2023
DESIGN AND VERIFICATION™
DVCON
CONFERENCE AND EXHIBITION
TAIWAN

Building a Virtual Driver for Emulator

CHEN CHIH-CHIANG

ANDES TECHNOLOGY

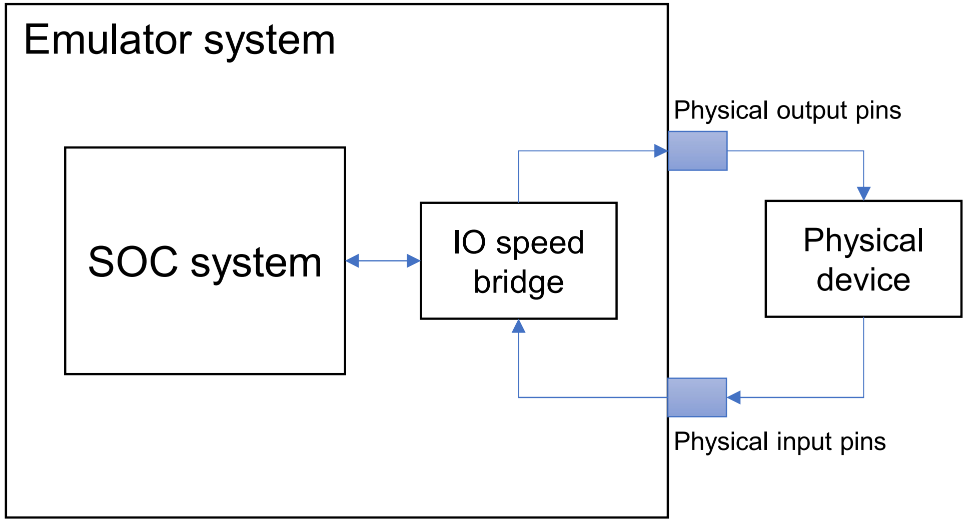
accellera SYSTEMS INITIATIVE

Agenda

- External device on the emulator
  - Physical device emulation
  - Virtual device emulation
- What is a Virtual Driver?
- Virtual System Overview
  - Hardware protocol virtualize.
  - Bind to the host environment.
- System emulator
  - GSFIFO Input Stream modeling
- Virtual Device Examples.
  - Example: DMI
  - Example: Mass storage device
  - Example: Network
- Experiment result
  - Full platform virtualization
- Future scope

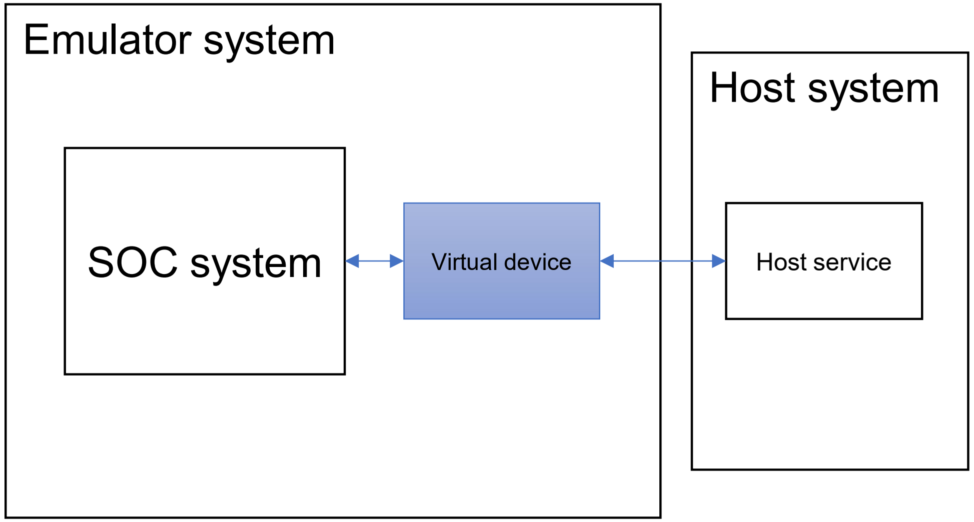
External device on the emulator (1)

- Physical device emulation
  - IO can connect with physical input/output pins
  - Hard to connect many devices with physical pins
    - For example:
      - UART 2 pins + ethernet MAC 16 pins + JATG 4 pins + SD card 10 pins = 32 pins for 1 system emulation
      - One emulator system can simulate up to 8 systems at the same time. Total needs 32*8 = 256 pins on different pins



External device on the emulator (2)

- Virtual device emulation
    - All protocol emulation complete on the virtual device
    - Protocol service action with the host system, For example:
        - UART send a character to the host system and output on the console
        - SD card handle block memory act as file IO on the host
    - Easy to expand emulation system device without physical limitation.


Emulator system

SOC system

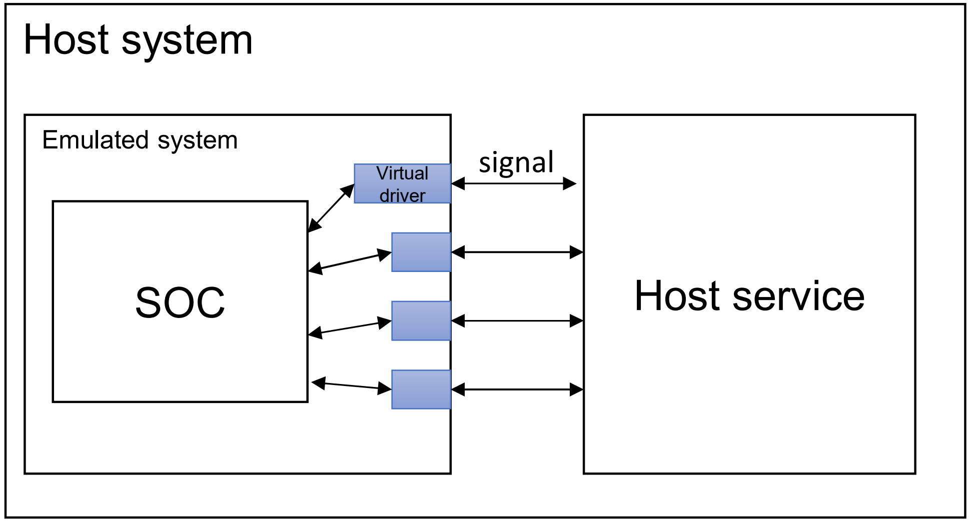
Virtual device

Host system

Host service

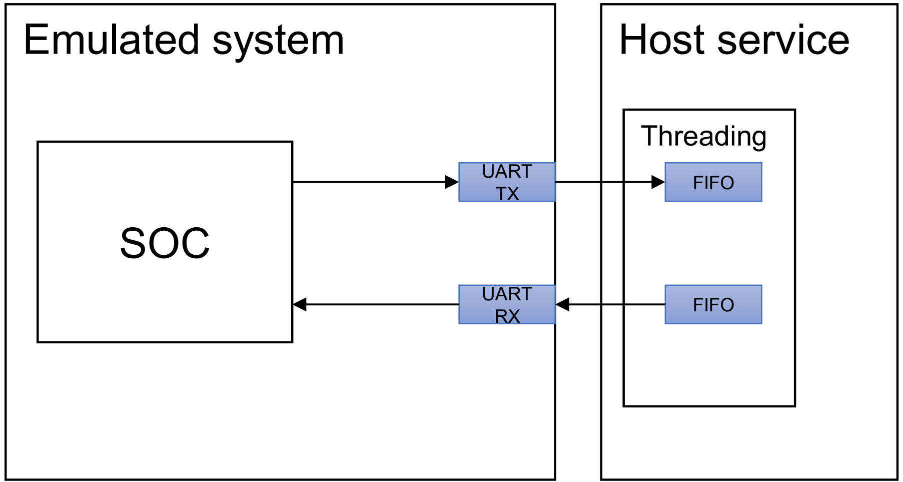
What is Virtual Driver?

- Simulated or emulated system issues the signal to an external host and receives the signal from the external host.



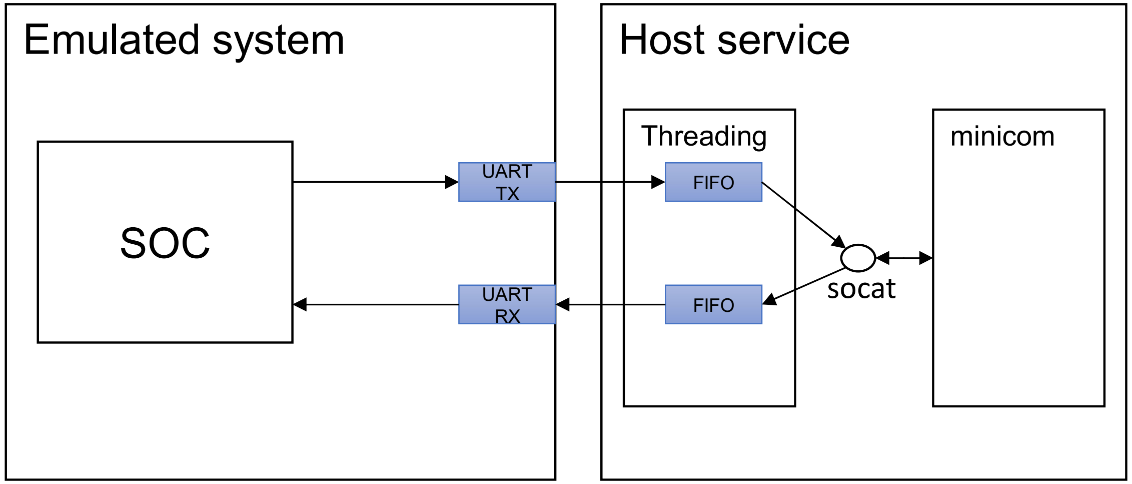
Hardware Protocol Virtualize

- The signal issues modeling as output FIFO
- The signal receives modeling as input FIFO.
- Host service holds another thread to handle input/output FIFO.
- For example:
  - UART TX
  - UART RX



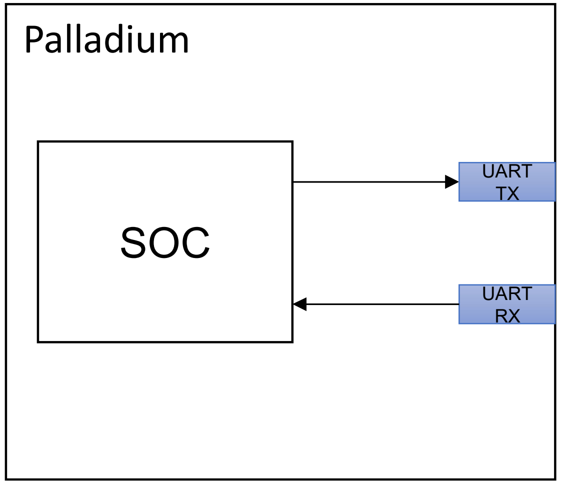
Bind to the host environment

- The input/output FIFO can be binding as a system signal channel
  - SOCK: use send/recv function to connect FIFO.
  - FILE IO: use read/write function to connect FIFO.
  - UNIX PIPE: use Linux redirect input/output function to connect FIFO.
- Example
  - Using socat (PIPE) or ncat to bind the UART to minicom for UART virtualize



Palladium Emulator

- Palladium support Systemverilog DPI
  - Virtualize the **protocol** to DPI-call
  - High-speed communication between emulator and system service

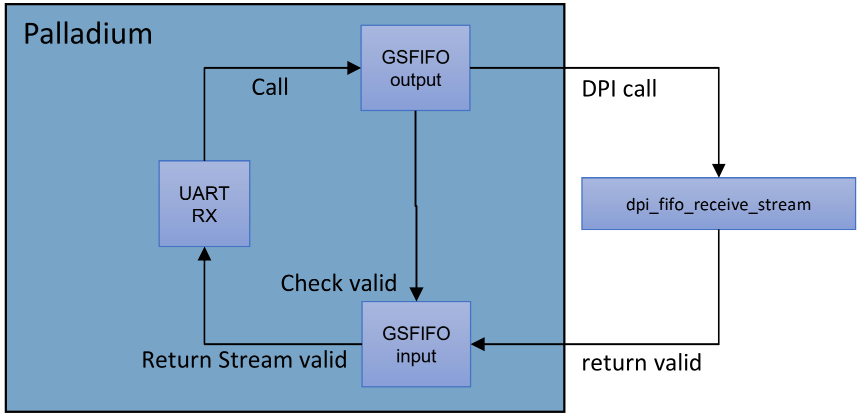


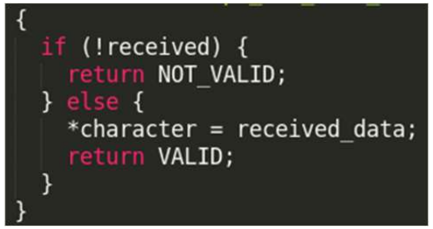
void dpi_fifo_send_stream (input byte unsigned tx_data);

int dpi_fifo_receive_stream (output bit [7:0] rx_data);

Palladium GSFIFO Input Stream modeling

- Receive data from virtual drive need do context switch
  - Very slow and emulation overhead is very big.
- Palladium/Protium support asynchronous communication between the emulator and host system.
- the GSFIFO INPUT STREAM (GSF_IS) help to virtual driver receiver as input stream on Palladium/Protium.

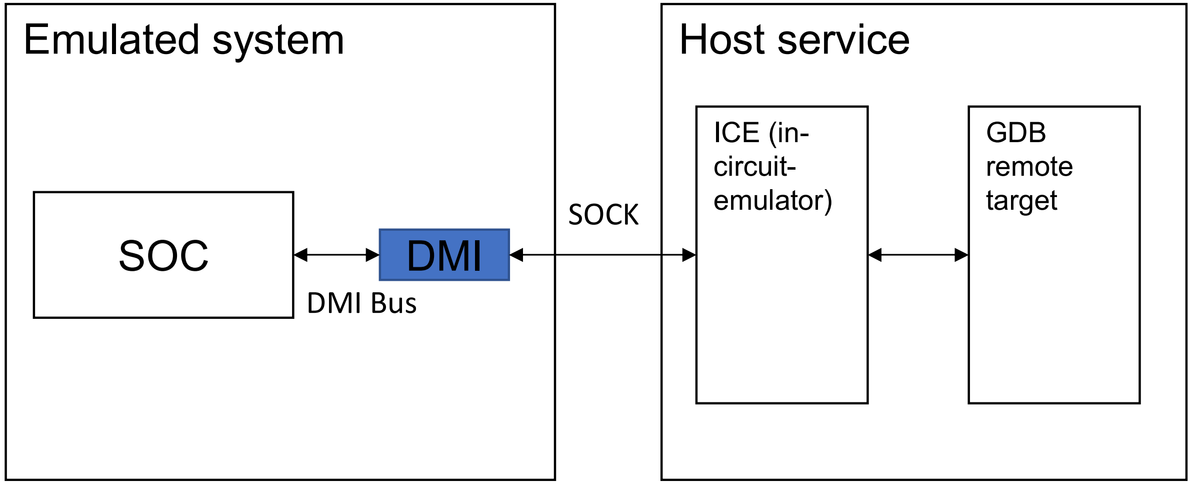




```
{
  if (!received) {
    return NOT_VALID;
  } else {
    *character = received_data;
    return VALID;
  }
}
```

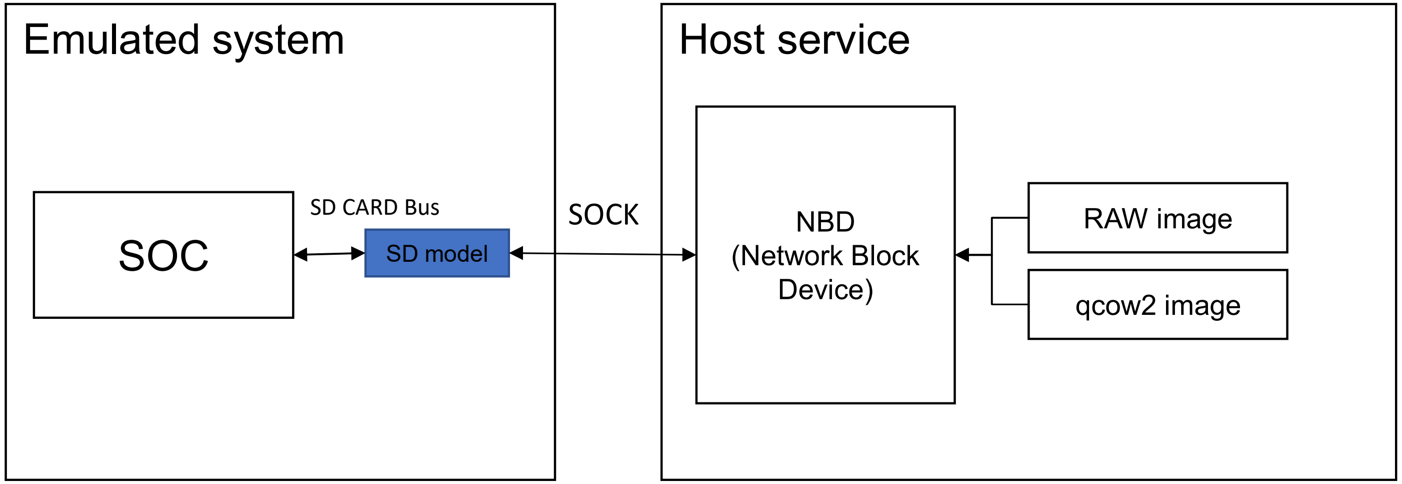
Virtual Device Examples. (DMI)

- DMI (Debug module Interface)
  - The RISCV-DMI virtual driver to communicate between the ICE and Andes Core
  - Direct access debug module to speed up the debug flow and handle memory access



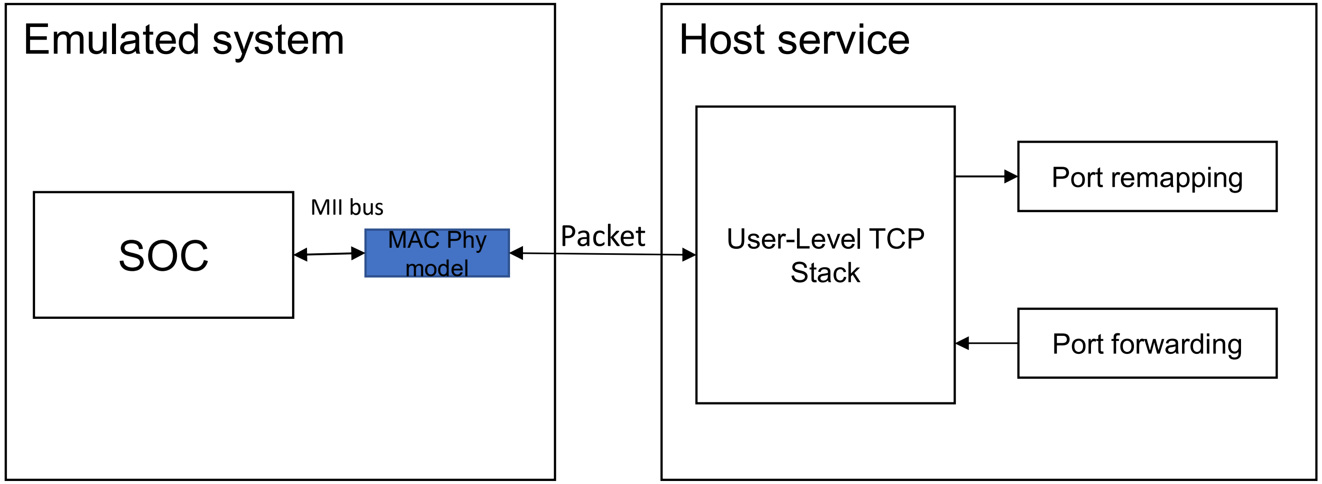
Virtual Device Examples. (Mass storage device)

- SD card can be modeling as block device
  - Using NBD protocol to model as virtual block device
  - NBD service can mount RAW image or qcow image



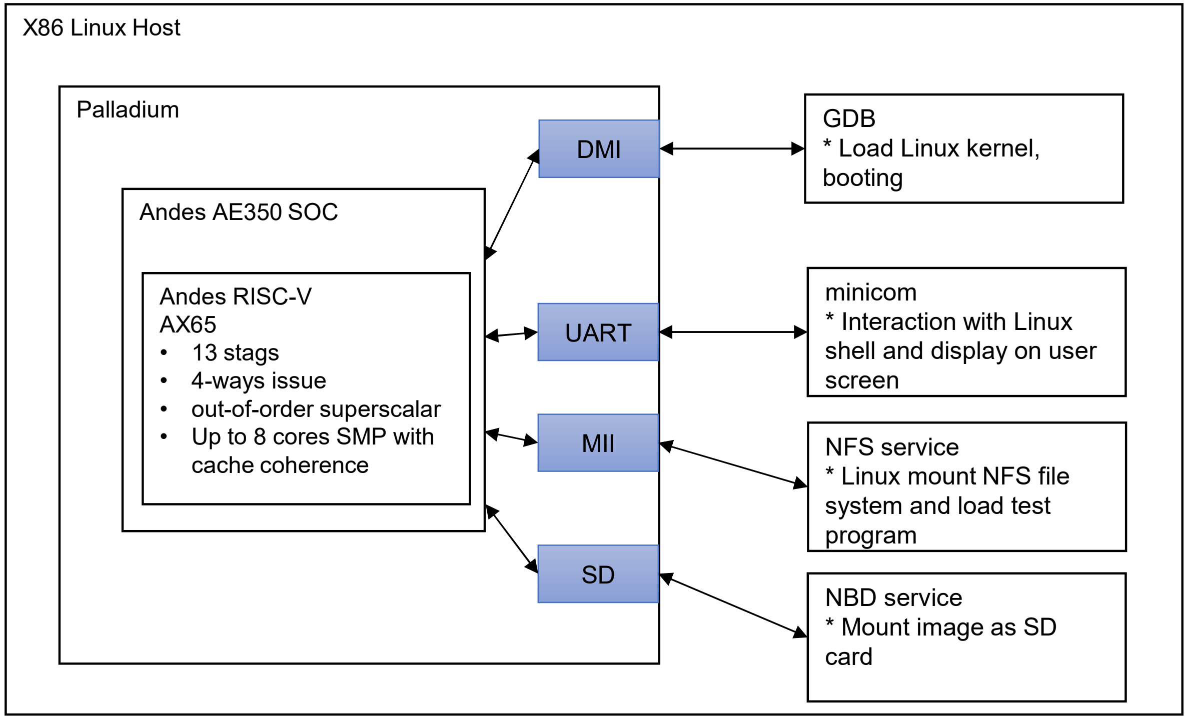
Virtual Device Examples. (Network)

- Model the MAC MII protocol as packet FIFO
- Exchange data with User-level TCP stack and issue packet to remote target and receive the new packet.



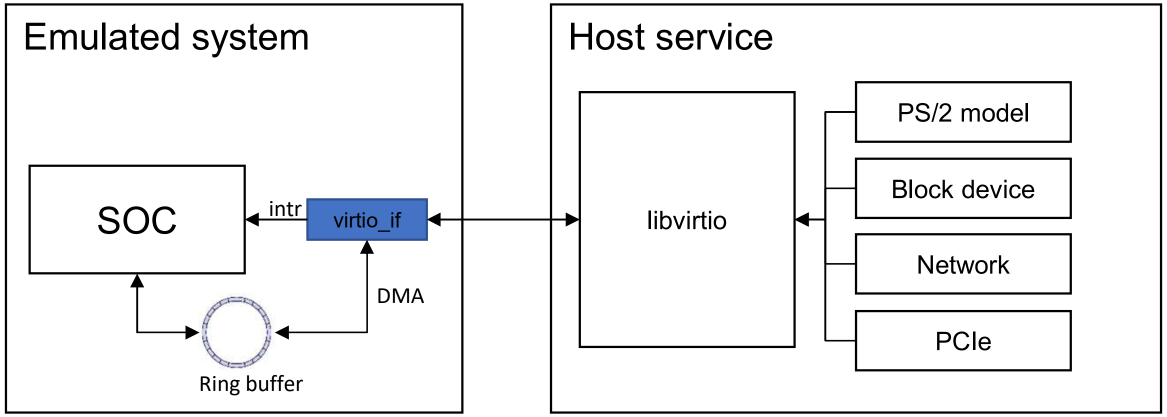
Full Platform Virtualization

- Andes AE350 platform virtualization
  - DMI <-> GDB
  - UART <-> minicom
  - MII <-> User-level TCP stack
  - SD <-> NBD
- Linux runtime
  - All system emulation, like on the FPGA emulation
  - Most SOC input/output is virtually connected to the host service.
  - Profiling the performance log.


X86 Linux Host
Palladium
Andes AE350 SOC
Andes RISC-V
AX65
• 13 stags
• 4-ways issue
• out-of-order superscalar
• Up to 8 cores SMP with cache coherence
DMI
UART
MII
SD
GDB
* Load Linux kernel, booting
minicom
* Interaction with Linux shell and display on user screen
NFS service
* Linux mount NFS file system and load test program
NBD service
* Mount image as SD card

Future Scope

- Full virtualized device
  - Virtual IO on Linux kernel
  - Virtualize PS/2 device
    - Mouse
    - Keyboard
  - Virtual block device, network, PCIe
- More complex system
  - Validate the Andes Core performance.
  - Emulate the real chip live in the virtual world.


Emulated system
SOC
intr
virtio_if
DMA
Ring buffer
Host service
libvirtio
PS/2 model
Block device
Network
PCIe