**Research on implementation of dma/vdma.**

This document contains information and links to usefull documents/forums in which information and implementation of dma applications can be found. I use this to document information gathered from different sources to get a better idea and be able to structure the vdma game project from hardware design to displaying pixels on the screen using hdmi via vdma.

**1.** [**LINUX DMA FROM USER-SPACE**](https://xilinx-wiki.atlassian.net/wiki/spaces/A/pages/18842418/Linux+DMA+From+User+Space)

Article is based on an existing DMA example updated to xilinx release 2018.1. Zynq 7000 is software coherent when using HP ports, and hardware coherent when using ACP port.

Non cached memory used for performing DMA operations, its mostly a system issue based on size of data being transferred and amount of data being touched by the CPU software. Large datasets going through CPU caches may have performance effects on the cpu software while smaller may not. Cached memory may not be needed if software minimally touches the data of the DMA operation.

Linux drivers are kernel drivers. Design used by this article is hybrid. Meaning it uses kernel driver and user space application.

Xilinx provides many dma engines, which may be used in embedded applications. It also provides the DMA-driver(Linux soft DMA driver), this driver uses Linux DMA engine subsystem, and provides an API for the user made linux kernel driver. This article focuses on soft IP DMA for the PL.

The DMA proxy Design provides 2 simple examples which illustrate how to use the xilinx DMA driver for AXI DMA through the Linux DMA subsystem. And illustrate a simple method of DMA from user space.

The article mentions a [**User space only without kernel driver example**](https://github.com/ikwzm/udmabuf) which makes use of the linux UIO driver. With the challenge of allocating non cached contiguous DMA memory.

**2.** [**LINUX DMA FROM USER-SPACE**](https://xilinx-wiki.atlassian.net/wiki/spaces/A/pages/1027702787/Linux+DMA+From+User+Space+2.0)

Follow up on first article from 2023. Explains the build process and file structure together with some insights.

**3.** [**VDMA TEST PROCEDURE KERNEL**](https://xilinx-wiki.atlassian.net/wiki/spaces/A/pages/18842337/Linux+Soft+DMA+Driver)

Shows a kernel driver which uses the XILINX soft ip DMA driver for testing dma functionality in a sanitized environment meaning not real world test scenario.

**3. CROSS COMPILING KERNEL MODULES**

Cross compiling linux kernel modules for the armv7 architecture needs some setup the source code with the makefile can be found in GITROOTDIR/app/… this ….(will fix text soon !neefix)

* [LINK](https://adaptivesupport.amd.com/s/question/0D52E00006iHkd5SAC/how-do-i-build-a-kernel-module-seperately?language=en_US), Forum post explaining what to do but it does not generate module.symvers needed for symbol definitions or something
* located the petalinux project folder and go to :

cd <petalinux-project>/build/tmp/work-shared/zynq-generic-7z020/kernel-source

* Generate .config file for kernel and prepare modules

- make ARCH=arm xilinx\_zynq\_defconfig

- make ARCH=arm CROSS\_COMPILE=arm-xilinx-linux-gnueabi- modules\_prepare

* If Module.symvers is still missing, build **a single dummy module in-tree** to generate it(this will take some time)

- make ARCH=arm CROSS\_COMPILE=arm-xilinx-linux-gnueabi- modules

* **Once done export the directory(in-tree kernel source)**

- export KERNEL\_src=<petalinux-project-folder>/build/tmp/work-shared/plnx-zynqmp/kernel-source

* Now you can use KERNEL\_src instead of the full path to fill in the KERNEL\_DIR variable inside the Makefile.
* [Cross compilation Makefile](https://gist.github.com/Miouyouyou/6ee23eec681b21782b17ec8a45258b87" \l "file-makefile-shortversion-L6)

All the definitions for the words used

* Cache coherence

Software coherency is defined as the CPU software is required to perform cache maintenance functions to ensure that the memory system is coherent with the CPU caches.

* DMA-engine

hardware component facilitating DMA. Enabling transfer between memory and devices without constant CPU involvement.

* Linux DMA subsystem

The Linux DMA subsystem provides a unified API for device drivers to **allocate**, **map**, and **synchronize** memory for DMA operations. It abstracts hardware-specific details, allowing drivers to work across different architectures.

Section dedicated to usefull links

* [understanding the linux DMA subsystem](https://www.linkedin.com/pulse/understanding-linux-dma-subsystem-david-zhu-vrjdc/), David Zhu, Linkedin, May 14, 2025