1 Design Flow

Place each token in the numbered slots to form the generic digital design flow sequence.
Note: *Each slot will be checked automatically and individually.* A sequence offset by one position will not be rewarded.

Help

RTL simulation	Device programming	Synthesis
Specification	VHDL design entry	Place and route
Gate level simulation	Static timing analysis	

1: Specification ✔
2: VHDL design entry ✔
3: RTL simulation ✔
4: Synthesis ✔
5: Gate level simulatic ✔
6: Place and route ✔
7: Static timing analys ✔
8: Device programmir ✔

Maximum marks: 2

2 Digital design knowledge

Select the best alternative for each of the thirty numbered tasks.

1: What does synthesizable HDL-code define:
Select one alternative:

- A program that will run on an FPGA
- A circuit description ✔
- A testbench
- Combinational logic
- Sequential logic

2: Which of these are **not** considered programmable logic:
Select one alternative

- CPLD, Complex Programmable Logic Device
- FPGA, Field Programmable Gate Array
- PAL, Programmable array Logic
- PLA. Programmable Logic Array
- MCU, Microcontroller unit ✔

3: Critical path is...
Select one alternative

- The longest possible chain of combinational logic between two registers ✔
- The path through all registers in a module
- The path through all registers in a design
- The shortest path between two registers
- impossible to change

4: Which of these are **not** considered an VHDL design entity?
Select one alternative

- Configuration
- Entity
- Architecture
- ASM-diagram ✔
- Package body

5: Code written at register transfer level (RTL) describes...
Select one alternative

- how registers are transferred in a module
- combinational logic only
- registers and the function of the combinational logic in a module ✔
- how registers shall work in a module
- how a simulation shall be run

6: Data flow code describes...
Select one alternative

- Gates (logical ports), and their connections ✔
- A block diagram
- How data are transferred between registers
- how a simulation flows
- Components and their connections

7: Code written at a structural level describes...

Select one alternative

- Components and their connections ✔
- Gates (logical ports), and their connections
- The variance of a component structure
- how a simulation flows
- registers and the function of the combinational logic in a module

8: Behavioral code describes...
Select one alternative

- A simulation or a non-synthesizable component ✔
- registers and the function of the combinational logic in a module
- Behavior of a syntheziable module
- VHDL architectures in general
- how registers shall work in a module

9: Generally VHDL is easiest to read when written using...
Select one alternative

- Endline Layout with UpperCamelCase
- Column layout with lowerCamelCase
- Column layout with underscore_case
- Bold layout with ALL_CAPS
- Block layout with underscore_case ✔

10: VHDL simulation is...

Select one alternative

- Cynical
- Cycle based
- Combinational
- Event based ✔
- Logical

11: A delta delay is...

Select one alternative

- the minimum time between two separate events
- the shortest nonzero delay in a simulation
- the same as an alpha delay, only longer
- used to separate events occuring at the same time in a simulation ✔
- shorter than an alpha delay

12: A D-flipflop...

Select one alternative

- has asynchronous reset
- reads indata when clock is high
- reads indata on a falling or rising clock-edge ✔
- reads indata when clock is low
- has synchronous reset

13: An System on chip with FPGA contains at least...

Select one alternative

- A system of look up tables and flipflops
- One or more microprocessor or microcontrollers connected to the FPGA-fabric ✔
- Block RAM and DSP modules
- A Zynq 7020 platform
- An AXI-bus and and ARM core

14: "SPI" as in the SPI-bus, is an abbreviation for...

Select one alternative

- Standard periphery interconnect
- Sequential peripheral index
- Serial Peripheral interface ✔
- Sequential particulate interconnect
- Serial Protection Interface

15: An SPI bus can have...

Select one alternative

- Several masters and several slaves
- One master and up to three slaves
- One master and one slave
- One master and several slaves ✔
- One master or three slaves

16: FPGA and ASICs: Which statement is true?

Select one alternative

- ASIC have a long development time but the first circuits are cheap to manufacture
- Prototyping ASIC on FPGA should be avoided because of the difference in coding style
- FPGA is better than ASIC when dealing with complex designs
- FPGA is more suitable than ASIC in products specified to have low power consumption
- A design implementation requires less chip area when placed on an ASIC rather than an FPGA ✔

17: A std_logic signal driven by two signals with values...

Select one alternative

- ...'U' and 'L' will be of the value 'L'
- ...'L' and '1' will be of the value '1' ✔
- ...'X' and '0' will be of the value '0'
- ...'L' and 'H' will be of the value 'U'
- ...'1' and 'Z' will be of the value 'Z'

18: What is correct about antifuse based FPGAs?

Select one alternative

- Antifuse FPGAs should not be used in space applications
- Antifuse FPGAs use more power than the other FPGA circuit technologies
- Antifuse FPGAs can easily change configuration
- Configuration is saved in the FPGA by making shorts using high voltage. ✔
- Antifuse FPGAs uses more area than those based on SRAM

19: What is correct about SRAM-based FPGAs?

Select one alternative

- they can be reprogrammed multiple times ✔
- they are generally a better choice than microcontrollers
- they have the disadvantage of low power usage
- they are one-time programmable
- they are not suited for prototyping

20: State machines: Which statement is true?

Select one alternative

- A Microcoded FSM will require more registers than non-microcoded FSMs
- RAM can be used to create Microcoded state machines ✔
- Microcoded FSMs does not require the use of registers
- The memory in a microcoded state machine must be one-time programmable
- A microcoded FSM will require fewer registers than non-microcoded FSM's

21: State machines: Which statement is true?

Select one alternative

- Microcoded microprocessors must have sequencers
- A sequencer for a microcoded state machine must not contain counters
- Microcode can not be used to create Mealy Machines
- Microcode can be used to create Mealy Machines ✔
- Microcoded FSMs are less reliable than other FSMs

22: Metastability: Which statement is true?

Select one alternative

- metastability never occurs in a well built system
- Metastability does not occur in sequential logic
- Propagating metastability will cause the power consumption in flipflops to drop
- Reading a metastable signal may yield a high, low or metastable outcome. ✔
- Metastability is the opposite of the probability of stability

23: The n-bit problem: Which statement is true:

Select one alternative

- Brute force synchronisers can be used with gray coded signals, provided that the receiving clock domain has the faster clock ✔
- Brute force synchronizers eliminate the n-bit problem
- Binary coded signals does not need brute force synchronizers when crossing clock domains
- Brute force synchronizers causes the n-bit problem
- Brute force synchronizers can sometimes be made with a single flipflop, for standalone signals.

24: FIFO synchronizers has a dual port RAM and...

Select one alternative

- uses pipelining for all data bits
- use gray code counters for read and write pointers. The pointers are brute force synchronized. ✔
- uses a five stage FSM to secure all data
- uses enable-synchronizers for all data bits.
- use brute force for all data bits

25: Flow control between a sender and a receiver can often be achieved using the signals valid and ready along with the data. Which statement is true?

Select one alternative

- The receiver is responsible for the valid signal
- The sender is responsible for the ready signal
- Pull flow control assumes the receiver is always ready
- Push flow control assumes the receiver is always ready ✔
- valid and ready is more important than the data

26: Data that are always valid...

Select one alternative

- Must use flow control
- is granulated
- never change
- Can be passed without flow control ✔
- Uses serialization

27: Isochronous timing is typically used for data such as music, speech and video. Which statements about data sent using isochronous timing are true?

Select one alternative

- Data-packets uses half the bitstream
- Uses the full bitstream bandwidth
- Data is sent with random time intervals
- Data is sent with regular time intervals ✔
- Data is brute force synchronized

28: Pipelining: Which statement is true?

Select one alternative

- Pipelining decreases latency in a system, because the clock frequency can be higher
- Asynchronous pipelines are superior to sequential pipelines
- Pipelining requires more resources than parallelization
- Stalling a pipeline requires some sort of flow control ✔
- Pipelining doubles the throughput of a non-pipelined module

29: Which of these parameters is generally most important in digital design?
Select one alternative

- Portability - The ability to be shifted between environments
- Reuseability- the ability to be used in other places than it was designed for
- Efficiency- The ability to perform tasks with less energy or area use
- Speed - the ability to perform a task quickly
- Verifiability - the ability to be read and tested ✔

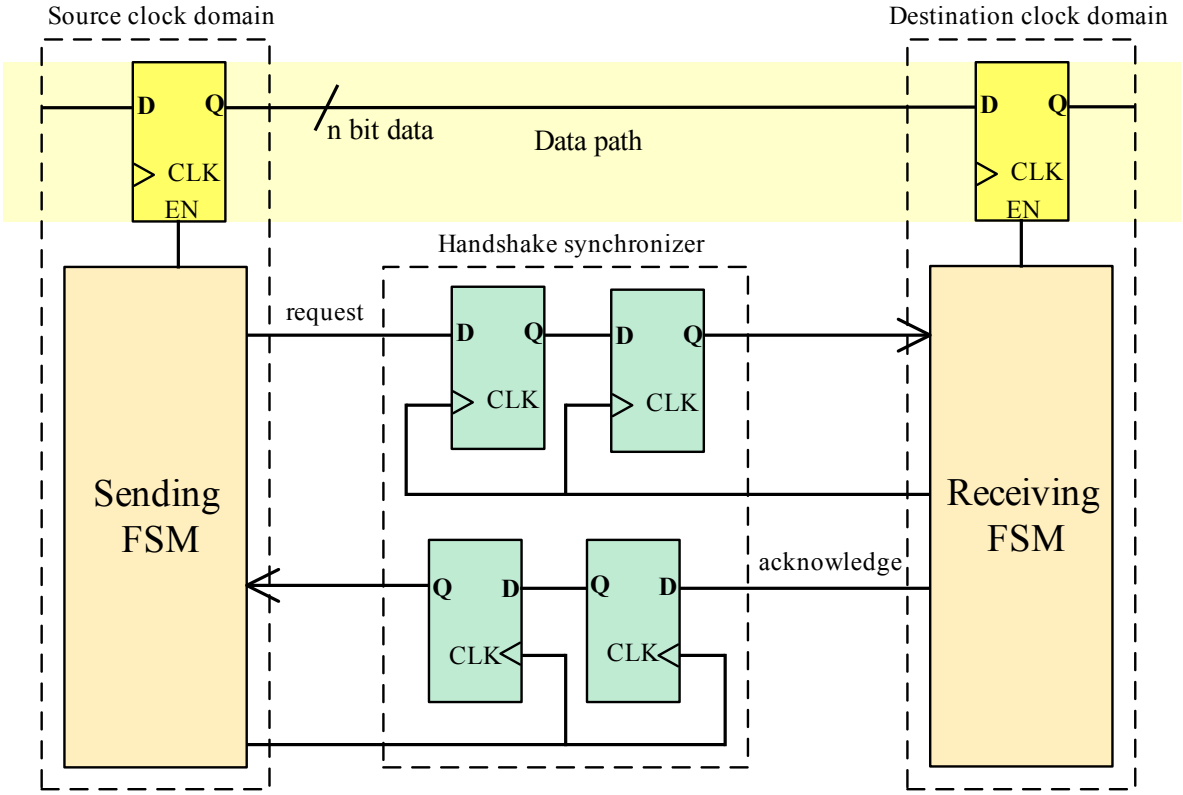
30: What is the purpose of a fault injection algorithm in a test bench?
Select one alternative

- To stir up the development team when they discover there are errors.
- To correct other faults in the DUT module
- To verify that the test bench is able to report fault conditions ✔
- To test the DUT for faulty behavior
- To verify that the test bench does not report wrong conditions

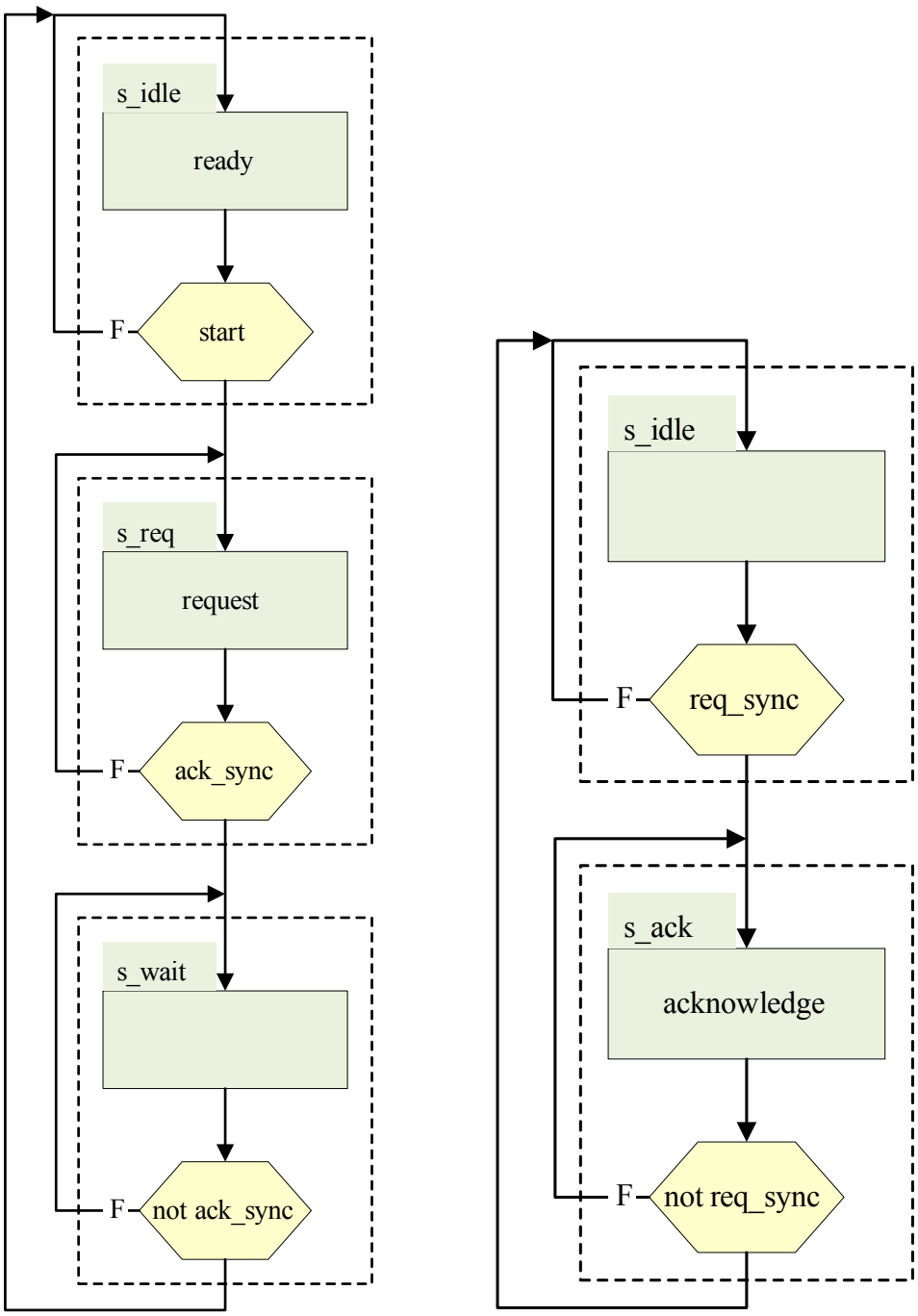
Maximum marks: 30

3 Diagrams

We have a system consisting of three modules; a sender "talker", a receiver "listener" and a handshake synchronizer, as shown in the block-diagram below:



The state diagram for the sender and receiver are as follows:



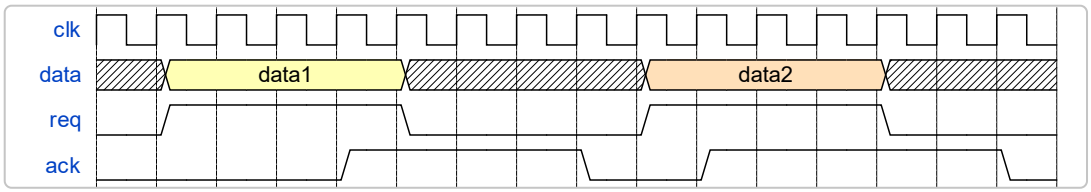
Below are timing diagrams depicting the process of sending data in the system.

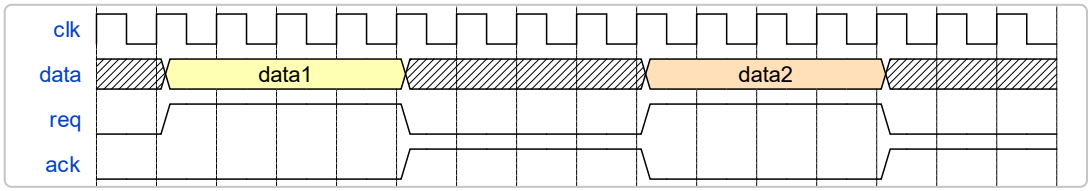
For simplicity, we assume the clocks in each domain use approximately the same frequency, and for the time being they are synchronous. The start signal is active at all times.

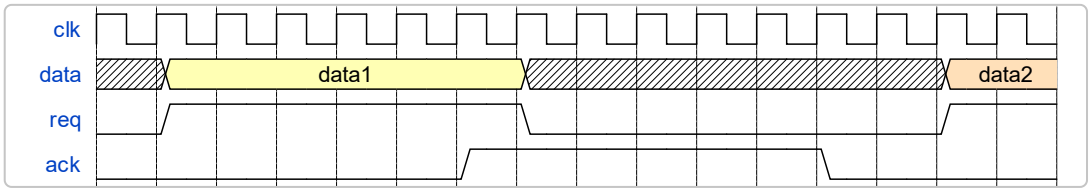
The timing diagrams uses the same name for synchronized and non-synchronized signals. It is your task is to place the diagrams according to what each module sees. A module does only see its own in and outputs, e.g. the talker does only see the synchronized version of the acknowledge signal, and the unsynchronized version of the request signal. The data is shown as valid from the viewpoint for each module- actual data validity may vary.

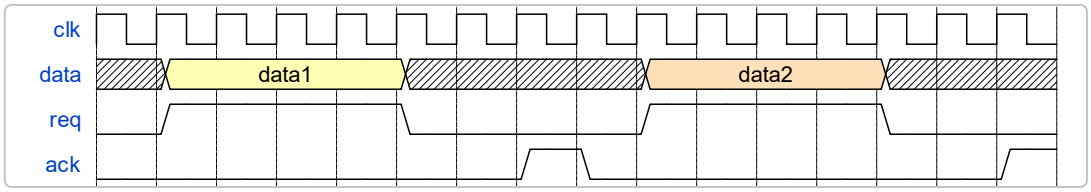
Place the correct diagrams in the slots for the talker, sender and the inputs of the handshake synchronizer. *The slots are located below the alternatives.*

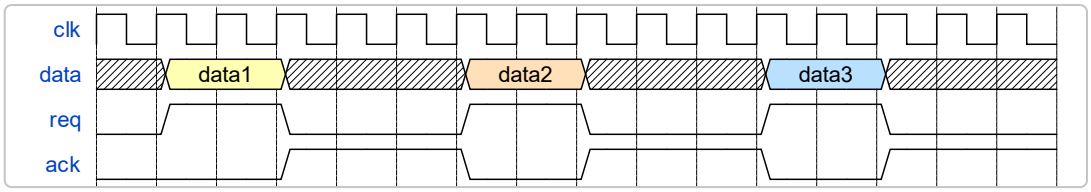
Help

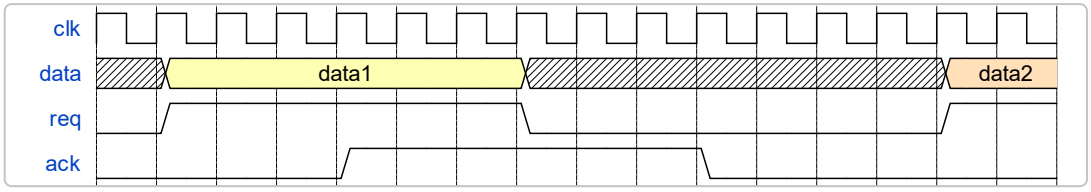

clk
data
req
ack
data1
data2

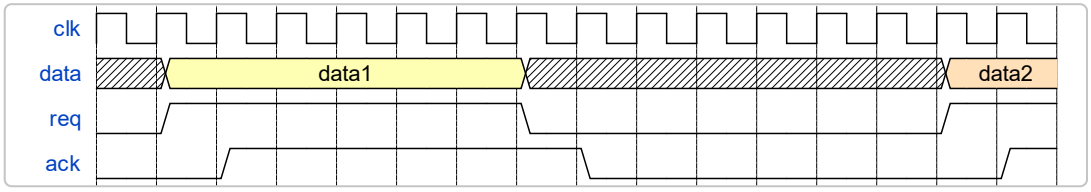

clk
data
req
ack
data1
data2

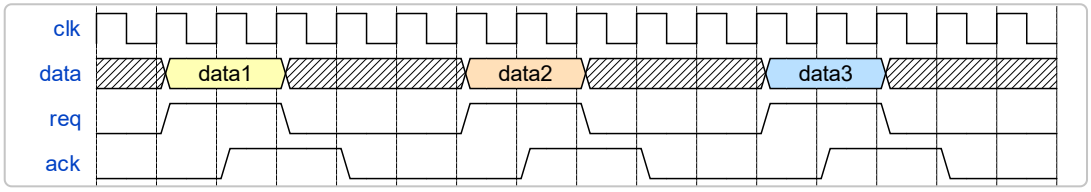

clk
data
req
ack
data1
data2

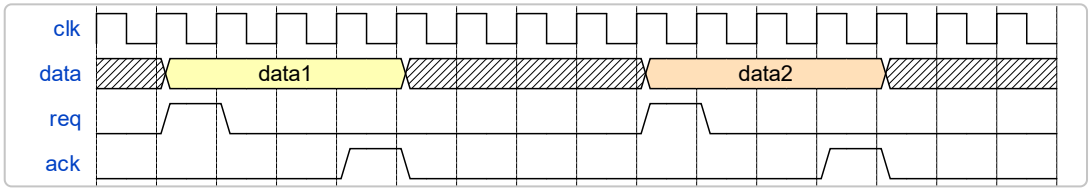

clk
data
req
ack
data1
data2

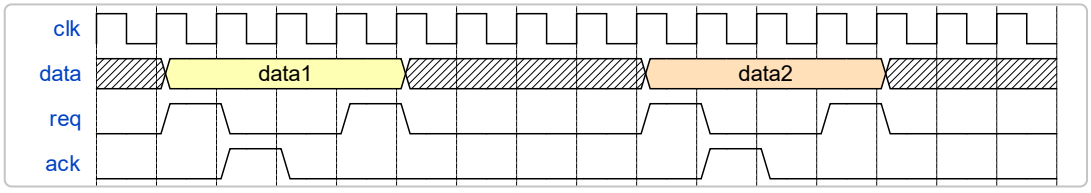

clk
data
req
ack
data1
data2
data3


clk
data
req
ack
data1
data2

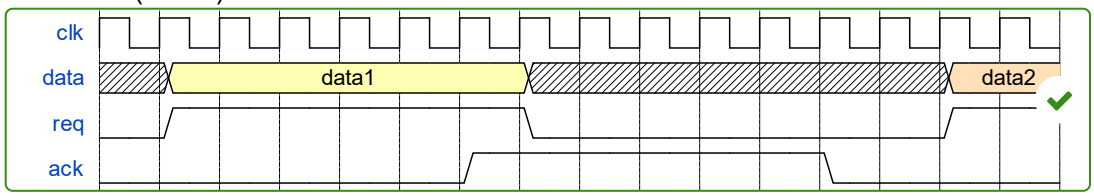

clk
data
req
ack
data1
data2


clk
data
req
ack
data1
data2
data3

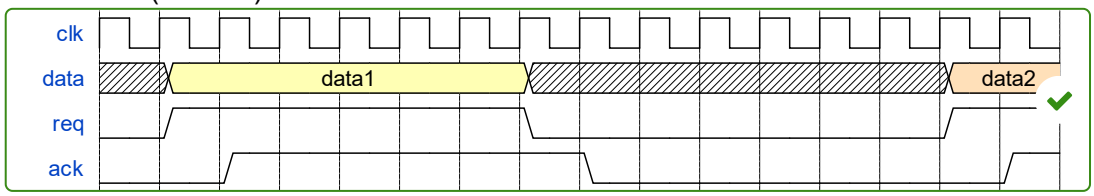

clk
data
req
ack
data1
data2


clk
data
req
ack
data1
data2

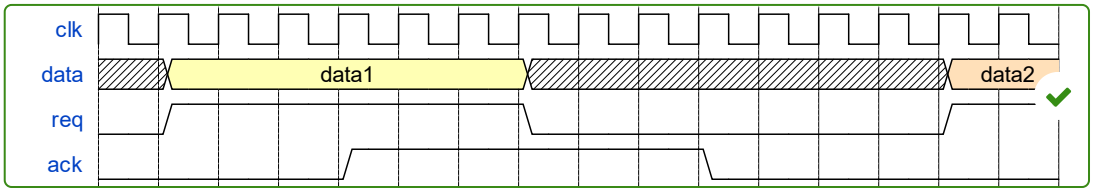
The sender (talker) sees:



The receiver (listener) sees:

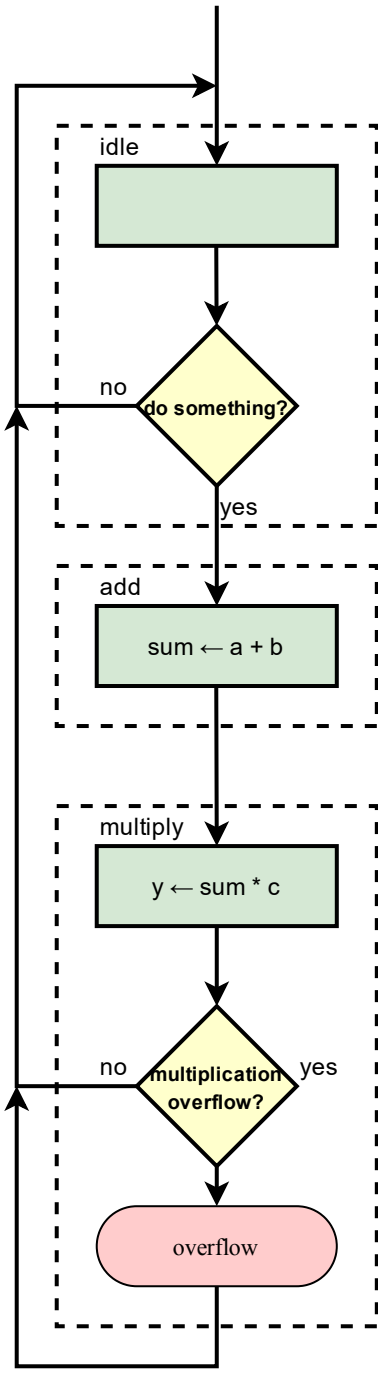


The handshake synchronizer sees:



Maximum marks: 6

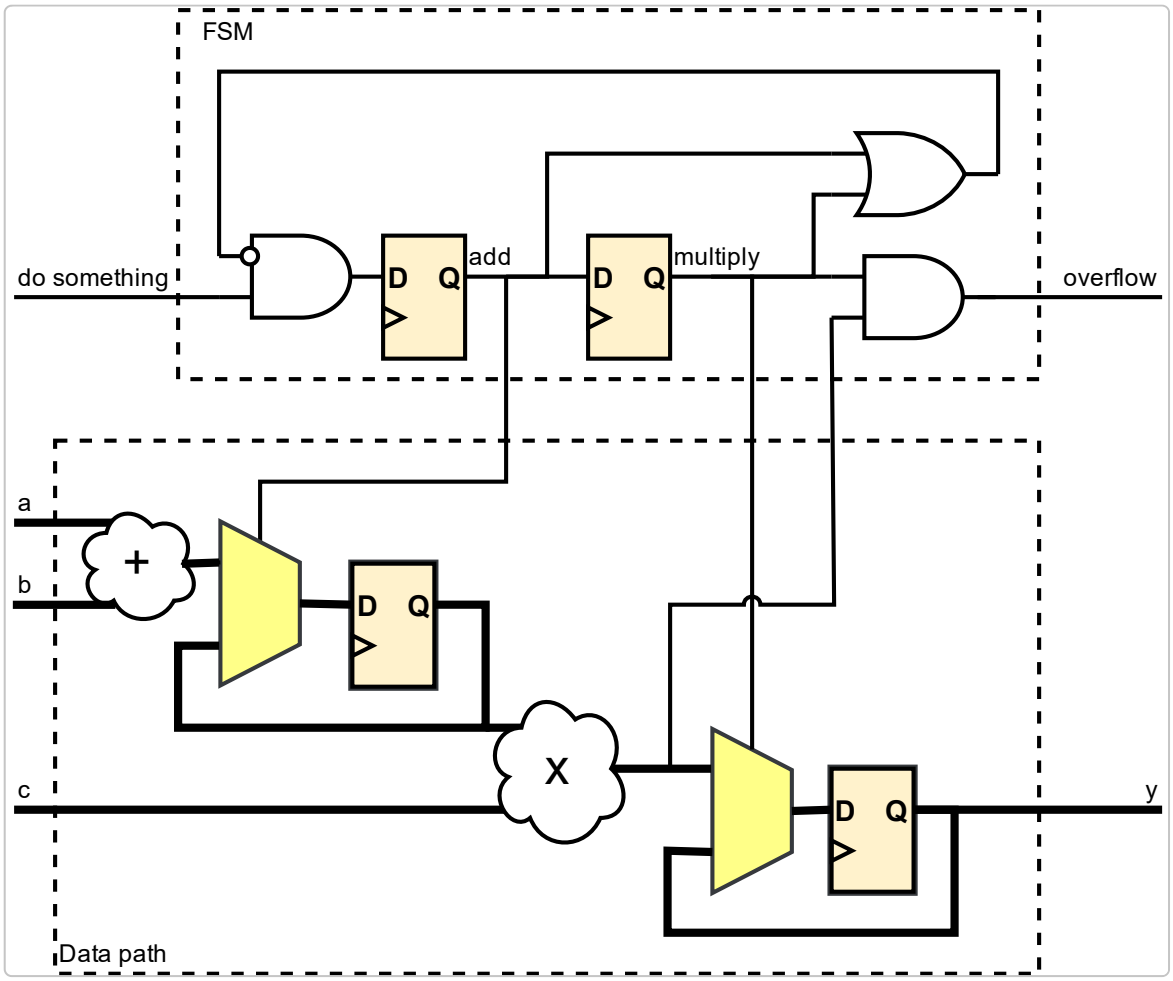
4 ASMD and datapath diagram

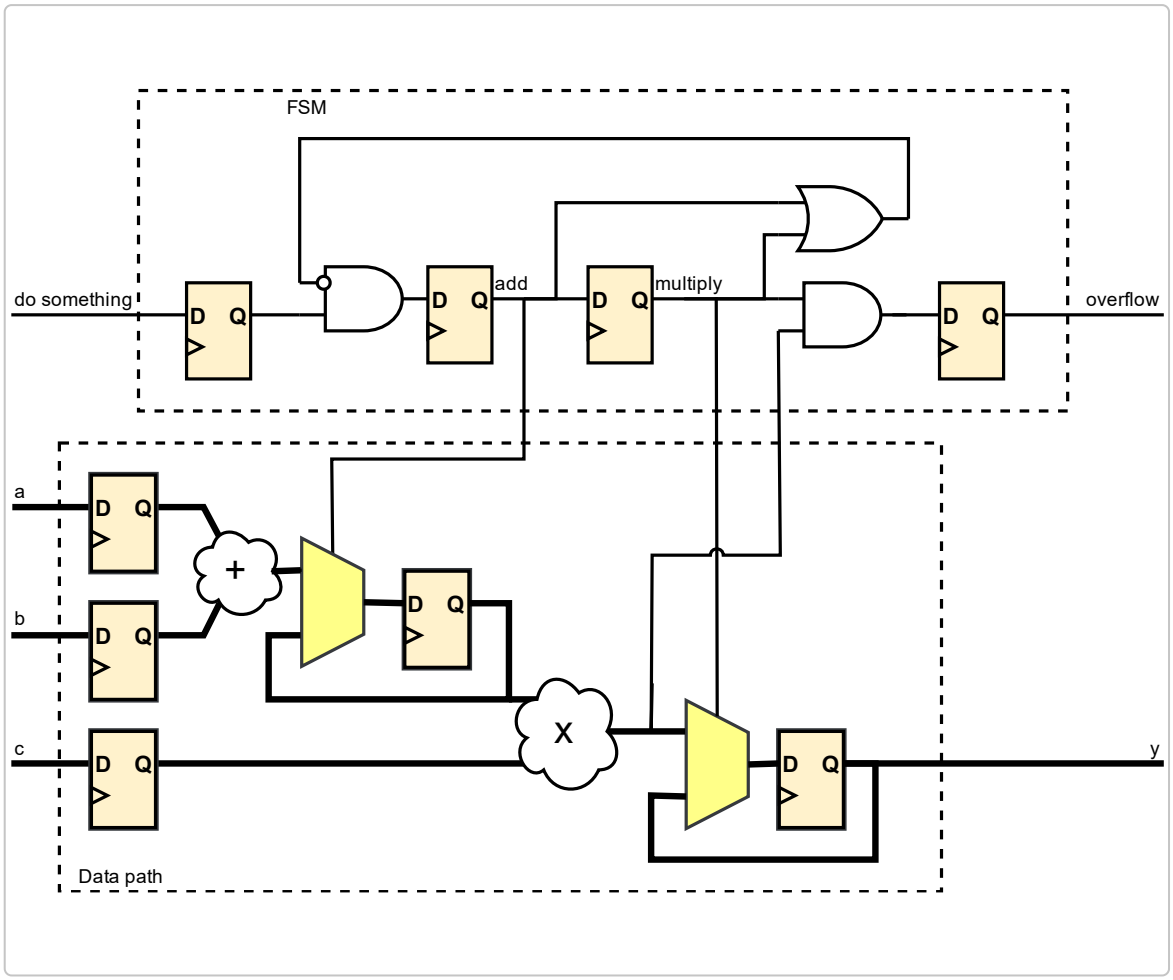


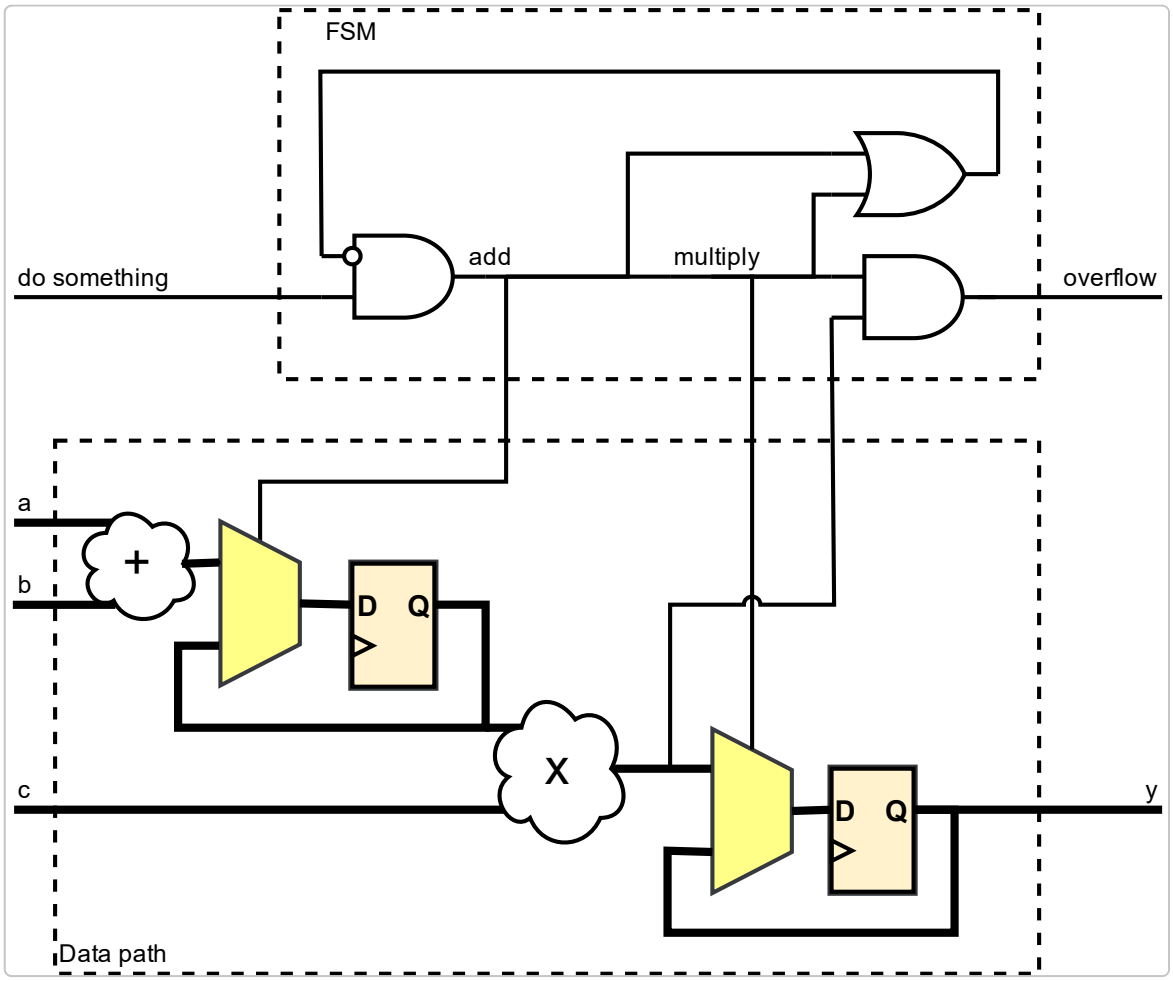
Which of the following datapath-diagrams shows a valid representation of the ASMD diagram above?

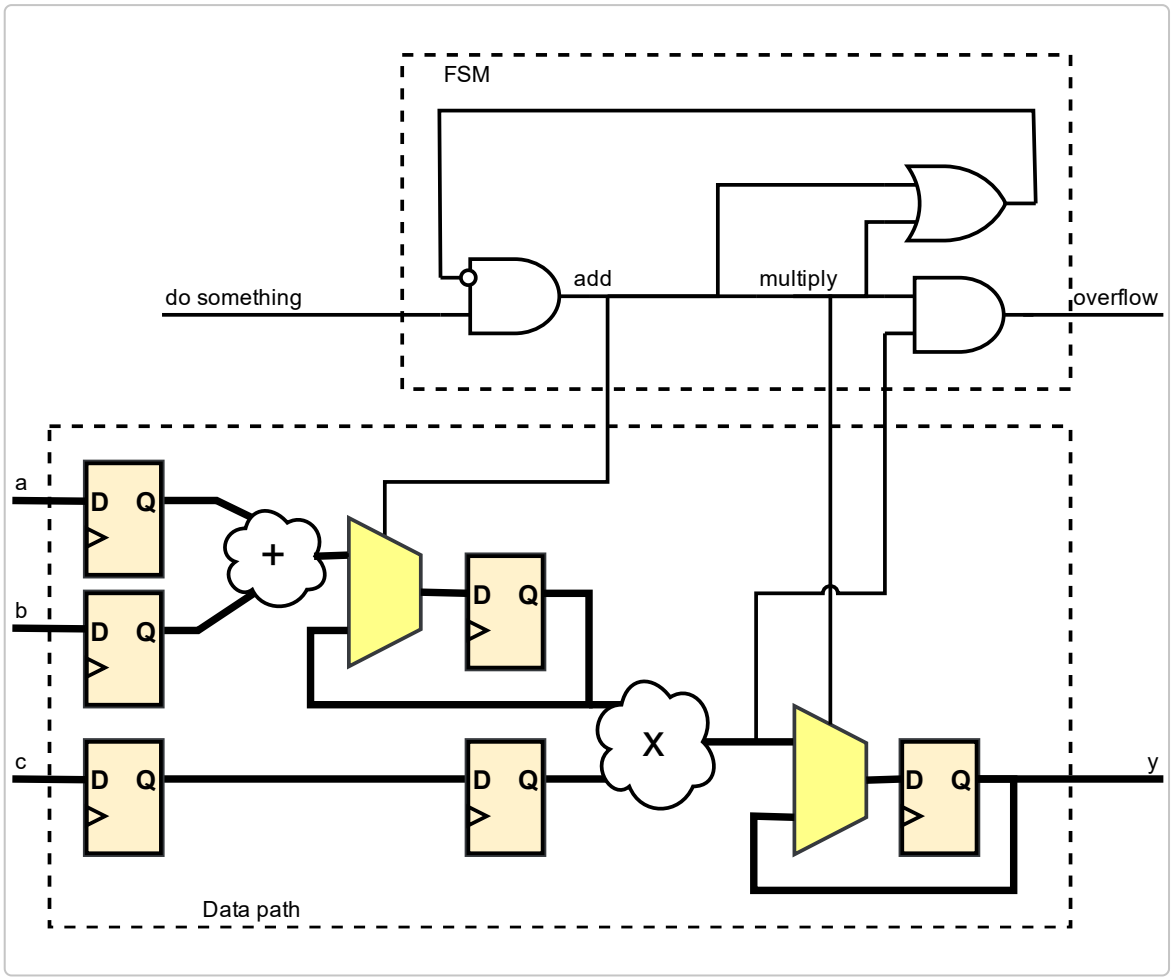
Place the corresponding diagram in the slot at the bottom of this exercise:

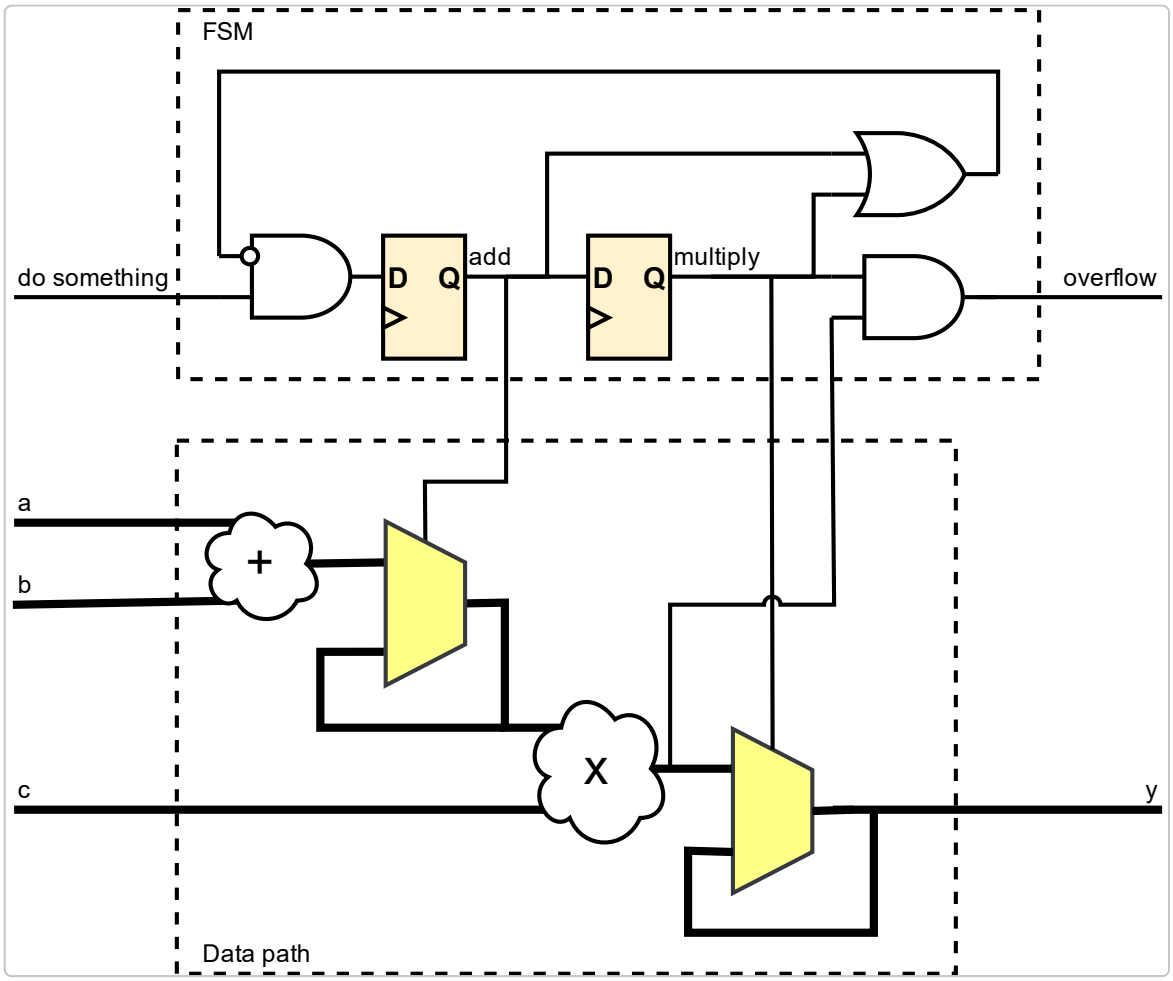
Help


FSM
do something
D Q
add
D Q
multiply
overflow
a
b
+
D Q
c
X
D Q
y
Data path

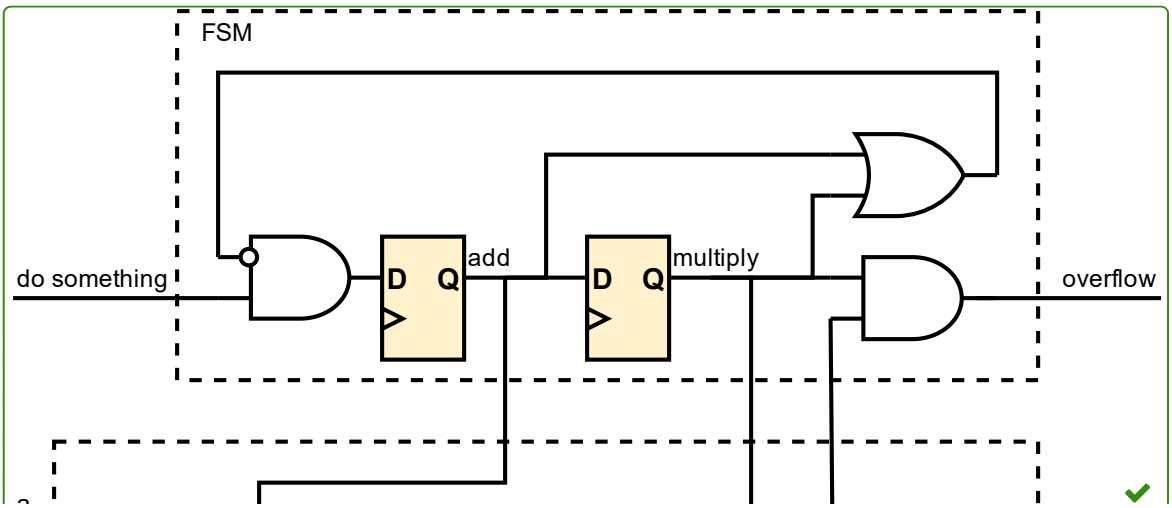

FSM
do something
D Q
D Q
add
D Q
multiply
D Q
overflow
a
D Q
b
D Q
+
D Q
c
D Q
X
D Q
y
Data path


FSM
do something
add
multiply
overflow
a
b
+
D
Q
X
c
y
Data path


FSM
do something
add
multiply
overflow
a
b
c
D
Q
+
X
y
Data path



Place the corresponding diagram here:



Maximum marks: 2

5 Datapath diagram

In this assignment, a synchronous two stage pipelined module *compute_pipelined* which computes *(a+b)*c* shall be implemented.

The computation shall use signed arithmetic operation.

The inputs (*a,b,c*) and the output (tdata) shall be standard logic vector. The vector length (for *a,b,c* and *tdata*) shall be specified based on a generic in the entity. The default value shall be 10.
The result shall have the required number of bits.

In additon, the input signal, *vdata*, determines when the inputs *a, b and c* have valid data (i.e. *vdata='1'* when *a, b* and *c* are valid).

The output signal *tvalid* shall be '0' when the output result is not valid and '1' when the output signal result is valid. When the computed output, *tdata,* is not valid the value shall be '0'.

The pipelined architecture shall allow new data each clock cycle.

Draw a datapath diagram of the of pipelined architecture according to the above description.
The datapath diagram shall show the names and bit widths of the signals.

In this exercise you can answer with digital hand drawing. Use your own sketch sheet (distributed). See instructions for filling in the sketch sheet in the link below the task bar.

Maximum marks: 10

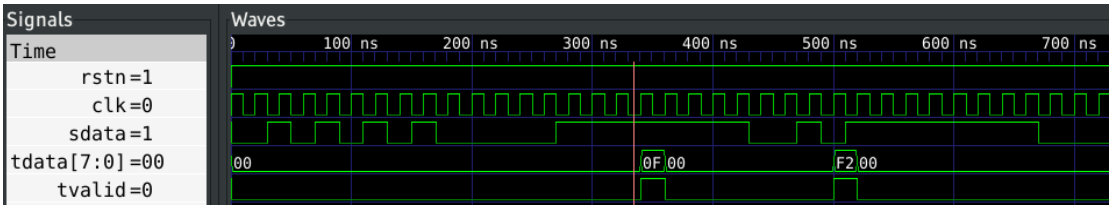
6 Pipeline implementation

Implement the architecture from the previous assignment in synthesizable VHDL-2008.

Fill in your answer here

```
1
```

Maximum marks: 15



In this task, a three process Moore type finite state machine (FSM) shall be implemented.

The FSM works as follows:

The FSM uses an active low reset signal (rstn).
The FSM detects the *sequence* 0xAA transmitted on a serial line (sdata).
After the sequence 0xAA is detected, the following *bytes* received shall be set on an output (tdata), and a data valid (tvalid) signal shall be set high for each valid byte.
If the *byte* 0xFF is detected, the FSM shall stop setting tvalid, and wait for another 0xAA sequence.

The timing diagram (above) shows an example of input and output values of the FSM module. In the example, the sdata input is 0xAA 0x0F 0xF2 0xFF.

7 ASM Diagram

Draw an ASM-diagram which depicts the finite state machine described above.

In this exercise you can answer with digital hand drawing. Use your own sketch sheet (distributed). See instructions for filling in the sketch sheet in the link below the task bar.

Maximum marks: 10

8 VHDL Implementation

Implement the FSM as a three process state machine in synthesizable VHDL-2008.

Fill in your answer here

```
1
```

Maximum marks: 13

9 Test bench

Implement a VHDL-2008 (or Python/cocotb) test bench that verifies and reports correct behaviour according to the task description.

Use the following test vectors: 0xAA 0x0F 0xF2 0xFF 0x02 0x03

Fill in your answer here

```
1
```

Maximum marks: 12