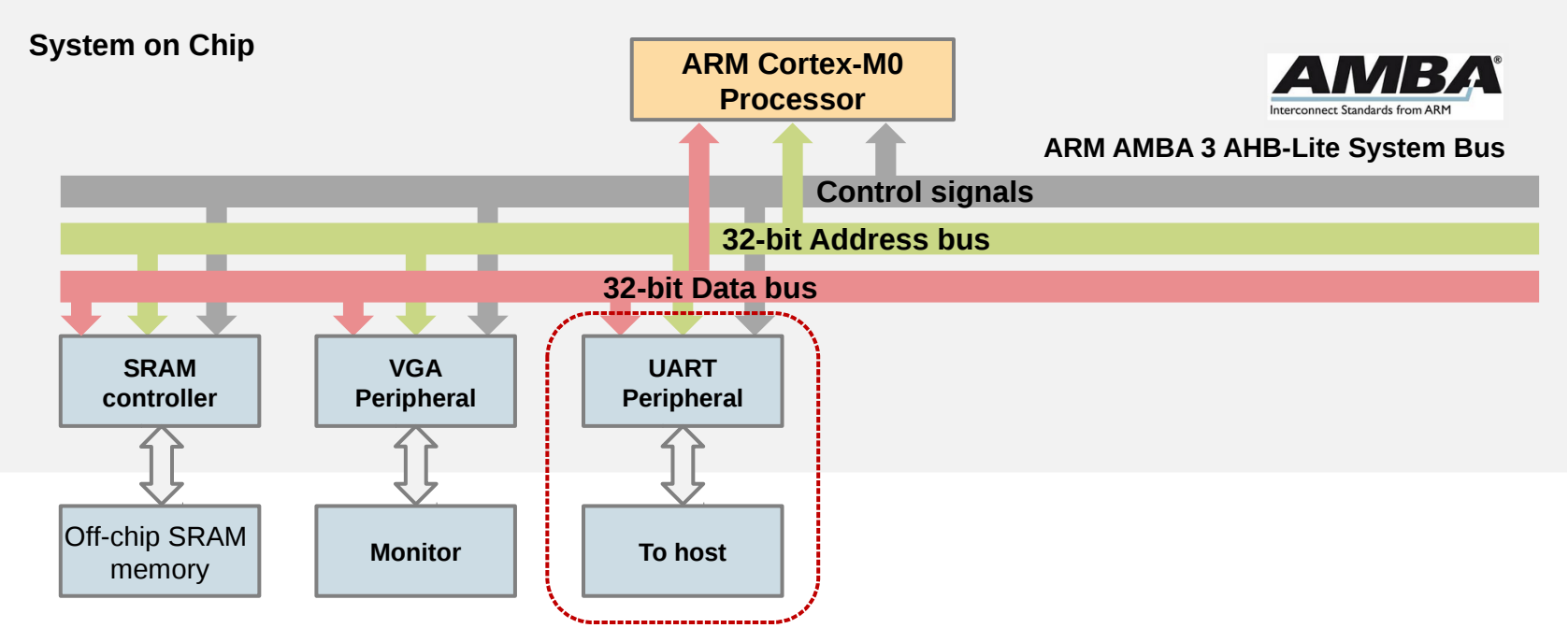
Design and Implementation of an AHB UART Peripheral

Module Overview

- Learn about the UART interface;
- Design and implement an AHB UART peripheral;
- Program peripherals using assembly;
- Lab Demonstration.



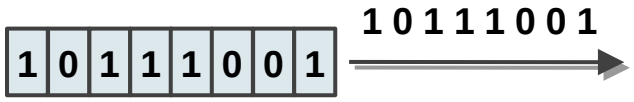
Module Syllabus

- Serial Communication
- Serial Vs. Parallel Communication
- Synchronous and Asynchronous Serial Transmission
- UART Protocol
- Character-Encoding Scheme
- AHB UART Implementation
- First In First Out (FIFO)
- AHB UART FIFO Implementation
- Lab Practice

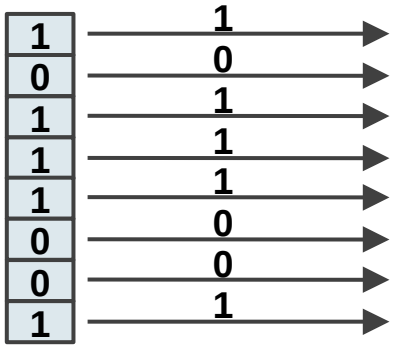
Serial Communication

Serial Communication

- Serial communication
  - Transmits data one bit at a time, in a sequential fashion;
  - In contrast to parallel communication, in which multiple bits are sent as a whole;
  - Commonly used for long-haul communication, modems and non-networked communication between devices.
  - Example are UART, SPI, I2C, USB, Ethernet PCI Express etc…



Serial communication



Parallel communication

Serial Vs. Parallel Communication

- Cost and weight
  - Less cost and weight, as less wires and smaller connectors are needed compared with parallel communication;
- Better reliability
  - Parallel communications may introduce more clock skews, as well as crosstalk between different wires;
- Higher clock rate
  - Due to the higher reliability, serial communication can be clocked in a higher frequency, hence increase the throughput;
- On the other hand, the conversion between serial and parallel data may consume extra overheads.

Types of Serial Communication

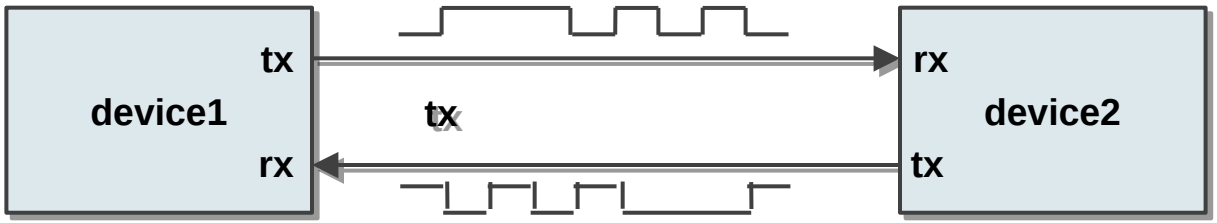
- Synchronous Serial Transmission
  - A common clock is shared by both the sender and the receiver;
  - More efficient transmission, since one wire is dedicatedly used for data transferring;
  - More costly since an extra clock wire is required.

- Asynchronous Serial Transmission
  - The sender does not have to send a clock signal;
  - Both sender and receiver agree on timing parameters in advance;
  - Special bits are added to synchronize transmission.

Universal Asynchronous Receiver/Transmitter (UART)

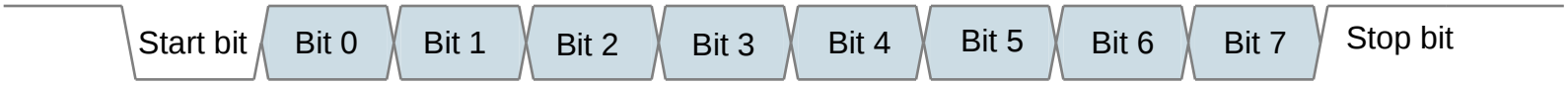
UART Overview

- Universal Asynchronous Receiver/Transmitter (UART)
  - Asynchronous communication, no clock wire required, pre-agreed baud rate;
  - Separated transmission and receiving wires;
- UART communication
  - Converts data from parallel to serial ;
  - Sequential data is transferred through serial cable;
  - Receives the sequential data and reassembles it back to parallel.



UART Protocol

- Data transfer starts with a starting bit, by driving logic to low for one clock cycle;
- In the next 8 clock cycles, 8 bits are sent sequentially from the transmitter;
- Optionally, 1 parity bit can be added to improve transfer reliability.
- In the end, the data wire is pull up to high to indicate finishing of the transfer.



Transfer one byte without parity bit

Transfer one byte with parity bit

Character-Encoding Scheme

- What data is sent to the UART in order to display a text?
- ASCII (Characters are coded in American Standard Code for Information Interchange)
  - Encodes 128 characters
    - 95 printable characters, such as ‘a’, ‘b’, ‘1’, ‘2’ …
    - 33 non-printing control characters, e.g. next line, back space, escape…
  - Can be represented by 7 bits, commonly stored as one byte for storage convenience.
- UTF-8 (UCS Transformation Format—8-bit)
  - Derived from ASCII from 2007;
  - Variable-width encoding scheme, which avoids the complication of endianness and byte order marks;
  - Widely used for the World Wide Web;
  - Compatible with the original ASCII.

ASCII Encoded Characters

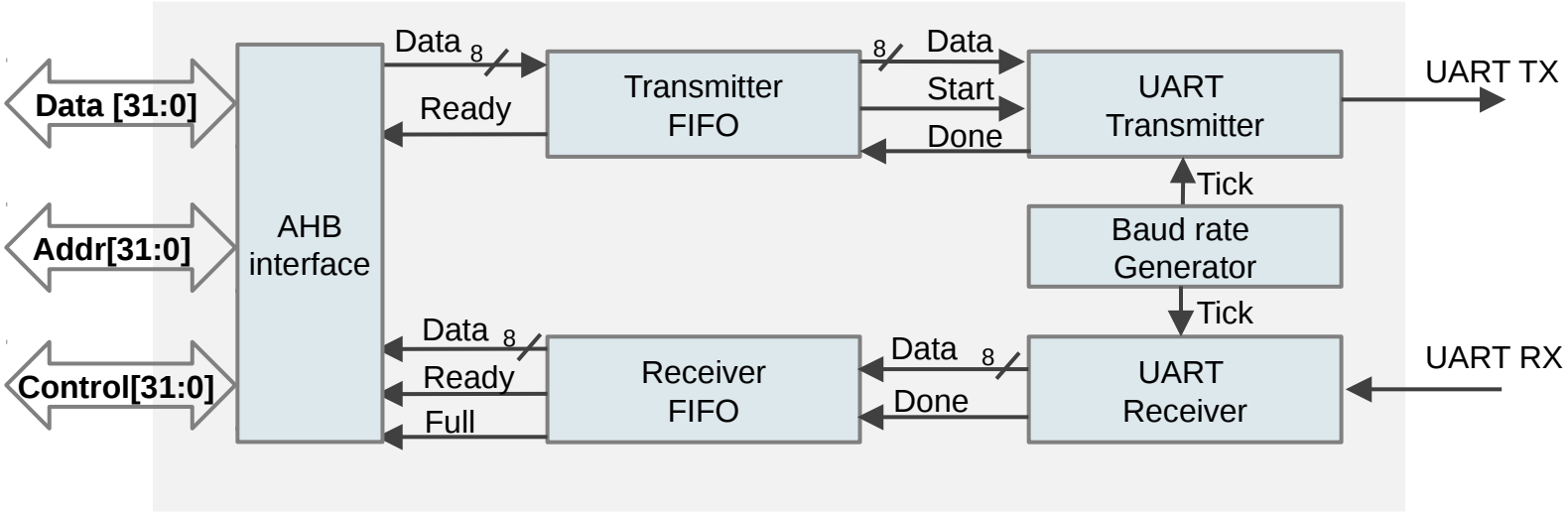
- For example, the table below lists some frequently used characters coded in ASCII. The full table can be found in Reference 2.

Hex	Character		Hex	Character		Hex	Character
0x30	0		0x41	A		0x61	a
0x31	1		0x42	B		0x62	b
0x32	2		0x43	C		0x63	c
0x33	3		0x44	D		0x64	d
0x34	4		0x45	E		0x65	e
0x35	5		0x46	F		0x66	f
0x36	6		0x47	G		0x67	g
0x37	7		0x48	H		0x68	h
0x38	8		0x49	I		0x69	i
0x39	9		0x4A	J		0x6A	J
…			…			…	

AHB UART Hardware Implementation

AHB UART Peripheral

- In our design, the UART peripheral consists of:
  - UART transmitter, receiver;
  - Transmitter FIFO and receiver FIFO;
  - Baud rate generator;
  - AHB bus interface.



UART Transmitter/ Receiver

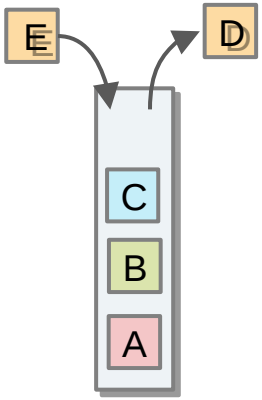
- UART transmitter
  - Read data (in byte) from the transmitter FIFO;
  - Converts a single byte data to sequential bits;
  - Send bits to the tx pin, clocked in a fixed rate provided from the baud generator.

- UART receiver
  - Receive the sequential bits from the rx pin, using the clock generated from the baud generator;
  - Reassemble the bits to a single byte;
  - Write the received byte to the receiver FIFO.

First In First Out (FIFO)

- First In First Out (FIFO) refers to a data buffer that outputs its earliest input data, such as data queue
- In contrast, Last In First Out (LIFO) is a buffer that outputs its latest input data, e.g. program stack;
- Synchronous FIFO
  - Same clock is used for both reading and writing
- Asynchronous FIFO
  - Different clocks are used for reading and writing.

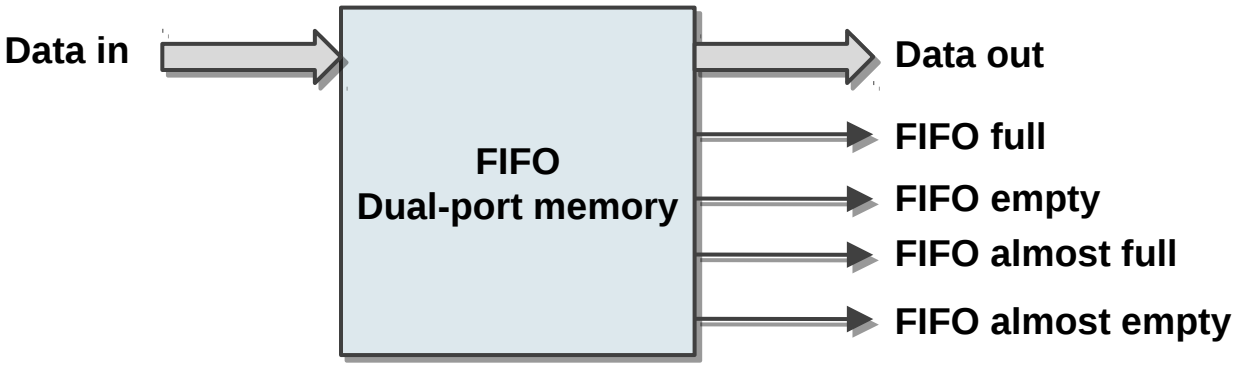
Queue: First In First Out



Stack : Last In First Out

First In First Out (FIFO)

- A FIFO is usually implemented on a dual-port memory, as one port is for reading and the other one for writing;
- Additional flags are used to indicate the status of the FIFO
  - FIFO full: all memory space is used, hence no more data can be written;
  - FIFO empty: no data in the memory, thus no more data can be read;
  - Some FIFOs also provide half full/ empty signal.



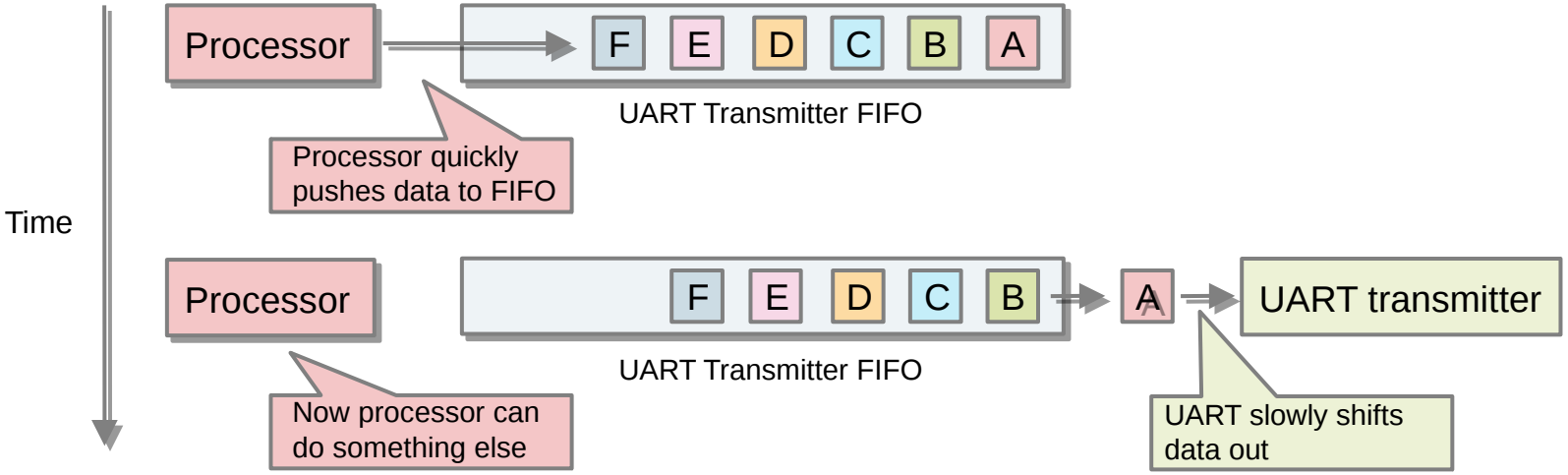
First In First Out (FIFO)

- When implemented as FIFO, the address of the two ports are automatically managed as data move to/ from the memory;
- The address can be coded in either natural binary code or Gray code;
- Binary code
  - Natural way of incrementing a number;
  - One or multiple bits are changed when incrementing the number;
  - Larger power consumption, longer switching time.
- Gray Code
  - Only one bit is changed in each increment;
  - Less power consumption, shorter switching time.

Dec	Gray	Binary
0	000	000
1	001	001
2	011	010
3	010	011
4	110	100
5	111	101
6	101	110
7	100	111

UART FIFO

- Improve system operation efficiency
  - Processor operates in a higher clock frequency, e.g. 50,000,000Hz;
  - UART is transmitted in a much lower frequency, e.g. 19,200 Hz;
  - If processor waits for the UART, a large amount of time is wasted;
  - Hence, FIFOs are used to improve the system efficiency;



Memory Space

- The memory space is allocated as follow:

Peripheral	Base address	End address	Size
MEM	0x0000_0000	0x4FFF_FFFF	167MB
VGA	0x5000_0000	0x50FF_FFFF	16MB
UART	0x5100_0000	0x51FF_FFFF	16MB

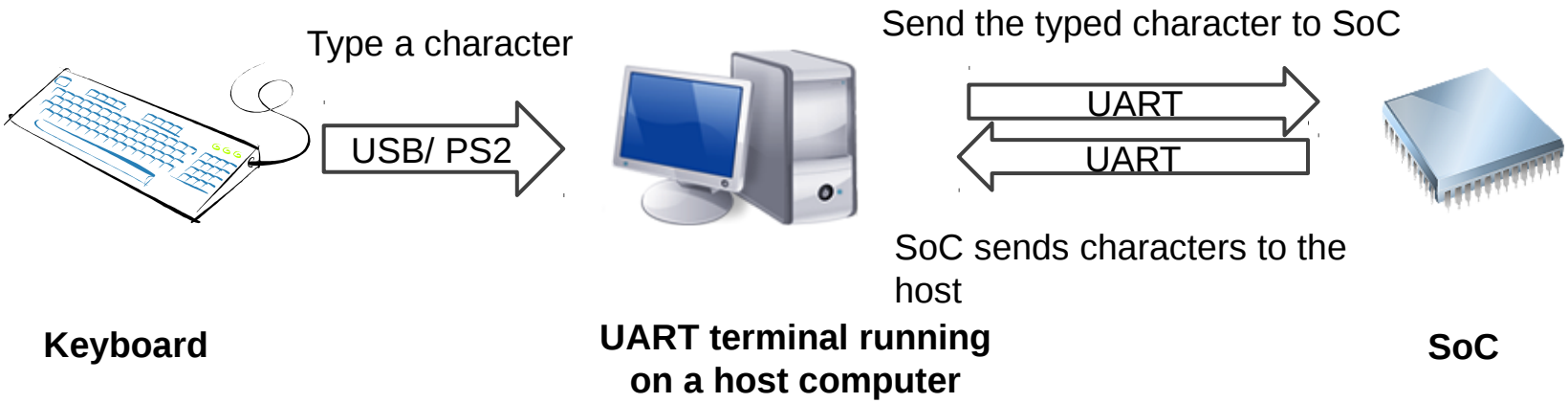
Memory Space

- The UART peripheral should have at least two registers
  - Data register
    - Used for both input and output data
  - FIFO status register
    - Bit0: Rx FIFO empty
      - If empty, processor cannot read from the FIFO.
    - Bit1: Tx FIFO full
      - If full, processor has to wait before writing to the FIFO

Register	Base address	Size
Data	0x5100_0000	4 Byte
FIFO status	0x5100_0004	4 Byte

UART Demonstration

- SoC receives characters from the host
  - Open an UART terminal, such as Putty, or Tera Term Pro;
  - Set terminal to serial communication, choose the comport number, e.g. COM4, and select the baud rate, such as 19200 bps;
  - Make sure the terminal window is activated;
  - Type the keyboard, the characters will be sent to the board via UART;
- SoC sends characters to the host
  - The characters are received by the host, and displayed on the terminal window.



Lab Practice

Lab Practice

- Step1- Hardware design
  - Design and implement the peripheral (UART peripheral) in hardware using Verilog;
- Step2- Software programming
  - Test the peripheral using Cortex-M0 processor programed in assembler language;
- Step3- System demonstration
  - Receive/ print characters from/ to a computer through UART port.

Useful Resources

- Reference1
  - Nexys3 Reference Manual:

    http://www.digilentinc.com/Data/Products/NEXYS3/Nexys3_rm.pdf
- Reference2
  - ASCII character table:

    http://en.wikipedia.org/wiki/ASCII