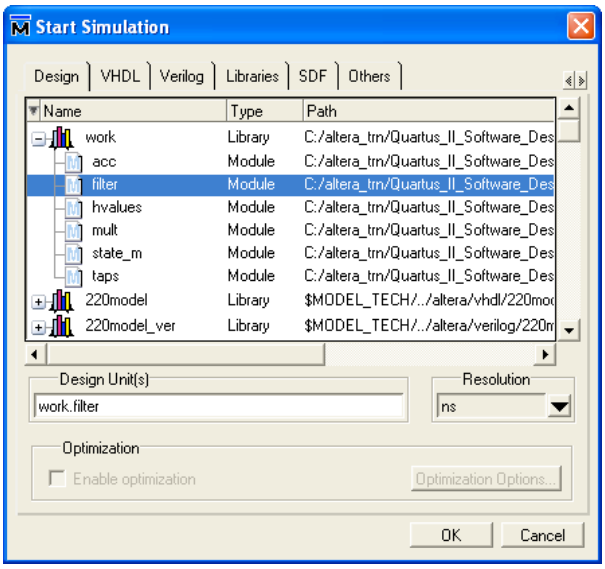
1. Установите порядок компиляции (файлы, используемые в верхнем уровне иерархии проекта, должны быть откомпилированы перед компиляцией файла верхнего уровня) и выполните компиляцию. Из меню **Compile** выберите пункт **Compile Order**.



2. Щелкните **Auto Generate** для автоматического упорядочивания файлов в соответствии с их иерархией для установления порядка компиляции.
3. Из меню **Compile** выберите пункт **Compile All** и выполните компиляцию файлов.

3: Запуск симулятора

1. Из меню **Simulate** выберите пункт **Start Simulation**.



*Появится диалоговое коно **Start Simulation**. Вкладка **Design** содержит перечень всех возможных к запуску на симуляцию первичных сущностей (entity, configuration, и **module).***

2. **!!!Обязательно установите минимальный квант моделирования**. Из раскрывающегося списка **Resolution** выберите соответствующий квант. Квант моделирования выбирается исходя из минимального временного периода используемого в проекте (в примере используются **ns)**.

3. Раскройте библиотеку **work**. Выберите сущность с именем **filter**.

4. Нажмите **OK**.

*Запустится симулятор и появятся некоторые дополнительные окна связанные с моделированием,это окна **Locals**, **Active Processes**, и **Objects**. Также откроется вкладка **sim** в окне Workspace. Окно **Transcript** должно отобразить, что все проектные сущности успешно загруженны для симулирования*

4: Интерфейс ModelSim

Перед продолжением моделирования, рассмотрим все окна интерфейса которые участвуют в процессе моделирования.

1. Из вкладки **sim** окна **Workspace**, дважды щелкните сущность **u2**.

*Сущность **u2** (код для блока **hvalues** проекта) откроется в окне **Source** исходного текста.*

2. Попробуйте открыть другие сущности из онка **Workspace** и поронаблюдайте за поведением среды.

3. Дважды щёлкните на сузщность **u3**, текст модуля **state_m** откроется в окне **Source**.

4. В окне исходного кода **Source**, найдите строчку кода 28 (**VHDL**), и щелкните на номер строки в левой стороне окна.

   Появится маленькая красная точка напротив номера строки кода.Данное обозначение указывает, на то что на этой строчке кода установлена точка останова(breakpoint).

   *Убедитесь в том, что это единственная активная точка останова. Если вы случайно установили другие точки останова, то щелкните на них правой кнопкой мыши и выберите пункт меню **Remove Breakpoint <line_number>**.*

5. Если окно **Objects** до сих пор не открыто, то в меню **View**, выберите пункт **Debug Windows** и выберите пункт **Objects**. Также можно набрать команду **view objects** в командном окне **Trancscript**.

6. Вернемся обратно в закладку **sim** окна **Workspace**, дважды щелкните на сущность **u2** и потом дважды щелкните на сущность **u1**.

   *Обратите внимание как окно **Objects** вместе с окном **Source**, изменяются в соответствии с выбранными сущностями в окне **Workspace**.*

7. Щелкните в окне **Objects** для его активации. В меню **View** выберите пункт **Filter** и выберите позицию **Internal Signals** для того чтобы отключить отображение внутренних сигналов в окне **Objects**.

   *Окно **Objects** теперь будет отображать только входные и выходные порты.*

8. В закладке **sim** окна **Workspace**, выберите сущность верхнего уровня с названием **filter**.

9. В окне объектов **Objects**, из меню **Add**, выберите пункт **Wave** и выберите команду **Signals in Region**.

   Откроется *окно временных диаграмм **Wave** содержащее все входные и выходные порты а также внутренние сигналы. Несмотря на то, что внутренние сигналы были отфильтрованы в меню **Object**, все сигналы были добавлены в окно **Wave** после исполнения предыдущей команды.*

10. Из меню **View**, выберите пункт **Debug Windows** и выберите команду **List**.

    *Откроется пустое окно **List**, без каких либо сигналов.*

11. Из закладки **sim** окна **Workspace**, выберите сущность верхнего уровня - **filter**, и перетащите его в окно **List**.

    *Все внутренние сигналы, входные и выходные порты отображены в заголовке онка **List**. Рассмотренная процедура демонстрирует другой способ добавления сигналов на окна **List** или **Wave**. Также можно перетаскивать сущности из окна **Objects**.*

12. В командном окне **Transcript** напечатайте команду **view process**.

    *Если окно активных процессов **Active Processes**еще не было открыто, то эта команда откроет его. Окно **Active Processes** отображает все процессы внутри сущности выбранной в закладке **sim** окна **Workspace** вне зависимости от их текущего состояния.*

13. В закладке **sim** окна **Workspace**, дважды щелкните на сущность **u4** для того, чтобы открыть проектный модуль **mult** в окне исходного кода **Source**, если он еще не открыт.

14. Раскройте сущность **u4** в закладке **sim** окна **Workspace** и выберите рдин из перечисленных процессов.

    *Процесс **line_13** (VHDL), подсвечен в окне **Active Processes**. Окно исходного кода **Source** отображает код, который генерирует данный процесс, стрелкой указывая на позицию с которой процесс начнет исполнение кода. Также, нужно отметить, что имя процесс совпадает с номером строки начала кода процесса в окне **Source**. Так будет происходить для каждого процесса, которому не назначено имя или метка в коде.*

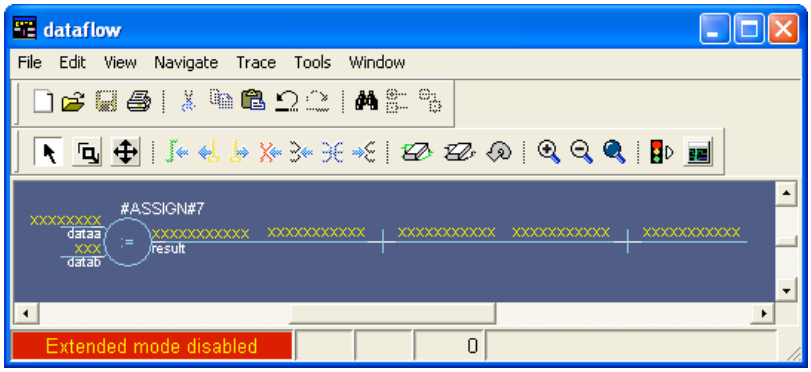
15. Из меню **View** выберите пункт **Debug Windows** и выберите команду **Dataflow**.

    *Откроется окно **Dataflow**. .*

16. В закладке **sim** окна **Workspace** выберите **filter**.

17. Активируйте окно **Objects** и включите отображение внутренних сигналов **Internal Signals** (отключенное в одном из предыдущих шагов).

18. Выберите шину **mult_to_acc** и перетащите её на окно **Dataflow**.



    *Окно **Dataflow** отображает как сигналы "проходят" по устройству.*

19. Пощелкайте на разные секции сигналов в окне **Dataflow**.

    *Отметим, что содержимое окон **Workspace**, **Objects**, **Active Processes** и **Source** изменяется в соответствии с тем, какая секция сигналов выбрана в окне **Dataflow**. В примере на иллюстрации, **dataa** и **datab** перемножаются для получения сигнала **result**. Это значение возвращается в сущность верхнего уровня **filter** как шина **mult_to_acc**.*

Шаг 5: Создание командного .do файла и управление моделированием

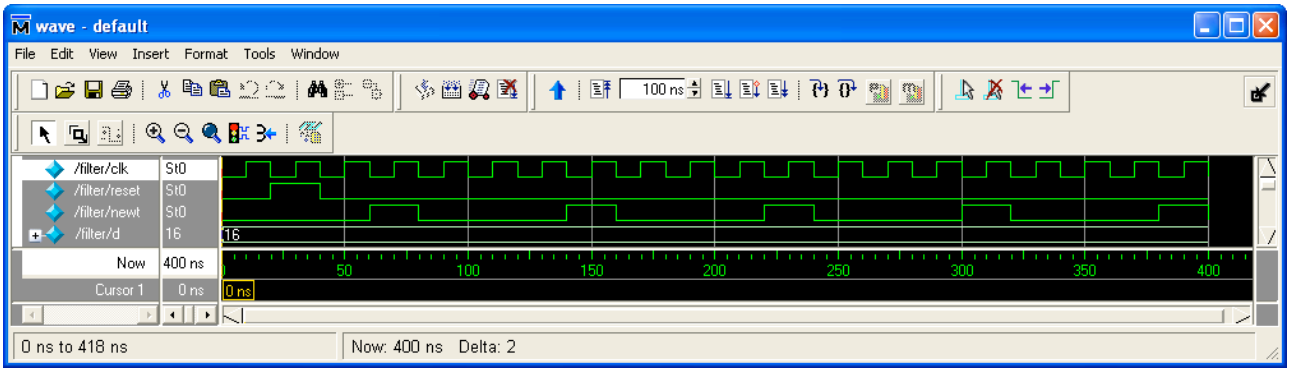
*Для запуска дальнейшего моделирования будет создан **.do** файл, содержащий информацию о воздействиях на входные сигналы.*

1. В меню **File** выберите пункт **New**, далее **Source**,и далее **Do**.

   *Новый неименованный **.do** файл будет создан и открыт в окне **Source**.*

2. Сохраните его с именем **stimulus.do**.

3. Запишите в этот файл команды **force** воздействия на входные сигналы, так, чтобы результирующая временная диаграмма совпадала с иллюстрацией (сигналу **d** присваивается десятичное значение 16):



*1: Необходимо только 4 строчки команд для задания сигналов **clk**, **reset**, **newt**, and **d**.*

*2: Для задания тактового сигнала **clk** можно воспользоваться следующей конструкцией:*

```
force /clk 0 0 ns, 1 {10 ns} -r 20 ns
```

*3: Ниже представлены заготовки для оставшихся 3 команд **force** (необходимо заполнить пробелы соответствующими значениями):*

```
force /reset 0  __ ns, 1  __ ns, 0  __ ns
force /newt 0  __ ns, 1  {__ ns} -r __ ns
force /d 10#__
```

4. Сохраните и закройте файл **stimulus.do**.

5. Из меню **Tools** выберите пункт **Execute Macro**. Выберите файл **stimulus.do** в диалогово окне и щелкните **Open**.

   ***Ничего не произойдет**! Так происходит потому, что исполнение данного файла лишь назначило события по изменению сигналов, однако для того чтобы увидеть изменения сигналов нужно запустить модельное время.*

6. Из меню **Simulate** выберите **Runtime Options**.

7. Измените значение **Default Run** на **100** ns, и щелкните **OK**.

8. Из меню **Simulate** выберите пункт **Run** и выберите команду **Run 100 ns**. Если даннаая команда недоступна, то введите команду **run @100 ns** в командном окне **Transcript**.

9. Переключитесь в окно **Wave**, или откройте его если оно до сих пор не открыто.

Временные диаграммы появляются по мере продвижения модельного времени. Для более детального изучения формы сигналов, есть возможность увеличения и уменьшения масштаба отображения.

10. В командном окне **Transcript** введите команду **run** для продолжения симулирования.

    *Команда **run** без параметров продивгает модельное время на 100 нс, значение по умолчанию, установленное в диалоге **Runtime Options**. Однако, на данном шаге симуляция остановится на точке останова установленной в сущности **u3** элемента **state_m** (28 строка **VHDL** кода). Симуляция приостанавливается, и в окне исходного кода **Source** открывается VHDL код **state_m** в котором стрелкой указано текущее место исполнения кода.*

11. Вернитесь обратно в окно **Transcript**,и введите команду **run -continue**.

    *Симуляция продолжится, переступив через точку останова, до окончания временного этапа второго запуска команды **run** из пункта №10. То есть до окончания 100 нс с точки модельного времени в котором была дана вторая команда **run**.*

12. В окне **Transcript**, выполните несколько раз команду **step**. Для вызов предыдущей команды из истории можно использовать стрелку вверх на клавиатуре.

    Эта команда исполняет одну строчку кода, последовательно попадая во все функции или подпрограммы вызываемые в исполняемой строке кода..

13. Теперь исполните команду "**run @300 ns"**. *(Обратите внимание на синтаксис написания команды)*

    *Данная команда продвинет модельное время симуляции до точки 300 нс с нулевого момента времени начала симуляции. Команда не будет завершена, потому что опять сработает точк останова в сущности **u3**. Не нужно путать данную команду с командой продвигающей модельное время нап **дополнительные** 300 на с текущей точки симуляции. Для выполнения такого продвижения нужно воспользоваться командой **run 300 ns**.*

    *Для завершения рассматриваемой команды также наберите команду **run -continue**.*

14. Из меню **Simulate** выберите пункт **Run** и выберите команду **Restart**.

15. Выключите позицию **Breakpoints** в диалоговом окне **Restart** и нажмите кнопку **Restart**.

    *Данная команда перезапустит моделирование и установит модельное время в 0,при этом все настройки окон интерфйеса будут сохранены, а все точки останова (breakpoints) в проектном элементе **state_m** будут удалены*

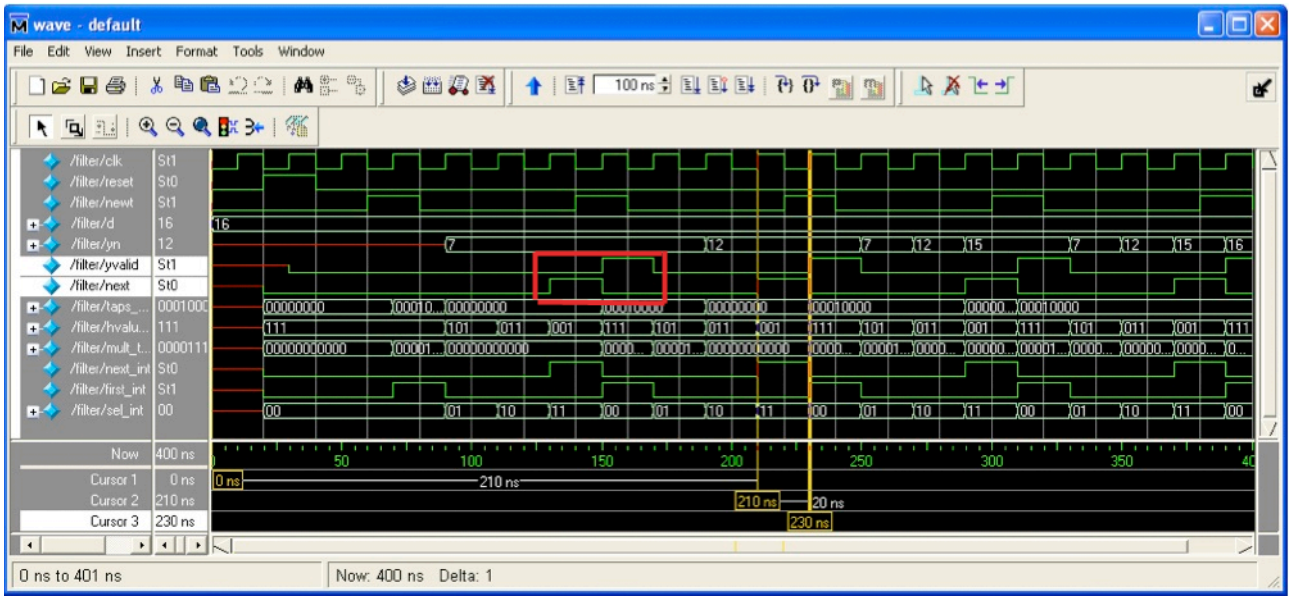
16. Снова выполните файл-сценарий **stimulus.do**.

    После перезапуска симуляции необходимо повторно применять все файлы-сценарии, сожержащие водействия на входные сигналы при помощи команд force.

17. Просимулируйте 400 нс модельного времени при попощи команды **run**.

Шаг 6: Анализ результатов симуляции

1. Щелкните правой кнопкой мыши на сигналы **/filter/d** и **/filter/yn** в окне временных диаграмм **wave** и установите параметр **Radix** в значение **Decimal**.



2. Из меню **Add** (или меню **Insert**, если окно **Wave** развернуто на весь экран), выберите позицию **Cursor**. Проделайте данную операцию дважды.

   *Новые курсоры будут отображены в окне **Wave**, внизу окна каждому куросору будет соответствовать своя аннотация. Одновременно управлять можно только одним курсором. Другие курсоры отображаются тонкими желтыми линиями, активный куроср отображается толстой линией. Для активации курсора необходимо щелкнуть на него левой кнопкой мыши. Внизу в аннотации отображается измерения времени между курсорами и между курсором и началом временной диаграммы.*

3. Из меню **View**, выберите пункт **Wave** далее пункт **Zoom** и выберите позицию **Zoom Range**. Если окно **Wave** развернуто на весь экран, последовательность команд будет следующей меню **View** -> **Zoom** -> **Zoom Range**.

4. В диалоговом окне **Wave Zoom** введите **0 ns** в поле **Start** и **400 ns** в поле **End**. Щелкните **OK**.

   *Такое-же действие можно произвести щелкнув правой кнопкой мыши в окне **Wave** и выбрав пункт контекстного меню **Zoom Range** или **Zoom Full**.*

5. Откройте окно **List**. Если окно **List** пусто, то перетащите на него сущность **filter** из окна **Workspace**.

   *В окне **List**, каждое изменение сигнала в процессе симулирования приводит к отображению новой строчки в таблица.*

6. В окне **List**, дойдите до модельного времени **90 нс**.

   *Рассмотрим строки +0, +1, +2, +3 на 90 нс. Окно **List** в дополнение к значениям времени отображает также значение дельта-циклы между шагами времени. Дельта-циклы это изменения сигналов которые происходят в один и тот же момент времени, но при этом каждый должен быть исполнен симулятором по отдельности. Например, процесс*

*порождает изменение сигнала "**a**" в момент времени **10**. Это будет считатся дельта-циклом **+0**. Если другой процесс, пораждающий сигнал "**b**" содержит в своем списке чувствительности сигна "**a**," тогда такой процесс будет исполнен в дельта-цикл **+1**. Таким образом, оба сигнала "**a**" "**b**" появляются в один момент времени **10**, но в различных дельта-циклах этого момента времени.*

*По умолчанию все дельта-циклы раскрываются в окне **List**. То есть отображается каждый индивидуальный дельта-цикл в каждый момент времени.*

7. Разверните окно **List** на весь экран. Из меню **Tools** окна **List** выберите команду **Window Preferences**.

8. На закаладке **Triggers**, в разделе **Deltas**, измените значение с **Expand Deltas** на **Collapse Deltas**. И нажмите кнопку **Apply** *(не нажимайте **OK**!!!)*.

   *Окно **List** теперь отображает только последний дельта-цикл для каждого момента времени. Это означает, что только отображается окончательное состояние всех сигналов в текущий момент времени.*

9. В разделе **Trigger On**, выключите опцию **Signal Change** и включите опцию **Strobe**. Установите значение **Strobe Period** в **100 ns** и значение **First Strobe** в **0 ns**. Щелкните **Apply**.

   *По умолчанию, каждое изменение значения приводит к тому что в окне **List** появляется новая строка. С включенной функцией **Strobe**, окно **List** будет отображать значения сигналов только в каждый моменты стробирования, то есть в нашем случае каждые 100 нс.*

10. Выключите **Strobe** и включите обратно **Signal Change**. Щелкните **OK**.

11. Рассмотрим временные шаги **110** и **120** в окне **List**.

    *Обратите внимание, что между этими двумя шагами времени единственным изменяющимся сигналом является сигнал, **clk**. Выберите столбец **/filter/clk**. Так как **каждое изменение любого сигнала** приводит к добавлению строки в окно **List** изменение сигнала clk привело к появлению в окне временной метки **120** несмотря на то что никакие сигналы больше не изменяются в проекте. Если вы анализируете проект работающей на высокой частоте, то у вас будут сотни строк в окне **List** отображающие только изменение тактового сигнала.*

12. Выберите сигнал **clk** в окне **List**.

13. Из меню **View** выберите пункт **Signal Properties**. Измените значение пункта **Trigger** со значения **Triggers line** на значение **Does not trigger line**. Щелкните **OK**.

    *Теперь изменения сигнала **clk** не приводят к появлению строчек в окне **List**. Обратите внимание что шаг времени **120** теперь больше не показывается.*

    *Опции **Strobe** вместе с опцией **Trigger** позволяют контролировать содержимое окна **List** и не перегружать его излишней детализацией.*

Шаг 7: Использование ModelSim для отладки проекта

1. В окне временных диаграмм **Wave** обратите внимание на сигналы **/filter/yvalid** и **/filter/nxt**. Есть разница, между видом временной диаграммы полученными в процессе симулирования и видом временной диаграммы представленной на иллюстрации в шаге 6 (выделена красным прямоугольником).
2. Используя полученные знания о пакете ModelSim и ваши знания языка описания аппаратуры HDL внесите необходимые изменения в исходный код, так чтобы временные диаграммы совпали.

   *Для того чтобы редактировать HDL код в окне **Source** необходимо развернуть это окно на весь экран,и в меню **Edit** отключить позицию **Read Only** для разрешения редактирования.*

   *Подсказка: Необходимо отредактировать строку кода №39 и № 41 в файле **acc.vhd**.*