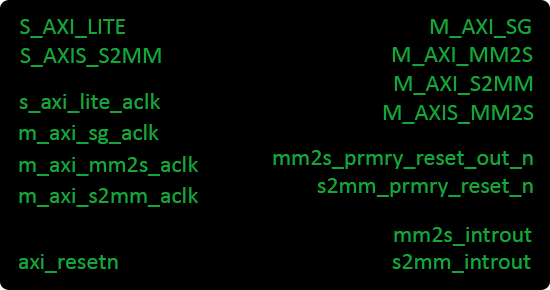
AXI Direct Memory Access



Мне кажется, для начала не помешает вспомнить, что означают некоторые префиксы, постфиксы и акронимы в именах интерфейсов и пинов.

Буква (префикс) M в начале имени интерфейса AXI означает Master.

Пример: M00\_AXI, M\_AXI\_MM2S

Буква (префикс) S в начале имени интерфейса AXI означает Slave.

Пример: S00\_AXI, S\_AXI\_LITE

Кроме того, в имени интерфейса указывается его протокол. Всего этих протоколов три:

AXI – для Memory Mapped интерфейсов.

Пример: M\_AXI\_MM2S

AXI\_LITE – облегченный (light-weight) протокол для Memory Mapped интерфейсов.

Пример: S\_AXI\_LITE

AXIS – протокол для Stream интерфейсов.

Пример: S\_AXIS\_S2MM

На блоке DMA также можно заметить аббревиатуры MM2S и S2MM. Их определения очевидны:

MM2S – Memory Mapped to Stream.

Пример: M\_AXI\_MM2S, M\_AXIS\_MM2S

S2MM – Stream to Memory Mapped.

Пример: S\_AXIS\_S2MM, M\_AXI\_S2MM

Теперь поговорим об именах пинов. Ниже я привожу список всех имеющихся на нашем DMA пинов.

Clock’и: s\_axi\_lite\_aclk, m\_axi\_sg\_aclk, m\_axi\_mm2s\_aclk, m\_axi\_s2mm\_aclk

Reset’ы: axi\_resetn, mm2s\_prmry\_reset\_out\_n, s2mm\_prmry\_reset\_n

Прерывания (Interrupts): mm2s\_introut, s2mm\_introut

Итак, начнем с Clock’ов. В именах Clock’ов часто можно встретить букву a непосредственно перед сокращением clk(clock).

Пример: s\_axi\_lite\_aclk

Эта буква означает, что Clock может работать в асинхронном режиме. Прошу отметить что “может” в данном случае не значит “должен” (если у кого-то есть вопросы по поводу синхронного и асинхронных режимов, об этом мы поговорим чуть позже).

В именах Reset’ов вызывает вопросы буква (постфикс) n стоящая в конце.

Пример: axi\_resetn

Она означает negative logic или по-другому active-low (если кто-то подзабыл что это такое, я расскажу попозже).

Что у нас там осталось? Прерывания, наверное.

В принципе с ними все понятно. intr означает interrupt (прерывание), out соответственно означает output (вывод/выход).

Пример: mm2s\_introut

Кажется, я собирался напомнить, что такое negative logic. Для начала представим два логических уровня (logic levels) – low(низкий) и high(высокий). В большинстве случаев low мы воспринимаем как логический ноль “0”, а high как логическую единицу “1”, но это не всегда так. Ничто не мешает разработчикам присвоить уровню low значение “1”, соответственно уровню high “0”. Иными словами существует два представления логики (двоичной): Positive logic (положительная логика) – когда low “0”, high “1”, и Negative logic (отрицательная логика) – когда low “1”, а high “0”. Иногда Positive logic называют Active-high, а Negative logic – Active-low.

Ладно, двигаемся дальше. Я хотел рассказать о синхронном и асинхронном режимах.

У DMA есть два clocking режима: синхронный и асинхронный (задействуется в параметрах блока, Customize->Enable Asynchronous Clocks) соответственно.

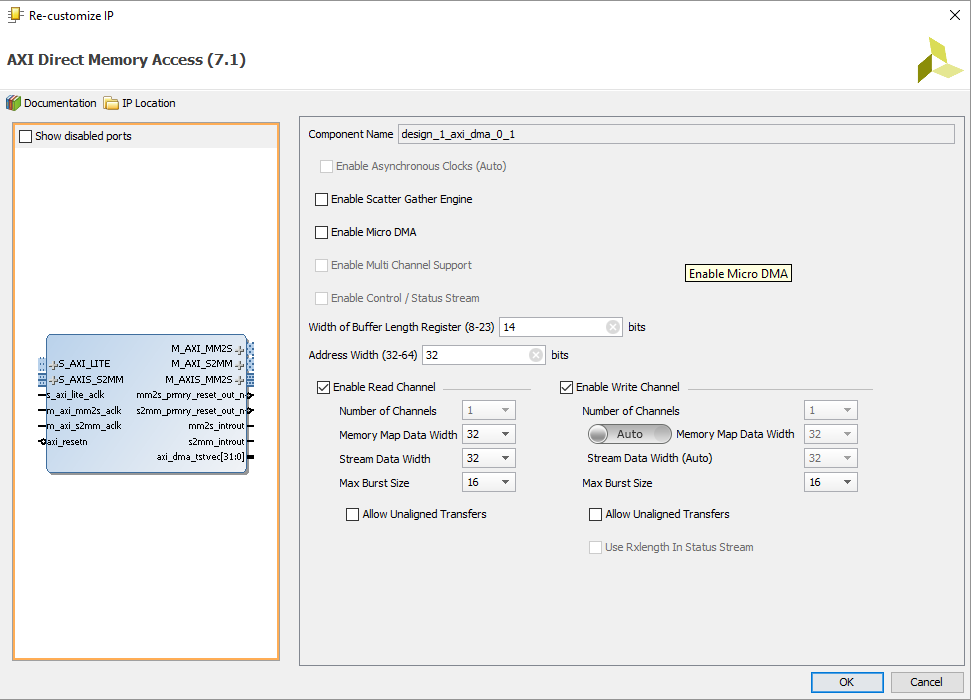
В синхронном режиме у нас есть только один clock domain (домен), иными словами все Clock’и (s\_axi\_lite\_aclk, m\_axi\_sg\_aclk (только если задействован Scatter Gather Engine), m\_axi\_mm2s\_aclk, m\_axi\_s2mm\_aclk) должны быть присоединены к одному и тому же источнику. В асинхронном режиме нам дается 3 (4, если задействован Scatter Gather Engine) независимых clock домена, то есть у каждого Clock’а свой.

Для того чтобы DMA могла работать в асинхронном режиме, необходимо соблюсти 2 условия: s\_axi\_lite\_aclk <= m\_axi\_sg\_aclk и

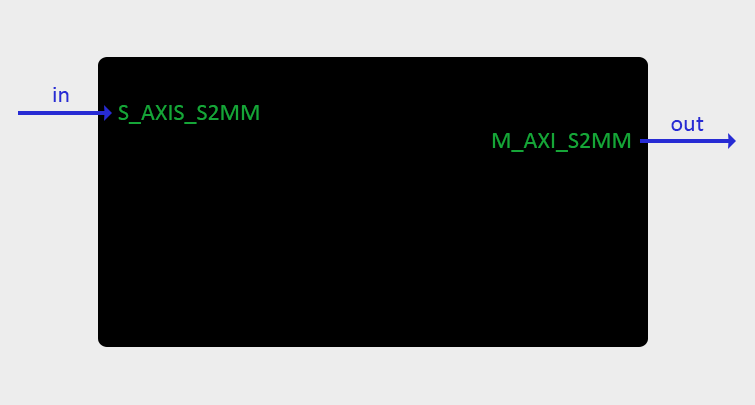
m\_axi\_sg\_aclk <= min(m\_axi\_mm2s\_aclk, m\_axi\_s2mm\_aclk)

Наверное, сейчас стоит уделить немного внимания тому, как DMA работает (забудем пока про Scatter Gather).

У DMA есть два канала Read и Write, если вы посмотрите на изображение ниже, то увидите два checkbox’а посередине (Enable Read Channel и Enable Write Channel) с помощью которых вы можете эти каналы задействовать.

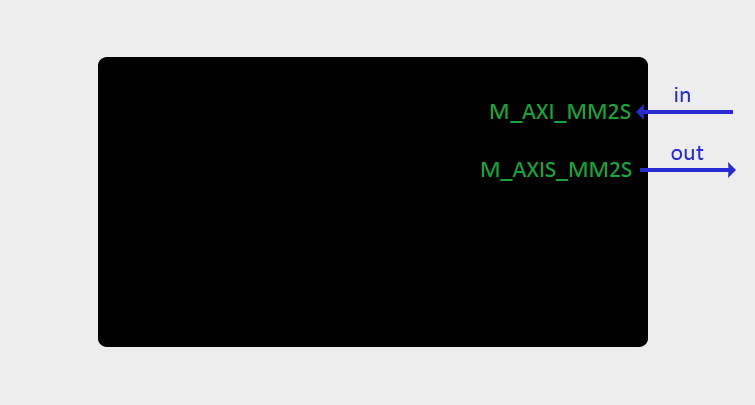


Сейчас оба канала задействованы, но мы пока отключим Read и представим следующую ситуацию – у нас есть некое ядро с Master Stream интерфейсом, через который мы собираемся посылать какие-то данные и мы бы хотели записывать эти данные в RAM.



На изображении можно увидеть, как данные принимаются на Slave интерфейсе S\_AXIS\_S2MM по протоколу Stream и отправляются дальше уже по протоколу Memory Mapped (записываются в память).

Отключаем Write и задействуем Read.



Смотрим, как DMA читает данные из памяти через Master интерфейс M\_AXI\_MM2S по протоколу Memory Mapped и отправляет их через Master интерфейс M\_AXIS\_MM2S по протоколу Stream.