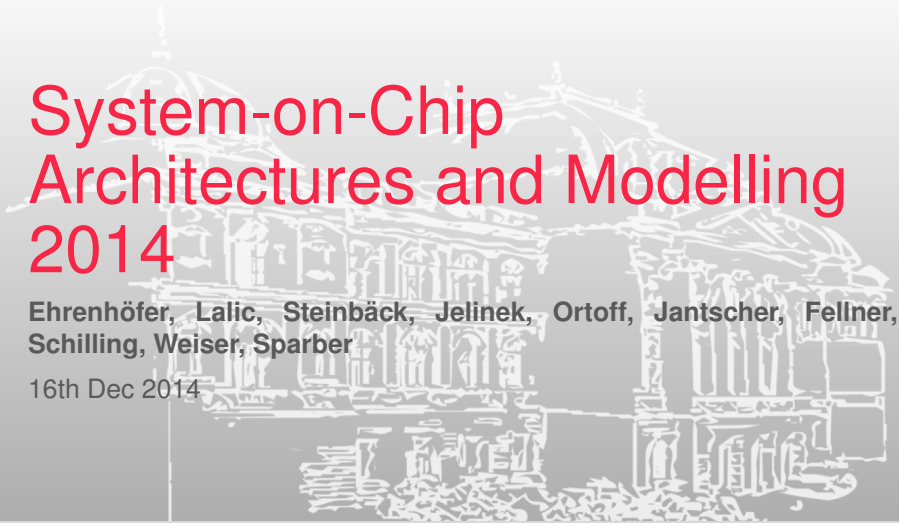


System-on-Chip Architectures and Modelling 2014

Ehrenhöfer, Lalic, Steinbäck, Jelinek, Ortoff, Jantscher, Fellner, Schilling, Weiser, Sparber

16th Dec 2014

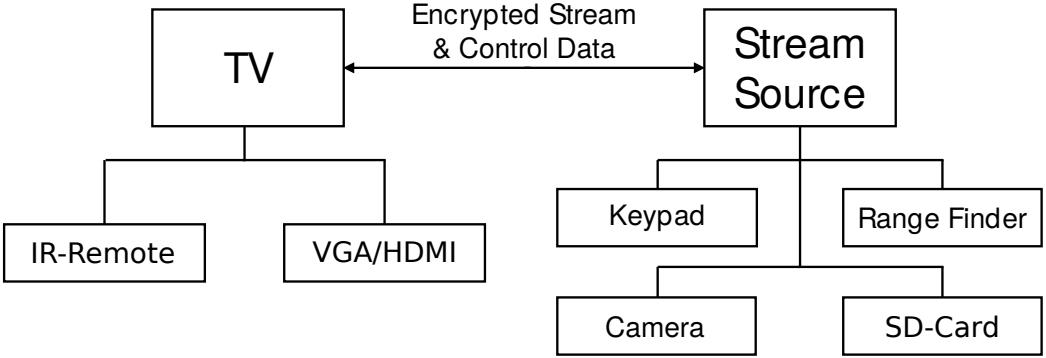
Contents

- Introduction
- Hardware
- Software
- Live Demo

Introduction

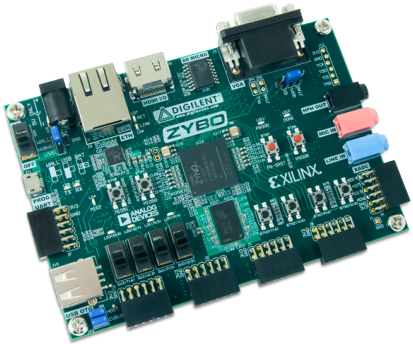
- Goals
  - Encrypted video streaming between two devices
  - USB camera support
  - Change stream source with IR-remote and keypad
  - Alarm system mode with range detector and camera
  - Two configurable devices (TV, Streaming Source)

System Architecture



Base

- Digilent ZYBO Zynq-7000
  - 650 MHz dual-core Cortex A9 running Petalinux
  - VGA/HDMI ports and IP-Cores
  - Ethernet port
  - Six PMOD connectors
  - 512 MB DDR3
  - Reprogrammable logic equivalent to Artix-7

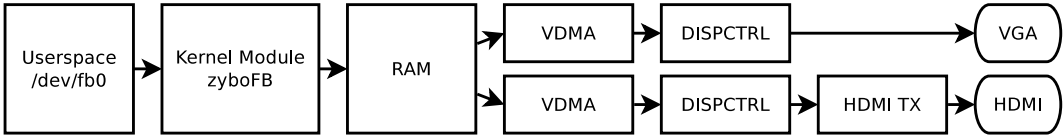


Hardware Components

Output using a Framebuffer

- Implemented a linux framebuffer driver
  - Two independent framebuffer devices (`/dev/fb0` and `/dev/fb1`) or
  - Mirror mode to output the same image on both
  - Resolution change using `fbset` (up to 1080p)
  - Textoutput using a virtual console
- VGA and HDMI have same interface
- Hardware already implemented in base_design from Digilent, Inc.
- Standalone version worked out of the box

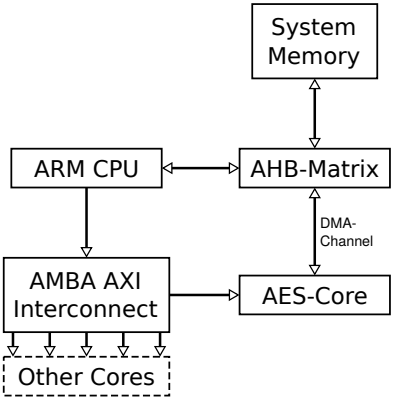
Block Diagram



- Kernel Module
- AXI Video Direct Memory Access (`axi_vdma`)
- AXI Display Controller (`axi_dispctrl`)
- HDMI Transmitter (`hdmi_tx`)

AES Hardware Core

- AES hardware-core provides only decryption mode
- KEY/IV setup over AMBA-interface
- Data transfer from/to core over AMBA interface or DMA controller (AXI4 streaming interface)
- Using DMA saves CPU resources

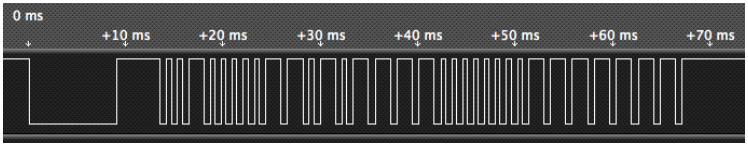


AES Kernel Module Driver

- Provide two kernel module drivers presented as character devices (`/dev/aes-state` and `/dev/aes-ofb`)
  - AES-STATE: read to save or write to restore state of AES core
  - AES-OFB: writing/reading to de-/encrypt data with *output feedback mode*
- Easy use with programs by piping data into character devices
- Using DMA controller writes a chunk of states into memory
- CPU loads a state and XOR it with plain-/ciphertext

IR-Sensor

- Direct interface to GPIO
- AXI-GPIO module inclusive IRQ to interface hardware
- Kernel Module
  - Measures time between pulses/spaces
  - Implements a LIRC device `/dev/lirc0`



LIRC Integration

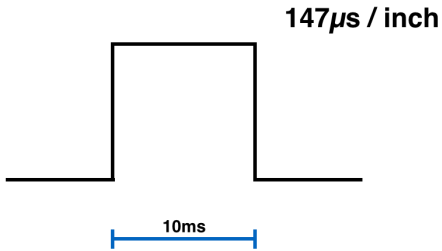
- **L**inux **I**nfrared **R**emote **C**ontrol (LIRC)
- Use `irrecord` to learn remote commands
- Integration
  - LIRC daemon started on TV
  - `irexec` used as client program to invoke other scripts

- Example button configuration

```
begin
  prog = irexec
  button = KEY_CHANNELUP
  config = /sd/bin/TV/ir_invoke.sh ch+
end
```

Range Finder

- 3 different interfaces
  - Analog interface
  - Serial interface
  - **Pulse-width interface**



Range Finder

- Hardware
  - Measure time between edges
  - 4-times averaging in HW
  - AXI-interface to Processing System (PS)
- Software
  - Implement character device `/dev/range-finder`
  - Reading returns distance in centimeters

USB/Camera

USB

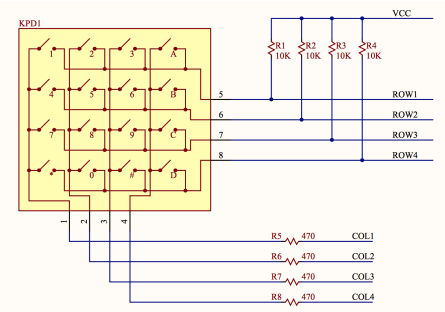
- USB directly attached to ARM-core
- Three modes: Host, Device, OTG

Camera

- Logitech HD Webcam C270
- Linux-UVC compatible
- Implements the V4L2 API

Keypad

- 16-button PMOD keypad
  - 4-pin column enable
  - 4-pin row state

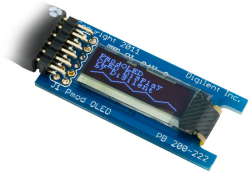


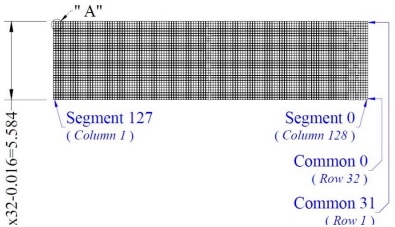
Keypad

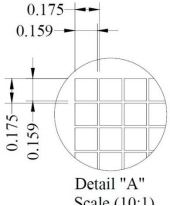
- Hardware
  - Column-select multiplexed
  - Reads row state pins, edge detection of key press
  - Pressed keys stored in 1024x8 FIFO
  - Reading FIFO triggers FIFO value to be sent over AXI-Bus
- Software
  - Implement character device `/proc/Keypad`
  - Reading returns all keys stored in FIFO

ZYBO 0led

- Integration
  - Directly attached to PS
  - Problems with SPI-bitbang
  - Modifying kernel → **PANIC!**

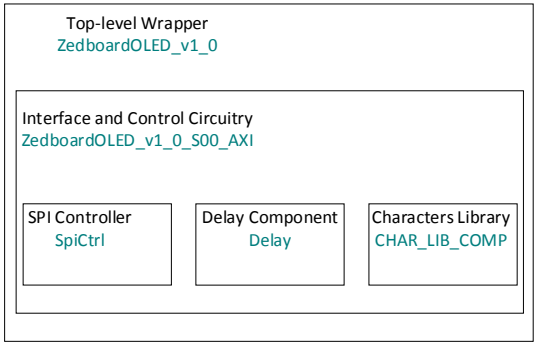






ZYBO 0led

- Hardware
  - A&M University Qatar
  - Controller overview
  - Issues with integration
- Software current status
  - Under development



Audio Codec

- SSM2603 codec on board
- Configuration with I2C
- PCM sound data with I2S
- Headphone out, Line in and Mic in

Implementation and Outcome

- Using Analog Digital, Inc. I2S core and kernel modules
- Adding of top level module to connect I2C and I2S (from Blackfin)
- Compilation of ALSA
- Kernel settings and device-tree modifications
- Debugging . . .

Result: Output of the audio codec is only sometimes right

Software

Root File System

- `BOOT.BIN` on first partiton of SD-card which contains *initramfs*
- *initramfs* mounted on `/`
- *initramfs* contains operating system
- runs `initsd.sh` on boot
  - Mounts partitions `/sd` and `/sdboot`
  - Executes `initsmartstream.sh` from SD-card
    - Initializes all kernel modules
    - Network configuration
    - Start our application based on GPIO-switch value

Network Streaming Protocol

Requirements

- Connection-less
- TV and Streamy independent
- Multiple TVs on same stream

The way to go

- TCP via netcat as a starting point
- UDP via netcat (baaad)
- UDP via C-application
- (S)RTP. . .

Streaming with TCP (`netcat`)

TCP

- ✓Reliable
- ✓Ordering of data packets
- ✗Connection-oriented
- ✗Slower than UDP

Streaming

- Different ports (format, content)
- Starting new `netcat` session for every transmission

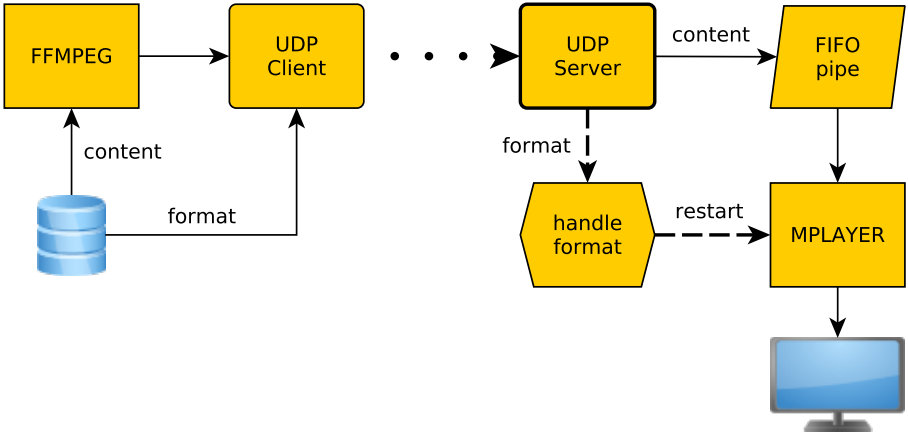
Streaming with UDP

- IP Multicast 239.0.0.1
- Header contains format & resolution
- Multi-threading with ringbuffer
- ✓Fast
- ✗Packet loss
- ✗Packets out of order

UDP packet layout

4 byte	*sequence no*
16 byte	*IV*
16 byte	*Header*
16 byte	*Data*
…	...
16 byte	*Data*
Total 1024 bytes	

UDP Streaming



Video Codecs

- MJPEG (∼1990)
  - Each frame compressed as JPEG image
  - Digital cameras, IP cameras, and webcams
  - Simple, inefficient
- MPEG-2 (1999)
  - Broadcast, DVD
  - Good compression, medium CPU effort
- H.264/MPEG-4 AVC (2004)
  - Broadcast, internet, Blu-ray
  - High compression, high CPU effort
  - Lack of available IP-core

Encoding

- Encoding on streaming source
  - No appropriate HW solution available (too large/expensive)
  - Encoding in software is time intense
- Avoid encoding:
  - MPEG-2 pre-encoding for video-files
  - MJPEG directly available from webcam
- `ffmpeg` used to transmit video to streaming channel
  - Real-time *encoding* option used

```
ffmpeg -re -i <input_clip.mpg> -vcodec copy -acodec copy -f mpegts - | nc ${TVHOST}
${STREAMPORT} -c
```

Decoding

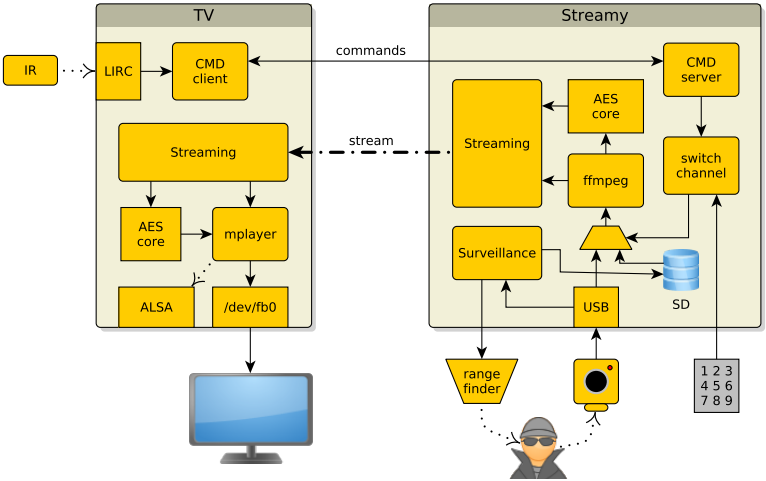
- Decoding on streaming target
  - Decoding is bottleneck (not bandwidth)
  - Use MPEG-2
- Mplayer used to receive video from streaming-channel
  - Plays videos in real-time
  - Output to frame-buffer device
  - Use cache to avoid lags

```
nc -l -p ${STREAMPORT} | mplayer -vo fbdev2:/dev/fb0 -demuxer +mpegts -tsprobe 1
-cache 1024 -
```

Command Channel

- TCP via netcat
- Fully `bash`-script-based
- HMACs & nonces for authenticity
- Commands
  - `ch0 ... ch9`
  - `on/off`
  - `...`

Big Picture


TV
IR
LIRC
CMD client
Streaming
AES core
mplayer
ALSA
/dev/fb0
commands
stream
Streamy
CMD server
AES core
Streaming
ffmpeg
switch channel
Surveillance
USB
SD
range finder
1 2 3
4 5 6
7 8 9

Stream Channel

- Stream source is selected by channels and subchannels
- TCP or UDP (Multicast)-Stream possible
- Can be encrypted if needed
- Source depending configuration is sent to the TVs
  - Resolution
  - Encryption
  - Format type
  - ...
- Fully `bash`-script-based

Live Demo