AVIONCHIP

ENGLISH EDITION

FPGA / EMBEDDED SYSTEMS / SOFTWARE

Understand the design. Build the capability.

A technical guide to 56 specialist courses: what you will study, how the ideas work and which engineering decisions they support.

01 / DESCRIBE

Clocked state

```
q <= q + 1;
```

02 / VERIFY

A next-cycle response

```
req |=> ack
```

03 / IMPLEMENT

An overlapped pipeline

```
latency 3 / II 1
```

Languages and verification.
Device architectures and interfaces.
Embedded software and accelerated computing.

56	10	1–4
courses	subject areas	days per course

ENGINEERING KNOWLEDGE, CONNECTED

From a topic list to a design decision.

AvionChip Technology Inc. was founded by Dr. Vecdi Emre Levent in 2022 at Boğaziçi University Technopark (BUDOTEK). The company brings together FPGA IP development, embedded systems, training and consultancy.

WHAT EACH COURSE PAGE GIVES YOU

01 Scope

The original subjects covered by the course, with its duration and entry knowledge.

02 An explained example

A code fragment, timing trace, data path, numeric result or implementation checkpoint that demonstrates a specific idea.

03 A design implication

A short interpretation connecting the example to an engineering decision.

04 Platform context

Tools and hardware context to discuss when choosing a course for your team.

Examples are simplified teaching illustrations. Register maps, device resources and tool flows must be checked against the selected platform. Course durations and core scope follow the training portfolio; delivery details are agreed with AvionChip.

FIND YOUR DIRECTION

Ten disciplines. One connected portfolio.

Start with the engineering problem you want to solve. Each section connects a technical perspective with the courses that support it.

01 HDL & verification
5 courses 08

02 Design environments
6 courses 12

03 Interfaces & connectivity
6 courses 16

04 Design methodologies
7 courses 20

05 Embedded Linux
4 courses 24

06 AMD / Xilinx platforms
15 courses 26

07 Altera / Intel platforms
4 courses 34

08 Programming & scripting
6 courses 36

09 Cloud development
1 course 39

10 Model-based design
2 courses 40

COURSE INDEX / 01 OF 03

Find your next capability.

	COURSE	DAYS	PAGE
01	Comprehensive Verilog	4	08
02	SystemVerilog for Verification	3	09
03	SystemVerilog for Design	2	10
04	Universal Verification Methodology	3	10
05	HLS Tools: Beyond the Xilinx Flow	1	11
06	Vivado & Vitis Design Environment	4	12
07	Quartus Design Environment	3	13
08	Lattice Diamond Design Environment	1	13
09	Libero Design Environment	1	14
10	Efinix Efinity Design Environment	4	14
11	Gowin EDA Design Environment	1	15
12	PCI Express	1	16
13	Ethernet	2	16
14	USB	1	17
15	High-Speed Transceivers	2	18
16	High-Speed Memory Interfaces	1	19
17	Low-Speed Interfaces	1	19
18	Clock Domain Crossing	1	20
19	Pipelining & Resource Sharing	1	20

Select a course to open its detail page.

COURSE INDEX / 02 OF 03

Find your next capability.

	COURSE	DAYS	PAGE
20	Verification & On-Chip Debug	2	21
21	Safety-Critical Design Techniques	1	22
22	Partial Reconfiguration	1	22
23	DSP IP Cores	1	23
24	Video Processing IP	1	23
25	Linux Fundamentals	1	24
26	PetaLinux	3	24
27	PetaLinux for Video Processing	1	25
28	Ubuntu Root Filesystems	1	25
29	Artix-7 & Kintex-7 Architecture	1	26
30	Zynq-7000 Architecture	3	26
31	UltraScale+ Architecture	1	27
32	Zynq UltraScale+ MPSoC Architecture	3	27
33	Zynq UltraScale+ RFSoC Architecture	1	28
34	Versal Architecture	2	28
35	Python Development for Zynq	1	29
36	Deep Learning on FPGA Platforms	1	29
37	High-Level Synthesis with Vitis HLS	2	30
38	Xilinx Runtime (XRT)	1	30

Select a course to open its detail page.

COURSE INDEX / 03 OF 03

Find your next capability.

	COURSE	DAYS	PAGE
39	Vitis Vision Library	1	31
40	Vitis AI Library	1	31
41	OpenCV in the Vitis Workflow	1	32
42	Vitis BLAS Library	1	32
43	Vitis Quantitative Finance Library	1	33
44	Cyclone FPGA Architecture	1	34
45	Intel SoC FPGA Architecture	1	34
46	Arria 10 Architecture	1	35
47	Intel FPGA OpenCL Synthesis	1	35
48	C Programming	3	36
49	C++ Programming	3	36
50	Python Programming	4	37
51	Perl Scripting	1	37
52	Linux Bash Scripting	1	38
53	Tcl for Vivado	1	38
54	FPGA Development on AWS	1	39
55	MATLAB HDL Coder	1	40
56	Vitis Model Composer	1	40

Select a course to open its detail page.

BUILD A COHERENT SEQUENCE

Start with the outcome. Connect the learning.

These are suggested routes, not fixed packages or prerequisites. Choose the depth and starting point around your team.

01 RTL design

- Comprehensive Verilog
- SV Design
- Clock Domain Crossing

02 Verification

- SV Verification
- UVM
- On-Chip Debug

03 Embedded platforms

- Zynq-7000 Architecture
- PetaLinux
- Video Processing

04 Hardware acceleration

- Vitis HLS
- Vitis Vision Library
- Xilinx Runtime (XRT)

NEW TO DIGITAL SYSTEMS?

Discuss foundational preparation before choosing a course. Entry knowledge is listed with every course.

AC-01

4 days

Comprehensive Verilog

Build a solid RTL foundation, from combinational and sequential logic to state machines, buses and memories. Understand how implementation choices affect area and performance.

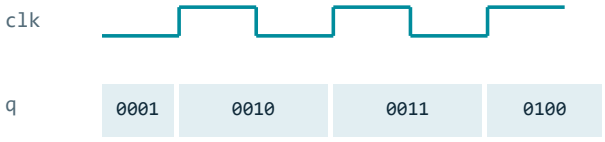
COURSE SCOPE

- Combinational logic
- Sequential logic
- State machines
- Bus elements
- Memories
- Optimization and tradeoffs

ILLUSTRATED EXAMPLE

A 4-bit registered counter

```
always @(posedge clk)
  q <= q + 1;
```



Readout. The register changes only at a rising clock edge: 0001 becomes 0010, then 0011.

Design implication. Keep the combinational increment separate from the state that stores its result.

APPLIED EXAMPLE

Model a counter and state machine, then compare simulated behavior with the intended clock-by-clock operation.

LEARNING FOCUS

Describe combinational and clocked behavior; reason about state, memory and implementation tradeoffs.

Tools. Vivado; Artix-7 FPGA

Entry. Basic knowledge of digital systems.

AC-02

3 days

SystemVerilog for Verification

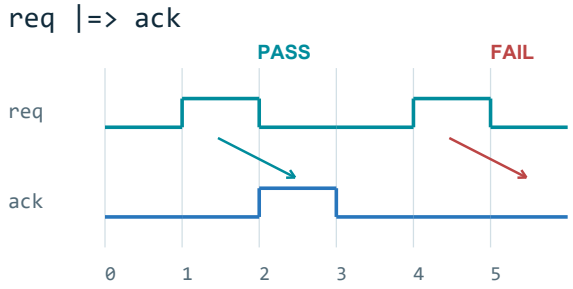
Bring structure and visibility to verification. Explore language features for stimulus, interfaces, assertions and coverage, together with object-oriented testbench construction.

COURSE SCOPE

- Data types
- Interfaces
- always_comb, always_latch and always_ff
- Assertions
- Clocking blocks
- Coverage
- Arrays and queues
- DPI
- Classes
- Inheritance
- Polymorphism
- Functional coverage

ILLUSTRATED EXAMPLE

Check the next-cycle response



Readout. The request at cycle 1 is acknowledged at cycle 2. The request at cycle 4 has no acknowledgement at cycle 5.

Design implication. An assertion turns a timing requirement into a repeatable check; coverage asks whether the scenario occurred.

APPLIED EXAMPLE

Trace a request/acknowledgement failure, then identify the stimulus and coverage point needed to reproduce it.

LEARNING FOCUS

Choose stimulus, assertions and coverage for a testbench; connect a failing check to a specific transaction.

Tools. QuestaSim / SystemVerilog simulation; Vivado

Entry. Basic knowledge of digital systems; Verilog basics.

AC-03

2 days

SystemVerilog for Design

Express hardware intent clearly with SystemVerilog data types, procedural blocks, interfaces and packages. Organize reusable RTL while distinguishing combinational and sequential behavior.

COURSE SCOPE

- Data types
- Procedural always blocks
- User-defined types
- Interfaces
- Packages

Tools. Vivado or Synplify; Artix-7 FPGA

Entry. Basic knowledge of digital systems; Verilog basics.

ILLUSTRATED EXAMPLE

Combinational vs. stored state

```
always_comb
  next_q = q + 1;

always_ff @(posedge clk)
  q <= next_q;
```

Readout. next_q is computed by combinational logic; q captures that value on a clock edge.

Design implication. Use procedural blocks that make the intended logic and storage behavior explicit.

AC-04

3 days

Universal Verification Methodology

Explore the reusable components and transaction-based communication at the heart of UVM. Connect data modeling, the factory and register models within a consistent verification structure.

COURSE SCOPE

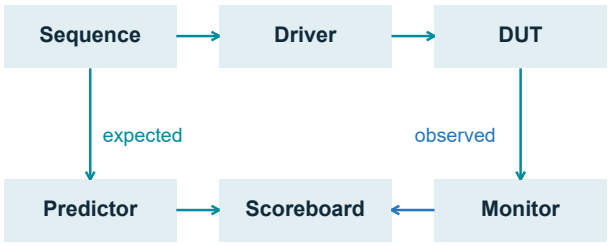
- Data modeling
- UVM classes
- UVM factory
- Transaction-level modeling (TLM)
- Register modeling

Tools. UVM-capable SystemVerilog simulator; QuestaSim

Entry. Basic knowledge of digital systems; Verilog and SystemVerilog knowledge.

ILLUSTRATED EXAMPLE

A transaction through UVM



Readout. Expected and observed transactions arrive independently at the scoreboard.

Design implication. Keep stimulus, signal driving, observation and checking separate so components can be reused.

AC-05

1 day

HLS Tools: Beyond the Xilinx Flow

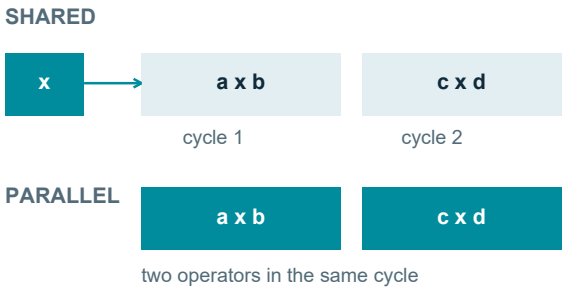
Compare approaches to high-level synthesis outside the Xilinx toolchain. Relate algorithm descriptions to generated RTL and the choices made by different HLS environments.

COURSE SCOPE

- HLS tools beyond the Xilinx toolchain

ILLUSTRATED EXAMPLE

One multiplier or two?



Readout. One operator performs two products in separate slots. Two operators can compute the products in parallel.

Design implication. Resource sharing reduces operator count but changes scheduling and possibly throughput.

APPLIED EXAMPLE

Compare a shared arithmetic operator with a parallel implementation and discuss the resulting resource and scheduling tradeoffs.

LEARNING FOCUS

Compare high-level synthesis approaches; reason about operator sharing, latency and parallelism.

Tools. Selected HLS toolchains; target agreed for the course

Entry. Basic knowledge of digital systems; Verilog knowledge.

AC-06

4 days

Vivado & Vitis Design Environment

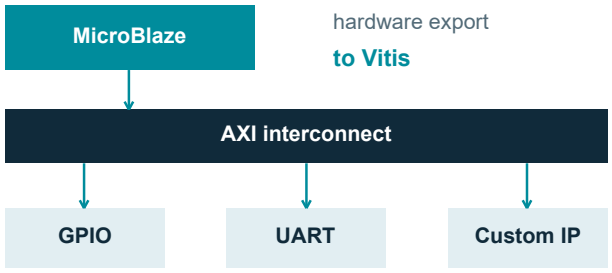
Follow a complete FPGA development flow, from projects and synthesis to IP integration and embedded software. Connect reusable IP, AXI systems and MicroBlaze-based designs.

COURSE SCOPE

- Project operations
- Synthesis analyses
- I/O planning
- Vivado IP cores
- Design with IP Integrator
- Creating an IP core
- IP packaging
- IP protection
- Using IP cores
- AXI interfaces
- Interconnect, SmartConnect
- Vivado reports
- MicroBlaze
- Vitis environment

ILLUSTRATED EXAMPLE

An AXI system in Vivado



Readout. A processor reaches GPIO, UART and custom IP through an AXI interconnect.

Design implication. Hardware connectivity and address mapping must agree with the software platform.

APPLIED EXAMPLE

Build the outline of a MicroBlaze system, map a custom peripheral and follow the hardware handoff into software development.

LEARNING FOCUS

Follow a project through synthesis and implementation; assemble and package IP around AXI and MicroBlaze.

Tools. Vivado and Vitis; Artix-7 / MicroBlaze

Entry. Basic knowledge of digital systems; Basic knowledge of RTL.

AC-07

3 days

Quartus Design Environment

Navigate the Quartus design flow through pin assignments, timing constraints and system integration. Use Platform Designer and Signal Tap to connect implementation with observation.

COURSE SCOPE

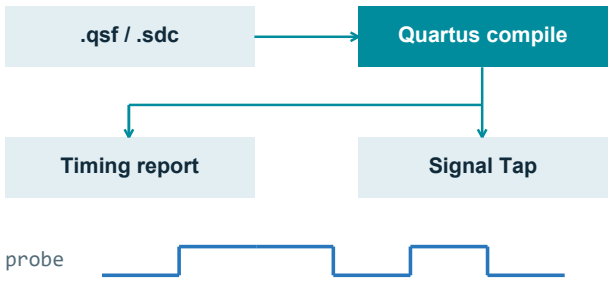
- Design flow
- Pin assignments
- Constraints
- Platform Designer (Qsys)
- Signal Tap logic analyzer

Tools. Quartus Prime; Cyclone V FPGA

Entry. Basic knowledge of digital systems.

ILLUSTRATED EXAMPLE

Constraints, timing and Signal Tap



Readout. Constraints feed implementation; the timing report and Signal Tap answer different questions.

Design implication. Static timing checks timing requirements. A capture shows what selected signals did during operation.

AC-08

1 day

Lattice Diamond Design Environment

Understand the design flow in Lattice Diamond, from a target-device project to implementation. Focus on how design entry, synthesis and programming fit together.

COURSE SCOPE

- Design flow

Tools. Lattice Diamond; Diamond-supported FPGA

Entry. Basic knowledge of digital systems; Basic RTL knowledge.

ILLUSTRATED EXAMPLE

Implementation checkpoints

RTL + LPF	Sources and physical constraints
Synthesis / MAP	Inferred and mapped logic
Place & route	Placement, routing and timing
Programming	Device and image match

Readout. RTL and physical constraints pass through synthesis, mapping and place-and-route before programming.

Design implication. A successful compile is only one checkpoint; inspect device selection, constraints and implementation reports.

AC-09

1 day

Libero Design Environment

Explore the Libero development flow for Microchip FPGA designs. Connect the project structure to synthesis, implementation and device programming.

COURSE SCOPE

- Design flow

Tools. Compatible Libero release; AGLN-NANO-KIT

Entry. Basic knowledge of digital systems; Basic knowledge of RTL.

ILLUSTRATED EXAMPLE

Implementation checkpoints

SmartDesign	IP interfaces and configuration
Synthesis	Implemented logic structure
Place & route	Physical and timing results
Programming	Selected device and image

Readout. IP integration precedes synthesis and physical implementation.

Design implication. Resolve interface and configuration mismatches before relying on the final programming image.

AC-10

4 days

Efinix Efinity Design Environment

Develop familiarity with the Efinity toolchain and the Trion platform. Trace the design flow from RTL and interface configuration to implementation.

COURSE SCOPE

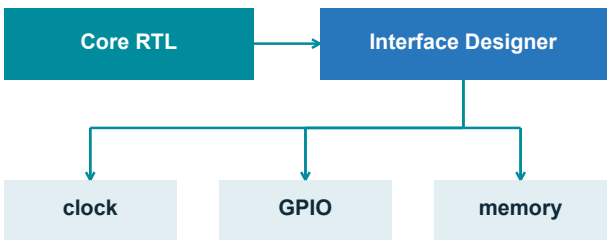
- Design flow

Tools. Efinix Efinity; Trion T20

Entry. Basic knowledge of digital systems; Basic RTL knowledge.

ILLUSTRATED EXAMPLE

Core logic meets device I/O



device interfaces

Readout. Core signals connect to configured clock, GPIO and memory interfaces.

Design implication. Peripheral configuration is part of implementation, not a substitute for core RTL.

AC-11

1 day

Gowin EDA Design Environment

Explore the Gowin FPGA development workflow. Understand how a project moves from design sources and constraints to a programmed target.

COURSE SCOPE

- Design flow

ILLUSTRATED EXAMPLE

Implementation checkpoints

RTL + CST / SDC	Logic, pin and timing inputs
Synthesis	Inferred hardware
Place & route	Physical and timing results
Programming	Board and device target

Readout. The RTL and device constraints travel through a defined implementation flow.

Design implication. Check that the constraints and programmed image refer to the same board and FPGA target.

APPLIED EXAMPLE

Trace a small design through RTL entry, device constraints, implementation and programming, checking the selected target at each step.

LEARNING FOCUS

Understand the Gowin project flow; carry physical and timing constraints into implementation.

Tools. Gowin EDA; DK-START-GW1NSR

Entry. Basic knowledge of digital systems; Basic RTL knowledge.

AC-12

1 day

PCI Express

Connect an FPGA accelerator to host software using XDMA. Examine reference designs, drivers and transfer behavior, including the performance tradeoffs between interrupts and polling.

COURSE SCOPE

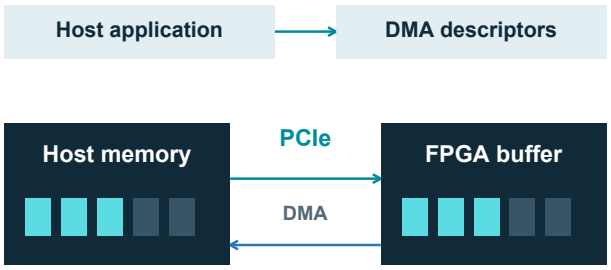
- XDMA
- Creating an XDMA reference design
- PCIe drivers and user-space software
- Windows and Linux drivers
- Performance optimization guidelines
- Interrupts versus polling

Tools. Vivado / XDMA; PCIe-capable UltraScale+ platform

Entry. Basic knowledge of digital systems.

ILLUSTRATED EXAMPLE

DMA across a PCIe link



Readout. Host software prepares the transfer; DMA moves a buffer between host memory and the FPGA.

Design implication. Transfer size, completion handling and software overhead all affect usable throughput.

AC-13

2 days

Ethernet

Explore Ethernet from PHY connections to FPGA IP and host software. Connect the protocol stack with TEMAC, MicroBlaze-based systems and higher-rate Ethernet IP.

COURSE SCOPE

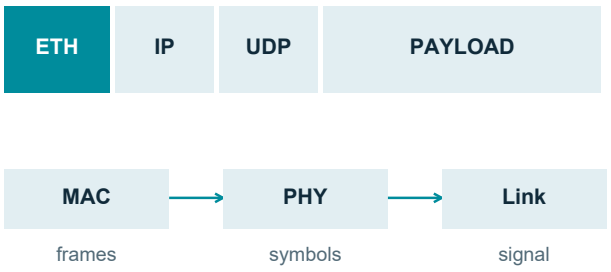
- OSI layers
- PHY interfaces
- PHY devices
- TEMAC IP
- Protocols: UDP, TCP, DHCP and ARP
- Host software development
- 10/25G Ethernet IPs
- MicroBlaze-based Ethernet

Tools. Vivado; Ethernet-capable board matched to link rate

Entry. Basic knowledge of digital systems.

ILLUSTRATED EXAMPLE

Inside an Ethernet frame



Readout. A UDP payload sits inside IP and Ethernet framing. MAC and PHY perform different parts of the path.

Design implication. Locate a problem at the correct layer before changing protocol software or physical-link settings.

AC-14

1 day

USB

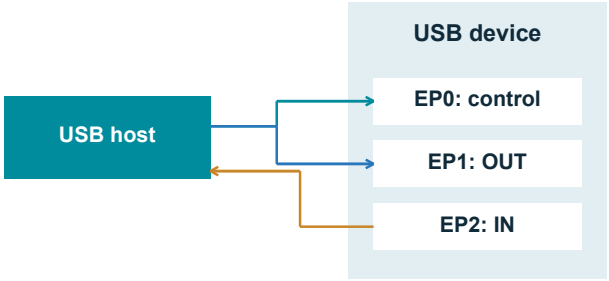
Understand USB data movement and the host software that controls it. Relate application-level transfers to the interface between a host and a device.

COURSE SCOPE

- USB data transfers
- Host software development

ILLUSTRATED EXAMPLE

USB transfers address endpoints



Readout. Endpoint 0 handles device control. Separate endpoints carry application payloads in the indicated directions.

Design implication. Design host behavior and device buffering together; a cable alone does not define the transfer model.

APPLIED EXAMPLE

Follow device control and payload transfers, identifying endpoint roles and the buffering needed by host software.

LEARNING FOCUS

Understand USB transfer behavior; connect device interfaces to host application responsibilities.

Tools. Vivado; FPGA board with suitable USB interface

Entry. Basic knowledge of digital systems.

AC-15

2 days

High-Speed Transceivers

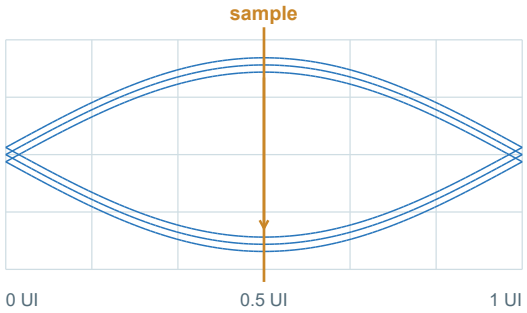
Explore the physical and digital building blocks behind serial links. Connect clocking, equalization and device resources with reference designs, IBERT and channel-bonding simulation.

COURSE SCOPE

- PMA
- PCS
- Reference clocks and PLLs
- Equalization
- Transmit de-emphasis
- Receiver equalization
- IBERT
- Line settings
- Transceiver selection
- Clocking settings
- Design with a Silicon Labs programmable oscillator
- Physical resources
- Reference design creation
- Reference design simulation
- Channel bonding
- Channel bonding simulation

ILLUSTRATED EXAMPLE

The sampling window of a serial link



Readout. The center of the synthetic eye marks the sampling region. The opening represents timing and amplitude margin.

Design implication. Relate jitter, noise and equalization to the sampling window; the illustration is not a measurement.

APPLIED EXAMPLE

Inspect a conceptual eye opening and explain how jitter, noise and equalization change the available sampling margin.

LEARNING FOCUS

Relate reference clocks and equalization to link behavior; evaluate transceiver and channel settings.

Tools. Vivado / IBERT; transceiver-equipped FPGA board

Entry. Basic knowledge of digital systems.

AC-16

1 day

High-Speed Memory Interfaces

Configure a memory interface with the Memory Interface Generator. Examine how controller settings, memory parameters and clocks shape the connection to external memory.

COURSE SCOPE

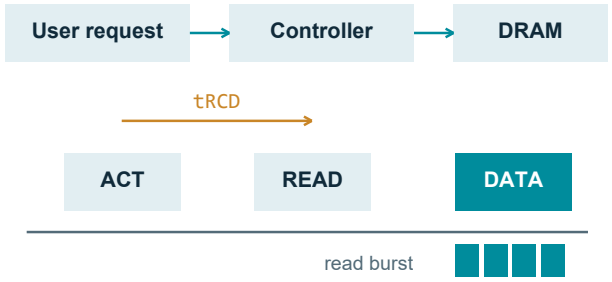
- MIG (Memory Interface Generator)
- Mode and interface selection
- Clocking
- Controller settings
- Memory settings

Tools. Vivado / MIG; FPGA board with external memory

Entry. Basic knowledge of digital systems.

ILLUSTRATED EXAMPLE

From a read request to a DDR burst



Readout. An ACT command precedes READ; data follows after the required command timing.

Design implication. Controller and memory settings must respect the device timing and selected interface mode.

AC-17

1 day

Low-Speed Interfaces

Connect control and peripheral devices using UART, SPI and I2C. Compare logic-based implementations with processor-controlled interfaces.

COURSE SCOPE

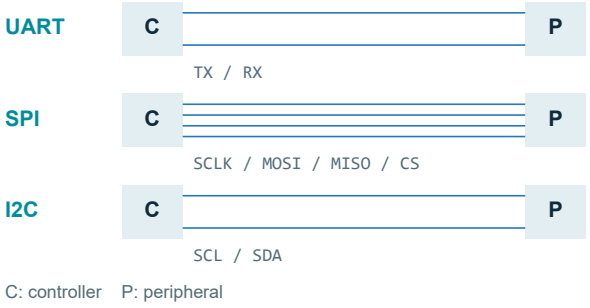
- UART
- SPI
- I2C
- Logic-based and processor-based implementations

Tools. Vivado; Artix-7 and interface peripherals

Entry. Basic knowledge of digital systems.

ILLUSTRATED EXAMPLE

UART, SPI and I2C wiring



Readout. UART uses transmit/receive signals, SPI adds clock and selection, and I2C shares clock and data.

Design implication. Choose the protocol around topology, peripheral requirements and control complexity.

AC-18

1 day

Clock Domain Crossing

Understand why asynchronous clock boundaries need deliberate treatment. Explore metastability, synchronization and the structure of designs that use multiple clocks.

COURSE SCOPE

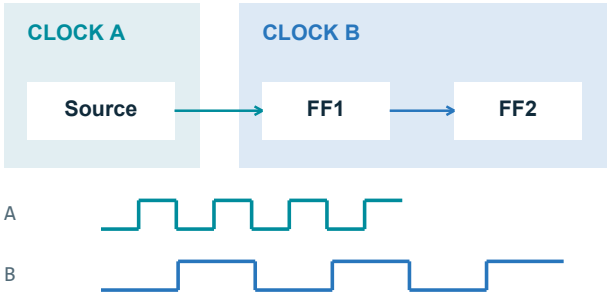
- Metastability
- Synchronization
- Multi-clock design

Tools. Vivado; Artix-7 FPGA

Entry. Basic knowledge of digital systems.

ILLUSTRATED EXAMPLE

One bit crosses a clock boundary



Readout. The stable control bit enters the receiving clock domain through two consecutive registers.

Design implication. This pattern is for a stable single bit. Pulses and multi-bit values require suitable transfer schemes.

AC-19

1 day

Pipelining & Resource Sharing

Explore the balance between clock frequency, latency and hardware area. Compare pipeline stages with shared resources when organizing a digital data path.

COURSE SCOPE

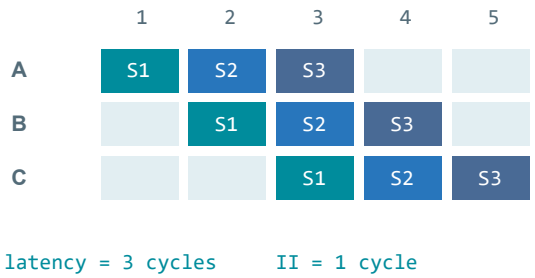
- Techniques for improving clock frequency
- Pipelining techniques
- Reducing area through resource sharing

Tools. Vivado; Artix-7 FPGA

Entry. Basic knowledge of digital systems.

ILLUSTRATED EXAMPLE

Three-stage pipeline schedule



Readout. Inputs A, B and C overlap in a three-stage pipeline. Latency is three cycles; a new input starts every cycle.

Design implication. Throughput and latency are different quantities. Adding stages does not automatically reduce time to first result.

AC-20

2 days

Verification & On-Chip Debug

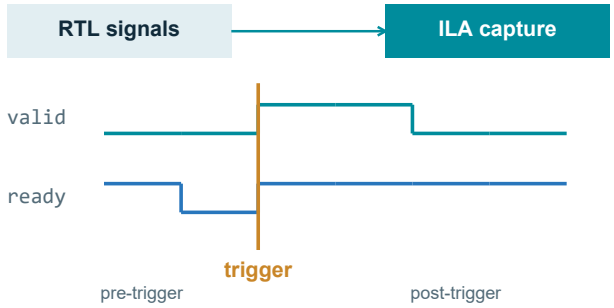
Build a verification workflow that reaches from testbenches to hardware. Combine automated checks and simulation constructs with ILA, VIO and JTAG-to-AXI observation.

COURSE SCOPE

- Verification methodology
- Testbench development
- Black-box and white-box verification
- Fork-join concurrency
- Automated testbench development
- Observation and monitoring functions
- Loops
- Tasks and functions
- File operations
- On-chip verification
- JTAG-to-AXI Master
- ILA (Integrated Logic Analyser)
- System ILA
- VIO
- Video verification IP

ILLUSTRATED EXAMPLE

Trigger and capture with an ILA



Readout. The trigger locates a condition within captured signal history; pre-trigger samples preserve the lead-up.

Design implication. Choose signals and trigger conditions that can test a specific hypothesis about the hardware failure.

APPLIED EXAMPLE

Choose internal signals and a trigger for an ILA capture, then relate the captured transaction to the expected testbench behavior.

LEARNING FOCUS

Combine simulation checks with on-chip observation; use trigger conditions to locate relevant behavior.

Tools. Vivado simulation and hardware debug; Artix-7

Entry. Basic knowledge of digital systems.

AC-21

1 day

Safety-Critical Design Techniques

Examine RTL practices that improve fault detection and resilience. Explore redundancy, error-control mechanisms and safe state-machine behavior alongside coverage.

COURSE SCOPE

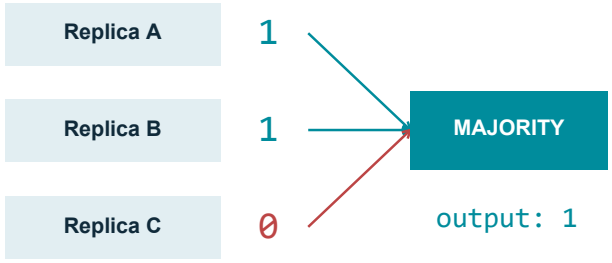
- Verilog best practices
- Code coverage
- TMR (Triple Modular Redundancy)
- ECC (error-correcting codes)
- CRC (cyclic redundancy check)
- Safe state machines

Tools. Vivado; Artix-7 FPGA

Entry. Basic knowledge of digital systems.

ILLUSTRATED EXAMPLE

Three results, one majority vote



Readout. Two replicas report 1 and one reports 0; the majority result is 1.

Design implication. Redundancy depends on a fault model. A voter does not make every correlated or common-mode fault harmless.

AC-22

1 day

Partial Reconfiguration

Understand how part of an FPGA design can be reconfigured while a static region remains in place. Explore partition requirements, Pblocks and PCIe-based update paths.

COURSE SCOPE

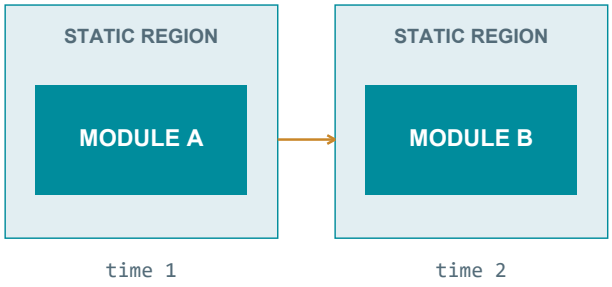
- Partial reconfiguration requirements
- Pblocks
- Partial reconfiguration over PCIe

Tools. Vivado; partial-reconfiguration-capable target

Entry. Basic knowledge of digital systems.

ILLUSTRATED EXAMPLE

A function changes; the shell stays



Readout. The active function changes from module A to module B while the static interface boundary is retained.

Design implication. Plan the partition and its interfaces before implementing independently reconfigurable functions.

AC-23

1 day

DSP IP Cores

Explore reusable arithmetic and signal-processing IP. Connect transforms, filters, oscillators and numeric formats when building FPGA processing chains.

COURSE SCOPE

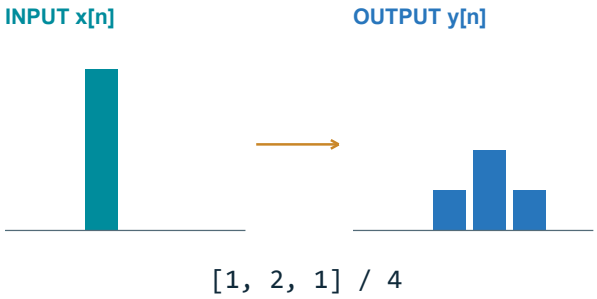
- CORDIC
- FFT
- DFT
- Complex multiplier
- Floating-point arithmetic
- DDS Compiler
- Adder/Subtractor
- FIR

Tools. Vivado DSP IP; Artix-7 FPGA

Entry. Basic knowledge of digital systems.

ILLUSTRATED EXAMPLE

A three-tap FIR smooths samples



Readout. The input impulse of 4 becomes output samples 1, 2, 1 under the three-tap filter [1, 2, 1]/4.

Design implication. Coefficients, sample history and numeric precision determine the output; the plot uses the same vertical scale.

AC-24

1 day

Video Processing IP

Explore FPGA video data movement and processing with VDMA, the Video Processing Subsystem and scaling IP. Connect frame storage with streaming processing.

COURSE SCOPE

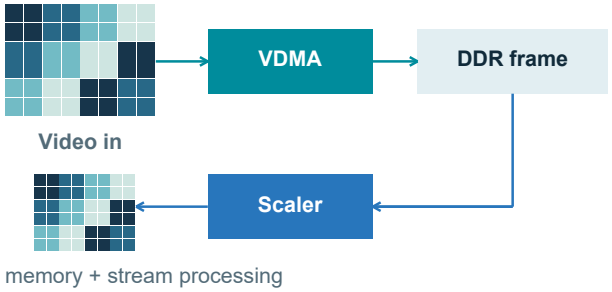
- VDMA
- VPSS
- Video scaling

Tools. Vivado video IP; compatible video platform

Entry. Basic knowledge of digital systems.

ILLUSTRATED EXAMPLE

Frame memory and video scaling



Readout. VDMA stores a frame in DDR; a processing stage reads the frame and produces a scaled output.

Design implication. Frame size, pixel format and buffering link memory bandwidth to stream behavior.

AC-25

1 day

Linux Fundamentals

Become comfortable with core Linux commands and the shell. Establish the command-line foundation needed for embedded development and script-based workflows.

COURSE SCOPE

- Basic Linux commands
- Introduction to Bash scripting

Tools. Linux workstation or virtual machine; Bash

Entry. Basic knowledge of digital systems.

ILLUSTRATED EXAMPLE

Navigate a Linux project

```
$ pwd
/home/user/project
$ ls
rtl  sim  build
$ cd rtl
```

Readout. pwd reports the working directory; ls lists its entries; cd selects a subdirectory.

Design implication. Make location and file selection explicit before automating a command sequence.

AC-26

3 days

PetaLinux

Explore the embedded Linux stack from U-Boot and the kernel to the root filesystem and applications. Connect software configuration with programmable-logic and processor cooperation.

COURSE SCOPE

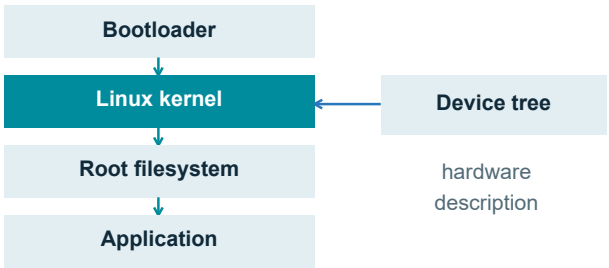
- PetaLinux toolchain
- Kernel
- U-Boot
- Filesystem
- Application development
- PL/PS co-processing

Tools. PetaLinux / Vivado; Zynq-7000 platform

Entry. Basic knowledge of digital systems.

ILLUSTRATED EXAMPLE

The embedded Linux boot chain



Readout. The bootloader starts the kernel; the kernel uses a hardware description and mounts user-space files.

Design implication. Distinguish boot, kernel and root-filesystem failures before changing the platform configuration.

AC-27

1 day

PetaLinux for Video Processing

Explore video transfer between programmable logic and the processing system. Understand the role of software and memory when moving frames across a heterogeneous platform.

COURSE SCOPE

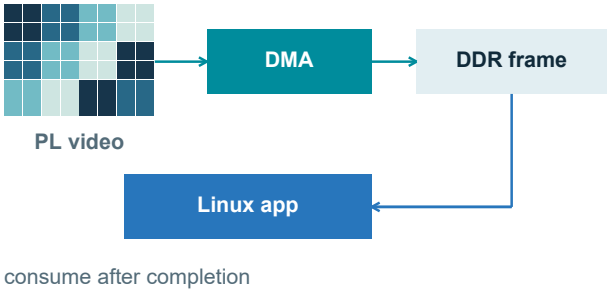
- PL/PS video-transfer techniques

Tools. PetaLinux / Vivado; Zynq video platform

Entry. Basic knowledge of digital systems.

ILLUSTRATED EXAMPLE

A video frame reaches Linux



Readout. A captured PL frame reaches a DDR buffer before a Linux application consumes it.

Design implication. Coordinate buffer ownership and transfer completion between hardware and software.

AC-28

1 day

Ubuntu Root Filesystems

Explore the use of an Ubuntu root filesystem with a PetaLinux-based platform. Distinguish the user-space filesystem from the bootloader, kernel and board-specific hardware setup.

COURSE SCOPE

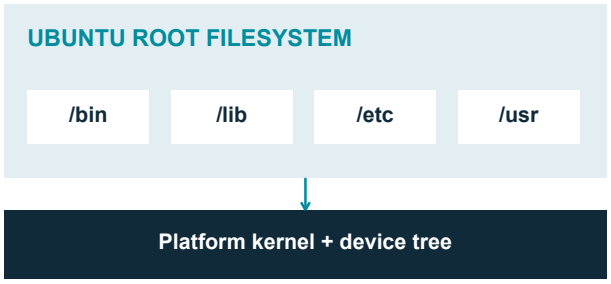
- PetaLinux with an Ubuntu root filesystem

Tools. PetaLinux; compatible Ubuntu rootfs and SoC board

Entry. Basic knowledge of digital systems.

ILLUSTRATED EXAMPLE

Root filesystem vs. platform kernel



Readout. The root filesystem contains user-space directories; the platform still needs its own kernel and device tree.

Design implication. Changing applications and libraries does not replace board-specific hardware support.

AC-29

1 day

Artix-7 & Kintex-7 Architecture

Explore the building blocks of 7-series FPGAs. Connect configurable logic, clocking, I/O drive characteristics and DSP resources to practical implementation choices.

COURSE SCOPE

- CLB
- MMCM
- PLL
- I/O clocking
- I/O drive characteristics
- DSPs

Tools. Vivado; Artix-7 / Kintex-7

Entry. Basic knowledge of digital systems.

ILLUSTRATED EXAMPLE

Match the function to the resource

CLB	Control logic and state
DSP48	Dedicated arithmetic
Block RAM	Local buffers and memories
MMCM / PLL	Clock generation and control
I/O	External signal interfaces

Readout. Logic, arithmetic, memory, I/O and clocking are separate resource classes.

Design implication. Map the design's needs to the selected device instead of treating all FPGA resources as interchangeable.

AC-30

3 days

Zynq-7000 Architecture

Understand the processing system and programmable logic in a Zynq-7000 device. Explore bare-metal applications, interfaces and AXI-based communication between PS and PL.

COURSE SCOPE

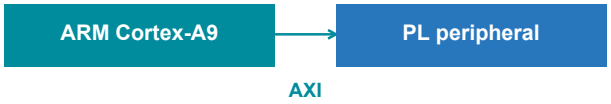
- Processing-system blocks
- Interfaces
- Bare-metal applications
- PL/PS communication
- AXI interfaces

Tools. Vivado / Vitis; Zynq-7000

Entry. Basic knowledge of digital systems.

ILLUSTRATED EXAMPLE

Software controls a PL register




```
base   = 0x40000000
offset = 0x10
write(base + offset, 1)
```

Readout. Software writes to an illustrative peripheral address: base 0x40000000 plus offset 0x10.

Design implication. Use the actual hardware address map; register offsets and access semantics are design-specific.

AC-31

1 day

UltraScale+ Architecture

Explore UltraScale+ logic, memory and I/O resources. Connect RTL and XPM-based UltraRAM use with device primitives, DSP resources and serial I/O structures.

COURSE SCOPE

- I/O drive characteristics
- UltraRAM inference in RTL
- UltraRAM with XPM macros
- Device primitives
- CLB
- DSP
- I/O SERDES

Tools. Vivado; UltraScale+ FPGA

Entry. Basic knowledge of digital systems.

ILLUSTRATED EXAMPLE

Match the function to the resource

CLB	Configurable logic
DSP48E2	Arithmetic datapaths
UltraRAM	Large on-chip buffers
Block RAM	Local memories
I/O SERDES	I/O serialization functions

Readout. The device combines configurable logic, DSP, embedded memory and I/O resources.

Design implication. Match RTL inference or primitive use to available resources and the selected device variant.

AC-32

3 days

Zynq UltraScale+ MPSoC Architecture

Explore application processors, real-time processing and multimedia resources in the MPSoC family. Understand how processing roles fit within a heterogeneous device.

COURSE SCOPE

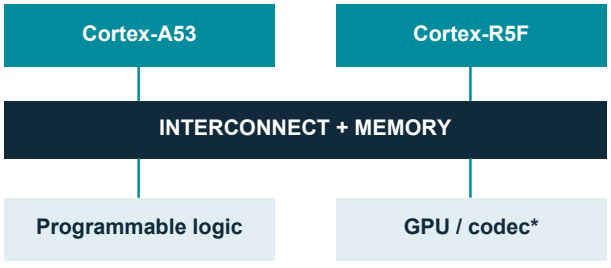
- Architecture
- APU
- Real-time processor
- H.264/H.265 codec (EV variants)
- Mali GPU (EG/EV variants)

Tools. Vivado / Vitis; Zynq UltraScale+ MPSoC

Entry. Basic knowledge of digital systems.

ILLUSTRATED EXAMPLE

Partition an MPSoC workload



* device-dependent

Readout. Application processors, real-time processors and PL share a system interconnect and memory context.

Design implication. Partition work by timing, compute and communication needs; GPU and codec availability varies by device.

AC-33

1 day

Zynq UltraScale+ RFSoC Architecture

Explore the relationship between processing resources and integrated data converters. Connect ADC and DAC paths with the programmable logic that handles digital signal processing.

COURSE SCOPE

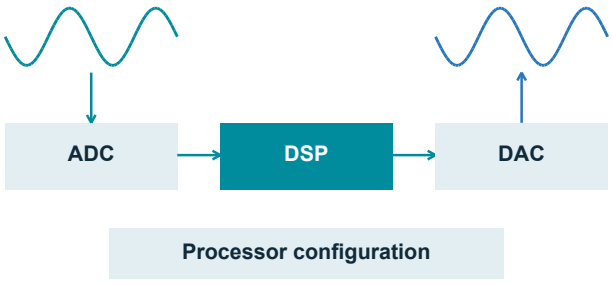
- Real-time processor
- DAC/ADC

Tools. Vivado; Zynq UltraScale+ RFSoC

Entry. Basic knowledge of digital systems.

ILLUSTRATED EXAMPLE

RF conversion and digital processing



Readout. ADC converts an analog input into samples; DSP processes them; DAC returns samples to an analog output.

Design implication. Identify sample-domain boundaries and configuration responsibilities before partitioning the signal chain.

AC-34

2 days

Versal Architecture

Explore embedded AI processing and machine-learning applications in the Versal family. Understand the roles of specialized compute resources within an adaptive platform.

COURSE SCOPE

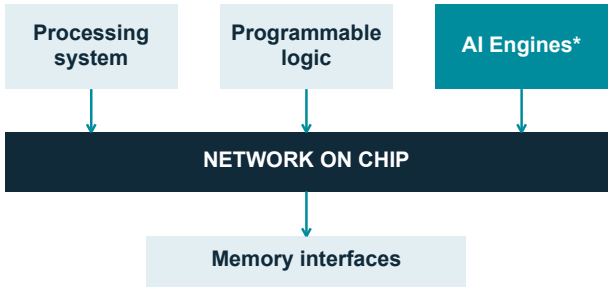
- Embedded AI blocks
- Machine-learning applications

Tools. Vivado / Vitis; suitable Versal AI-capable device

Entry. Basic knowledge of digital systems.

ILLUSTRATED EXAMPLE

Versal compute and data paths



Readout. Processing, programmable logic and AI resources reach memory through system interconnect.

Design implication. A compute engine needs a viable data path; AI Engine availability depends on the chosen device.

AC-35

1 day

Python Development for Zynq

Explore a Python-led development approach for Zynq platforms. Connect a software-friendly workflow with hardware functions implemented in programmable logic.

COURSE SCOPE

- Python-based development methodology

Tools. Python environment; compatible Zynq platform

Entry. Basic knowledge of digital systems.

ILLUSTRATED EXAMPLE

Python controls programmable logic

```
from pynq import Overlay
ol = Overlay("design.bit")
ip = ol.my_accel_0
ip.write(0x10, 1)
```



Readout. The example loads an overlay, selects a named IP block and writes a register.

Design implication. The overlay's IP names and register map must match the Python code that controls the hardware.

AC-36

1 day

Deep Learning on FPGA Platforms

Explore the development methodology for deploying deep-learning inference with a Xilinx DPU. Relate the trained model and software stack to a compatible accelerator platform.

COURSE SCOPE

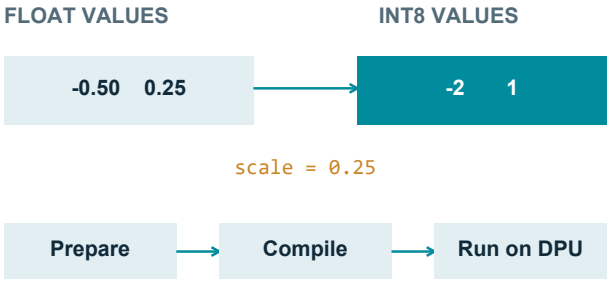
- AI development methodology with Xilinx DPU

Tools. Compatible Vitis AI / DPU release and SoC platform

Entry. Basic knowledge of digital systems.

ILLUSTRATED EXAMPLE

Prepare a model for inference



Readout. With scale 0.25 and zero-point 0, -0.50 becomes -2 and 0.25 becomes 1.

Design implication. Quantization changes numeric representation; validate the prepared model and the target-specific deployment flow.

AC-37

2 days

High-Level Synthesis with Vitis HLS

Translate algorithmic descriptions into hardware using Vitis HLS. Explore I/O interfaces and the relationship between latency, resource use and implementation choices.

COURSE SCOPE

- Vitis HLS
- I/O interface types
- Latency reduction
- Resource-utilization optimization

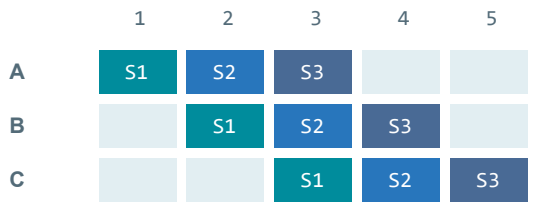
Tools. Vitis HLS / Vivado; supported FPGA target

Entry. Basic knowledge of digital systems.

ILLUSTRATED EXAMPLE

A pipelined loop in Vitis HLS

```
#pragma HLS PIPELINE II=1
```



Readout. II=1 requests a new loop iteration each cycle; the staggered stages show the intended overlap.

Design implication. Dependencies, operator resources and interface bandwidth may prevent the requested initiation interval.

AC-38

1 day

Xilinx Runtime (XRT)

Explore the runtime layer that connects host software to an accelerator. Understand its position between the application, data buffers and hardware execution.

COURSE SCOPE

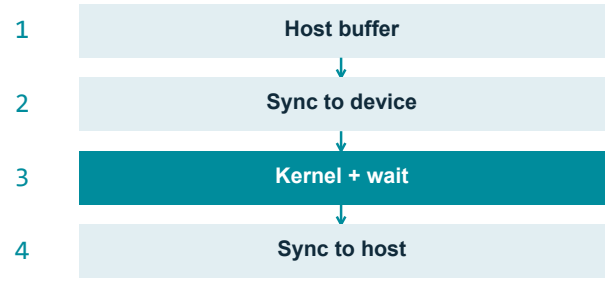
- Xilinx Runtime environment

Tools. XRT; supported accelerator platform and toolchain

Entry. Basic knowledge of digital systems.

ILLUSTRATED EXAMPLE

The XRT buffer lifecycle



Readout. The host transfers input, starts a kernel, waits for completion and retrieves output.

Design implication. Explicit ordering prevents either side from consuming data before it is ready.

AC-39

1 day

Vitis Vision Library

Explore image-processing functions expressed as HLS library components. Connect algorithm-level image operations with hardware-oriented data movement.

COURSE SCOPE

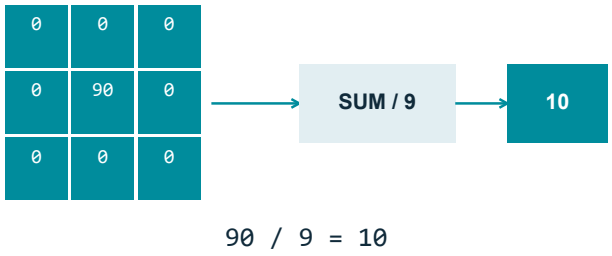
- Image-processing HLS library

Tools. Vitis Vision; compatible HLS flow and target

Entry. Basic knowledge of digital systems.

ILLUSTRATED EXAMPLE

A 3 x 3 image neighborhood



Readout. The nine pixels sum to 90. A 3 x 3 box filter divides by 9, giving a center output of 10.

Design implication. A neighborhood operation depends on nearby pixels, boundary handling and the chosen numeric representation.

AC-40

1 day

Vitis AI Library

Explore reusable AI application components in the Vitis AI ecosystem. Understand how library APIs sit above the model-execution layer on a supported platform.

COURSE SCOPE

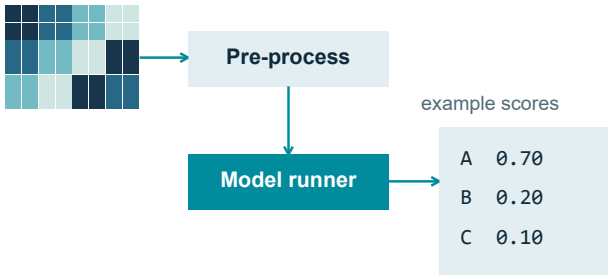
- AI library components

Tools. Vitis AI Library; supported model and platform

Entry. Basic knowledge of digital systems.

ILLUSTRATED EXAMPLE

The inference application pipeline



Readout. Preprocessing prepares an input; the runner returns model outputs; the application interprets the scores.

Design implication. Keep input transformation and output interpretation consistent with the deployed model.

AC-41

1 day

OpenCV in the Vitis Workflow

Explore the relationship between OpenCV image-processing concepts and a Vitis-based FPGA workflow. Connect software-oriented image operations with accelerated processing.

COURSE SCOPE

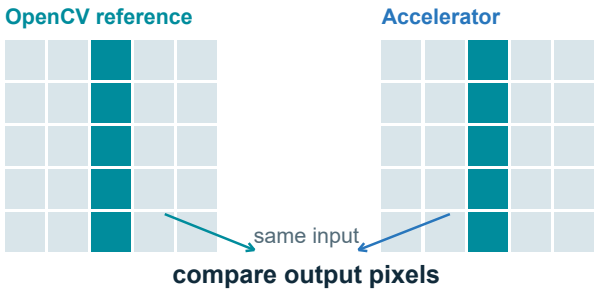
- OpenCV-based video-processing workflow

Tools. OpenCV; compatible Vitis vision / HLS workflow

Entry. Basic knowledge of digital systems.

ILLUSTRATED EXAMPLE

Reference and accelerated images



Readout. Both paths receive the same input representation; their output pixels are compared.

Design implication. Define the comparison tolerance and boundary behavior before judging an accelerated result.

AC-42

1 day

Vitis BLAS Library

Explore hardware-oriented building blocks for linear algebra. Connect vectors, matrices and arithmetic kernels to an FPGA acceleration workflow.

COURSE SCOPE

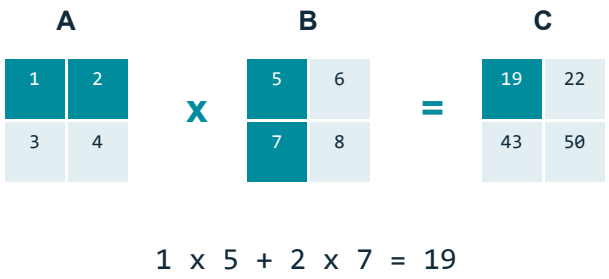
- Linear algebra library operations

Tools. Vitis BLAS; compatible HLS acceleration flow

Entry. Basic knowledge of digital systems.

ILLUSTRATED EXAMPLE

One matrix multiplication



Readout. The highlighted output is 19 because row [1, 2] meets column [5, 7]: 1 x 5 + 2 x 7.

Design implication. Matrix acceleration depends on operand reuse and data supply as well as parallel arithmetic.

AC-43

1 day

Vitis Quantitative Finance Library

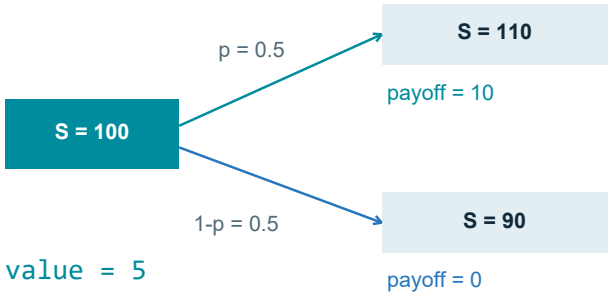
Explore HLS library components for financial computation. Connect a mathematical model to a hardware-oriented calculation flow.

COURSE SCOPE

- HLS library for quantitative finance

ILLUSTRATED EXAMPLE

A one-step option-pricing tree



Readout. At strike 100, terminal prices 110 and 90 give call payoffs 10 and 0. With r=0 and p=0.5, value is 5.

Design implication. Establish a numerical reference before comparing execution speed or hardware resource use.

APPLIED EXAMPLE

Evaluate a simple one-step option-pricing example and compare its numerical reference with an accelerated computation strategy.

LEARNING FOCUS

Connect quantitative-finance algorithms with HLS libraries; separate numerical correctness from hardware performance.

Tools. Vitis Quantitative Finance; compatible HLS flow

Entry. Basic knowledge of digital systems.

AC-44

1 day

Cyclone FPGA Architecture

Explore the architecture of the Cyclone FPGA family. Understand the relationship between programmable logic, embedded resources and the device-oriented design flow.

COURSE SCOPE

- Cyclone FPGA architecture

Tools. Quartus Prime; Cyclone family target

Entry. Basic knowledge of digital systems.

ILLUSTRATED EXAMPLE

Match the function to the resource

ALM	Logic and registers
M10K	Embedded memory
DSP	Dedicated arithmetic
PLL	Clocking
I/O / transceivers	Device-dependent interfaces

Readout. The schematic distinguishes logic, memory, arithmetic, clocking and interface resources.

Design implication. Check the actual Cyclone part; resource availability and peripheral features vary by device.

AC-45

1 day

Intel SoC FPGA Architecture

Explore Arm Cortex-A9-based SoC FPGA systems. Connect the hard processor system with the FPGA fabric and understand their distinct roles.

COURSE SCOPE

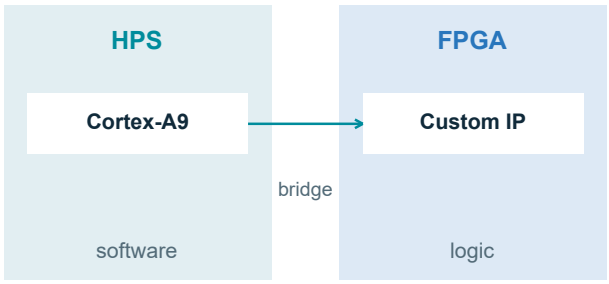
- Arm Cortex-A9 architecture
- SoC processor / FPGA architecture

Tools. Quartus Prime; Cortex-A9-based SoC FPGA

Entry. Basic knowledge of digital systems.

ILLUSTRATED EXAMPLE

The HPS-to-FPGA boundary



Readout. Processor software accesses custom logic across an HPS-to-FPGA bridge.

Design implication. Address-space configuration and peripheral behavior must agree across the hardware/software boundary.

AC-46

1 day

Arria 10 Architecture

Explore the Arria 10 FPGA architecture and its floating-point-capable DSP resources. Connect arithmetic-intensive workloads to the device's implementation resources.

COURSE SCOPE

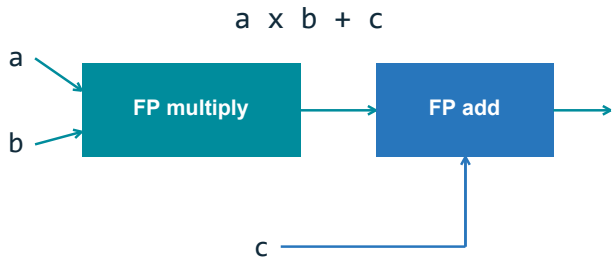
- Arria 10 architecture
- Embedded floating-point resources

Tools. Quartus Prime; Arria 10 FPGA

Entry. Basic knowledge of digital systems.

ILLUSTRATED EXAMPLE

Floating-point DSP arithmetic



Readout. The datapath computes a x b + c using multiplication followed by addition.

Design implication. Arithmetic resources need a sustained operand stream; precision and mapping choices affect implementation.

AC-47

1 day

Intel FPGA OpenCL Synthesis

Explore an OpenCL-based route to FPGA acceleration on SoC platforms. Relate host code, kernels and a compatible board support package to the compilation flow.

COURSE SCOPE

- OpenCL synthesis for SoC platforms

Tools. Compatible Intel FPGA OpenCL SDK and board package

Entry. Basic knowledge of digital systems.

ILLUSTRATED EXAMPLE

Host code and an OpenCL kernel

```
__kernel void add_one(
    __global int *a) {
  int i = get_global_id(0);
  a[i] += 1;
}
```



Readout. Each OpenCL work-item updates one element using its global index.

Design implication. The host still owns buffer setup, arguments, execution ordering and result handling.

AC-48

3 days

C Programming

Build a foundation in the C language. Develop a clear understanding of how program structure and basic language constructs describe software behavior.

COURSE SCOPE

- C language fundamentals

Tools. C compiler and development environment

Entry. Basic knowledge of digital systems.

ILLUSTRATED EXAMPLE

A C function receives and returns

```c
int add_one(int x) {
    return x + 1;
}
```



Readout. The function receives 4, adds 1 and returns 5.

Design implication. State the function's input, output and data representation before expanding the program.

AC-49

3 days

C++ Programming

Explore C++ through object-oriented programming. Connect data and behavior in classes and use these concepts to organize software.

COURSE SCOPE

- Object-oriented programming in C++

Tools. C++ compiler and development environment

Entry. Basic knowledge of digital systems.

ILLUSTRATED EXAMPLE

Objects keep independent state

```cpp
class Counter {
  int value = 0;
public:
  void tick() { ++value; }
};
```



Readout. Calling a.tick() changes a from 0 to 1; b retains its independent state.

Design implication. A class defines shared behavior, while each object owns its instance data.

AC-50

4 days

Python Programming

Explore Python from fundamental constructs to advanced and object-oriented programming. Build a coherent view of the language as a tool for reusable software.

COURSE SCOPE

- Fundamental and advanced Python
- Object-oriented programming

Tools. Python interpreter and development environment

Entry. Basic knowledge of digital systems.

ILLUSTRATED EXAMPLE

A Python function maps a list

```
def add_one(x):
    return x + 1
list(map(add_one, [1,2,3]))
```



Readout. Mapping add_one over [1, 2, 3] produces [2, 3, 4].

Design implication. Separate a reusable transformation from the collection that it processes.

AC-51

1 day

Perl Scripting

Explore basic Perl script development. Understand the role of a scripting language in automating text-oriented and repeatable software tasks.

COURSE SCOPE

- Basic Perl script development

Tools. Perl interpreter; scripting environment

Entry. Basic knowledge of digital systems.

ILLUSTRATED EXAMPLE

A Perl expression captures text

```
my $line = "count=42";
if ($line =~ /count=(\d+)/) {
    print "$1\n";
}
```



Readout. The capture group extracts 42 from the text count=42.

Design implication. Validate the input shape and decide how the script should handle a missing or malformed field.

AC-52

1 day

Linux Bash Scripting

Explore the structure and use of Bash scripts in a Linux environment. Connect commands into workflows that can be repeated consistently.

COURSE SCOPE

- Bash scripting

Tools. Linux workstation or virtual machine; Bash

Entry. Basic knowledge of digital systems.

ILLUSTRATED EXAMPLE

A shell loop processes log files

```
for file in *.log; do
    wc -l "$file"
done
```

```
Example: 128 run.log
```

Readout. The loop runs wc -l on each matching log file and quotes the filename argument.

Design implication. Quoting preserves filenames as single arguments; error handling is a separate scripting decision.

AC-53

1 day

Tcl for Vivado

Explore scripting in the Vivado Tcl environment. Connect project operations with script-driven execution to make development tasks easier to repeat.

COURSE SCOPE

- Scripting in the Vivado Tcl environment

Tools. Vivado Tcl console and script environment

Entry. Basic knowledge of digital systems.

ILLUSTRATED EXAMPLE

A reproducible Vivado Tcl run

```
open_project design.xpr
launch_runs synth_1
wait_on_run synth_1
open_run synth_1
report_timing_summary
```

```
project > run > wait > report
```

Readout. The script opens a project, launches synthesis, waits, opens the run and produces a timing report.

Design implication. Sequence run-dependent operations explicitly so a report refers to the intended completed design.

AC-54

1 day

FPGA Development on AWS

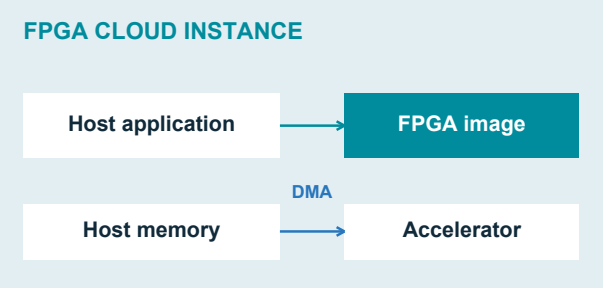
Explore FPGA-oriented development using Amazon's cloud infrastructure. Connect the host application and accelerator development flow to a supported cloud environment.

COURSE SCOPE

- Development using Amazon cloud infrastructure

ILLUSTRATED EXAMPLE

Host and FPGA on a cloud instance



Readout. The host application and FPGA image form one deployment; memory transfers connect software to the accelerator.

Design implication. Match instance, image, runtime and host application before diagnosing application-level behavior.

APPLIED EXAMPLE

Trace a workload from the remote host application into an FPGA accelerator and identify the artifacts needed for a compatible deployment.

LEARNING FOCUS

Understand the cloud FPGA development context; align image, runtime and host software requirements.

Tools. AWS FPGA-capable instance; compatible development kit

Entry. Basic knowledge of digital systems.

AC-55

1 day

MATLAB HDL Coder

Explore RTL generation from MATLAB and Simulink designs with HDL Coder. Connect a high-level algorithm or model to synthesizable hardware descriptions.

COURSE SCOPE

- RTL generation with MATLAB HDL Coder

Tools. MATLAB / Simulink; HDL Coder; FPGA toolchain

Entry. Basic knowledge of digital systems.

ILLUSTRATED EXAMPLE

A model becomes equivalent RTL




```
// Equivalent RTL sketch
assign y = u * 3;
```

```
u = 4    ->    y = 12
```

Readout. A gain of 3 maps to the RTL expression y = u * 3; input 4 produces output 12.

Design implication. Check numeric widths and signedness when comparing the model and generated hardware behavior.

AC-56

1 day

Vitis Model Composer

Explore model-based hardware design in the MATLAB and Simulink environment. Connect graphical models and AMD blocks with generated implementation artifacts.

COURSE SCOPE

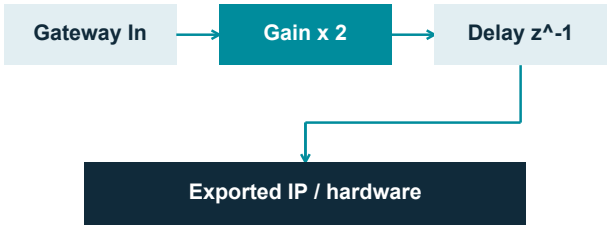
- RTL generation with Vitis Model Composer in Simulink

Tools. MATLAB / Simulink; Vitis Model Composer; AMD tools

Entry. Basic knowledge of digital systems.

ILLUSTRATED EXAMPLE

A sampled model becomes an IP block




```
input: 0,1,2,3   output: 0,0,2,4
```

Readout. A gain of 2 followed by a unit delay maps input 0,1,2,3 to output 0,0,2,4.

Design implication. Treat sample timing and numeric formats as part of the model-to-hardware contract.

AVIONCHIP

YOUR NEXT ENGINEERING STEP

Choose the capability. Connect the learning.

Start with what your team needs to build, verify or understand. Use the course scope, illustrated examples and entry knowledge to shape a focused training route.

01 Define the challenge

Name the design problem, platform transition or capability gap.

02 Select the courses

Use the index and suggested routes to connect the relevant topics.

03 Align the delivery

Discuss tools, hardware, existing experience and scheduling.

info@avionchip.com

avionchip.com