802.11p - Optimierungsmöglichkeiten

Dominik Bayerl

3. Optimierungsmöglichkeiten und weitere Ideen

Zum derzeitigen Zeitpunkt bestehen noch offene Punkte zur weiteren Optimierung der Software, die aufgrund des beschränkten zeitlichen Rahmes des Projektes nicht mehr umgesetzt werden konnten.

Dabei handelt es sich größtenteils um Unschönheiten und Performance-Maßnahmen in der Software der High-CPU (Sniffer-Applikation), die jedoch nicht die grundsätzliche Funktion einschränken. Einige Verbesserungen sollen im Folgenden knapp skizziert werden.

3.1. Optimierung des IP-Stacks

Der IP-Stack wird immer dann benötigt, wenn ein Paket vom Wireless auf das LAN-Interface übertragen wird und umgekehrt. Beispielsweise ist es zur Verpackung der WLAN-Frames notwendig, diese in das RFtap-Format zu bringen, wobei dieses aus einem UDP-Frame besteht. Derzeit ist dies in der Datei `wlan_mac_high_sniffer/rftap.c` als Chainable-Funktionen implementiert. Dies bedeutet, dass die einzelnen Bestandteile des Ethernet-Frames stückweise konstruiert und dem Buffer hinzugefügt werden. Dadurch, dass der Buffer front-alloziert ist (d.h. es ist lediglich die Start-Adresse und die Länge des Buffers bekannt) ist es nicht möglich, die Header der Frames direkt dem Beginn des Buffers hinzuzufügen, da andernfalls der Datenbereich des Frames überschrieben werden würde. Um dies zu umgehen werden alle Header in einem separaten Buffer abgelegt und anschließend der Datenteil an das Ende der Header kopiert (Funktion `mpdu_rx_process()` in `wlan_mac_high_sniffer/wlan_mac_sniffer.c`). Der Kopiervorgang ist dabei potentiell ein Performance-Flaschenhals. Die Notwendigkeit eines einzelnen Buffers der den kompletten Ethernet-Frame enthält, ergibt sich durch die derzeitige Verwendung des Ethernet-Interfaces des Field Programmable Gate Array (FPGA) im einfachen Direct Memory Access (DMA)-Modus. Die tatsächliche Übertragung der Daten auf die Ethernet-Schnittstelle erfolgt anschließend ohne weitere Beteiligung der CPU durch das Ethernet-Peripheral.

Im erweiterten DMA-Modus bietet der Ethernet-Intellectual Property Core (IP-Core) die Möglichkeit der Datenübertragung zur Ethernet-Schnittstelle aus verschiedenen Speicherbereichen. Dieses Konzept wird bei Xilinx als "Scatter-Gather-DMA" (*grobe Übersetzung*: Verstreutes-Sammeln-DMA) bezeichnet [1]. Die Funktionsweise besteht darin, dass der DMA-Schnittstelle nicht mehr die Buffer-Adresse und deren Länge übergeben wird, sondern die Adresse eines sogenannten "Buffer Descriptors". Diese Datenstruktur besteht unter anderem aus der Buffer-Adresse und einem Längenfeld (siehe fig. 1). Sobald das DMA-Peripheral alle Daten aus dem im Buffer Descriptor referenzierten Buffers übertragen hat, wird ein Interrupt an die CPU ausgelöst.

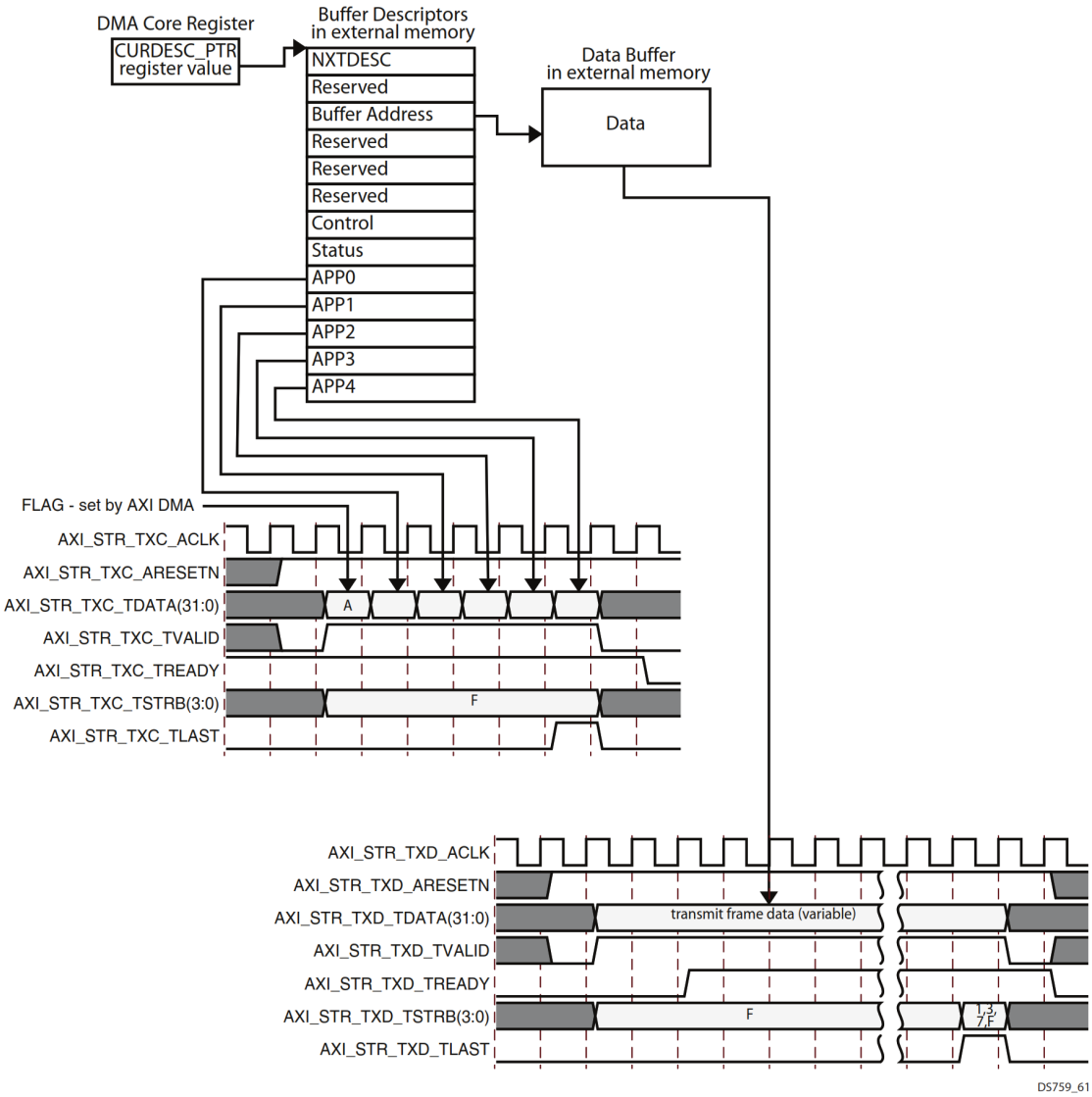

Abbildung 1: Xilinx DMA Buffer Descriptors[1].

Der Vorteil dieses Verfahrens besteht darin, dass eine zweite Indirektionsschicht eingeführt wird; dadurch ist es nicht mehr notwendig, dass sämtliche DMA-Daten sequentiell im Speicher liegen. Üblicherweise implementiert man dazu eine Single-Linked-List (First in First out (FIFO) Queue) aus Buffer-Descriptors, die nach jedem DMA-Interrupt weitergeschalten wird. Für den konkreten Fall der Ethernet-Frames ermöglicht dieses Verfahren, die einzelnen Header-Bestandteile unabhängig im Speicher ablegen zu können. Dadurch ist ein echter Zero-Copy Modus - also ohne Daten kopieren zu müssen - möglich.

3.2. Nutzung der Center Frequency Offset (CFO)-Estimates

Im Empfangspfad des WARPv3 existiert bereits ein Block zur Korrektur eventuell vorhandener Frequenzabweichungen der Sender bzw. des Empfängers (**TODO**: Referenz Aufbau Reference Design). Dabei wird mittels des Long training sequence (LTS)-Feldes über mehrere Frames die Frequenz des empfangenen Signals gegenüber der Center-Frequenz des gewählten Channels bestimmt und anschließend zur Korrektur der Fourier-Transformation der einzelnen Carriers genutzt.

In den meisten kommerziellen WiFi-Transceivern wird die Informationen anschließend verworfen, da sie für die darüberliegende Schicht (Layer 2, MAC Layer) nicht benötigt wird. Nicht so im WARP Reference Design: die geschätzte CFO wird durch die Low-CPU an die High-CPU in der Methode `mpdu_rx_process()` in der Datei `wlan_mac_high_sniffer/wlan_mac_sniffer.c` als Feld `cfo_est` der Struktur `rx_frame_info_t` übergeben. Gleichermaßen wird die Information bereits durch die Sniffer-Applikation in den RFtap-Frames an die Ethernet-Schnittstelle übertragen (*Flag 3, Frequency offset field is present*). Dies ermöglicht die Auswertung der CFO-Estimates beispielsweise im Wireshark eines angeschlossenen Computers.

Die Information ist besonders deswegen interessant, da sie (unter anderem) für zwei Anwendungsfälle genutzt werden kann, die im Folgenden dagelegt werden sollen:

3.2.1. Doppler-Effekt

Wie jedes elektromagnetische Signal unterliegen auch die 802.11p-WLAN Signale dem Doppler-Effekt. Dieser besagt, dass ein Signal der Frequenz f_S das von einem Sender S an einen Empfänger B derart übertragen wird, dass Sender und Empfänger eine Relativgeschwindigkeit $v_S - v_B = v \neq 0$ besitzen, beim Beobachter eine Frequenzabweichung $f_B = \frac{f_S}{\gamma} = f_S\sqrt{1 - \frac{v^2}{c^2}} \approx f_S\left(1 - \frac{v^2}{2c^2}\right)$ erfährt.

Diese Frequenzabweichung muss durch den Empfänger detektiert und kompensiert werden. Für die Anwendung WLAN wird dies durch die Korrektor der CFO übernommen. Hat ein Empfänger nun Kenntnis über den statischen CFO (bedingt durch ungenaue Oszillatoren) eines Senders, kann er durch die Messung des aktuellen CFO eine dynamische Frequenzabweichung bestimmen, bei der der Doppler-Effekt eine nicht unerhebliche Rolle

spielt. Für 802.11p bedeutet dies, dass zwei Fahrzeuge, die miteinander im Funkkontakt stehen, ihre gegenseitige Relativgeschwindigkeit ohne Zusatzhardware über die WLAN-Schnittstelle bestimmen könnten. Dies ermöglicht eine Reihe weiterer Funktionen, wie Notbremsassistenten, Adaptive Tempomaten und ähnliches.

3.2.2. PHY-Fingerprinting

In einer separaten Teilgruppe des Projektes wurde ein Verfahren zur Manipulation von Funknetzen, sog. MAC-Spoofing und Evil-Twin-APs evaluiert. Die Verfahren basieren darauf, dass die Merkmale die zur Identifikation der Funkteilnehmer verwendet werden, sehr leicht manipulierbar sind (MAC-Adresse bzw. SSID/BSSID). Für nicht weiter kryptographisch gesicherte Netzwerke (Offene WLANs) stellen diese Attacken ein erhebliches Sicherheitsrisiko dar, da dadurch eine Reihe weiterer Angriffe (Man-in-the-middle, Phishing, ARP-Spoofing, …) ermöglicht werden.

Das Mango-Board bietet gegenüber kommerzieller WLAN-Hardware die Möglichkeit, sämtliche Parameter der Funkübertragung zu erfassen, insbesondere auch die des physikalischen Layers, beispielsweise in Form der Center-Frequency-Offsets. Diese Parameter sind unabhängig von den gesendeten Daten, sondern werden ausschließlich durch die RF-Charakteristik der Hardware des Senders beeinflusst und eignen sich dadurch als Merkmal zur Identifikation eines einzelnen Sende-Moduls. Durch ein geeignetes Fingerprinting-Verfahren über mehrere verschiedene Merkmale (CFO-Estimates, Signal-Power, Noise-Power) kann dadurch eine Zuordnung anderer Merkmale (MAC-Adresse, SSID) zu einem physikalischen Sender geschaffen werden. Dadurch wird es ermöglicht, oben genannte Angriffe erkennen zu können, da im Falle eines vorhandenen Angreifers zwei verschiedene Sendemodule (mit unterschiedlichen Fingerprints) die selben High-Level Merkmale (MAC-Adresse, SSID) nutzen würden.[2]

3.2.3. Linux Kernel

Im Verlauf des Projekts zeigte sich, dass der Ansatz des 802.11 Reference Designs als Bare-Metal Software (d.h. ohne Betriebssystem) mehrere Schwächen besitzt: eine Iteration der entwickelten Software bedingt stets eine komplette Neuprogrammierung des FPGA-Designs. Desweiteren ist es nicht ohne weiteres möglich, Konfigurationsparameter (Channel, Baseband, …) während des Betriebs anpassen zu können. Diese Funktion wurde zwar rudimentär über eine UART-Konsole eingebaut, hat jedoch Schwächen in der Bedienbarkeit und Robustheit.

Weitere fehlende bzw. nur im Ansatz vorhandene Funktionen sind eine Debugging-Schnittstelle (`xil_printf()` über die UART-Konsole), ein Scheduler (Scheduling auf der CPU-High implementiert, non-preemptive round-robin mit Auflösung im Millisekunden-Berich) und die Möglichkeit zur Nutzung der von Xilinx bereitgestellten Peripheral-Treiber (insbesondere die Ethernet-Schnittstelle).

Diese offenen Punkte können durch den Einsatz eines Betriebssystems gelöst werden. Xilinx bietet bereits einen an die MicroBlaze-Architektur angepassten Linux Kernel [4] an. Zusätzlich sind für die meisten IP-Core Linux-Treiber vorhanden, die einfach integriert werden können [3].

Ungelöst ist dabei die Problematik der Treiber für benutzerdefinierte Peripherals - insbesondere für die *radio_controller*, die zentraler Bestandteil des WARP Reference Designs sind. Hier ist eine Anpassung der standalone-Treiber an die Schnittstelle des Linux-Kernels notwendig.

Literatur

[1] *logicore ip axi ethernet (v3.01a)*. 2012. URL: https://www.xilinx.com/support/documentation/ip_documentation/axi_ethernet/v3_01_a/ds759_axi_ethernet.pdf.

[2] *using rftap to detect mac spoofing*. 2016. URL: https://rftap.github.io/blog/2016/09/01/rftap-wifi.html.

[3] *xilinx wiki - linux drivers*. URL: http://www.wiki.xilinx.com/Linux+Drivers.

[4] *xilinx wiki - microblaze*. URL: http://www.wiki.xilinx.com/MicroBlaze#x-MicroBlaze%20Linux.

Abkürzungsverzeichnis