Points	Grade

1327039 Philipp Lehninger
1027440 Benedikt Morgenbesser
...

Digital Integrated Circuits Lab (LDIS)

384.088, Summer Term 2019

Supervisors:
Christian Krieg, David Radakovits, Axel Jantsch

Task 2:

Design Characterization, Bus Communication

1 Characterize your design from Task 1

In order to make your design implementation comparable to other implementations, simulate your design using Vivado and perform the following measurements:

1. Timing analysis
2. Power analysis
3. Resource consumption

Create a design space vector for your design.The design space vector holds the following parameters, where t_{max} is the maximum delay for the critical path, P_{avg} is the average power consumption for your design (vector-less post-place-and-route power estimation), and r is the percentage of resources used in your implementation for the Nexys 4 DDR board (for sake of simplicity, use the percentage of slices):

$$\vec{v} = \begin{bmatrix} t_{max} \\ P_{avg} \\ r \end{bmatrix} \tag{1}$$

You will find information on power estimation[1] and timing analysis[2] on the web.

Get the design space vectors of your colleagues, and visualize the three-dimensional design space. Use black crosses for the vectors of your colleagues, and a filled circle for your own vector.

2 Create an AMBA APB interface

The goal of this subtask is to make your design accessible from other intellectual property (IP) cores via the advanced microcontroller bus architecture (AMBA) advanced peripheral bus (APB). Consult the APB specification,[3] and:

1. Implement a bus controller that implements the use case for your task
2. Implement a bus interface for any sub-component of your design from Task 1 (sampling, data processing, output).
3. Implement a module that takes the user input for runtime parameters and connect it to the bus

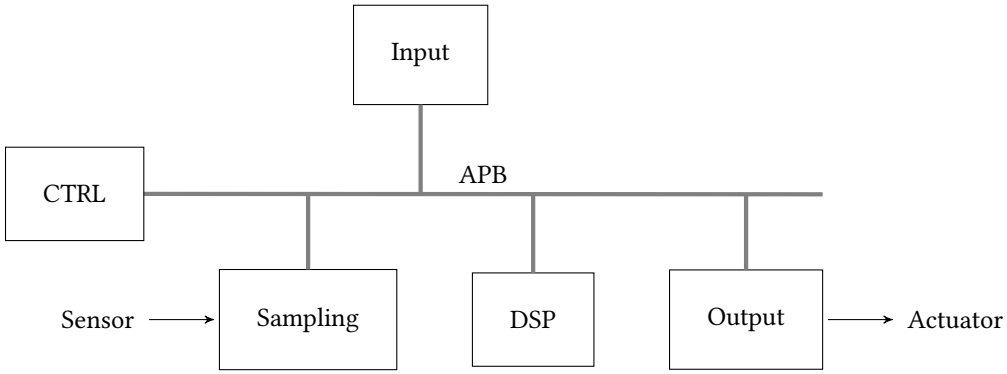

Figure 1: System of Task 1 equipped with an APB

Your system should be able to take the user input, parametrize the system according to the user input, and perform the actual task. Instead of directly connecting each of the sub-components directly, the modules should communicate over the APB. The audio system can be treated as one module; as it is kind of a real-time system, it wouldn't make a lot of sense to let it communicate over the APB. The user-input (e.g., cut-off freuqency) should be set over the APB.

[1] **xpe**.
[2] **timing**.
[3] **apb**.

3 Task 1 Characterization

1. **Timing analysis** For the timing analysis we have constrained the derived clocks with the *Constrains Wizard* and performed a *report_timing* command in the *TCL console*. Data Path Delay: $t_{max} = 5.994\,\text{ns}$.
2. **Power analysis:** The average estimated power consumption can be found under *Implementation -> Report Power*. A screenshot of the power report is shown in figure 2. For our design $P_{avg} = 0.1\,\text{W}$.
3. **Resource consumption:** We have used the command *report_utilization* in the *TCL console* for a utilization report. We have calculated the sum of all slice logic types used and divided this by the sum of all available slices. This gives a resource consomption $r = 1.6\,\%$.

$$\vec{v} = \begin{bmatrix} 5.994\,ns \\ 0.1\,W \\ 1.6\,\% \end{bmatrix} \tag{2}$$

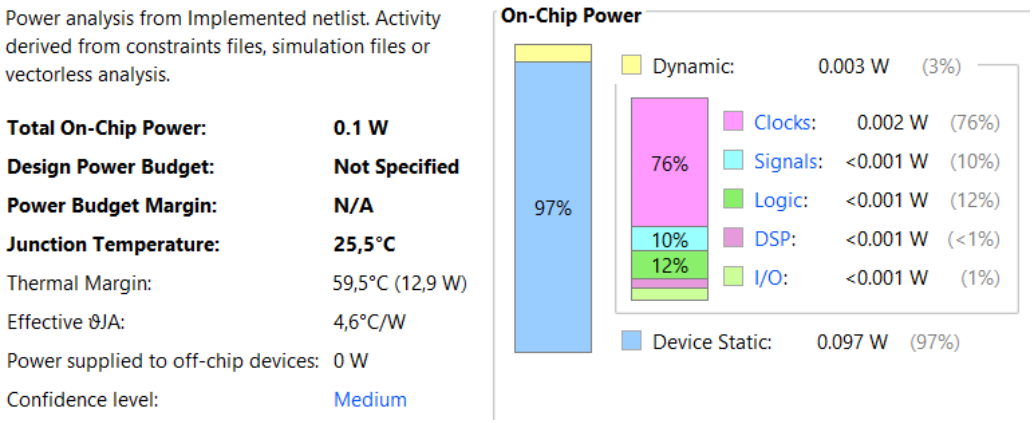

Figure 2: Power report for task 1.

1. Slice Logic

Site Type	Used	Fixed	Available	Util%
Slice LUTs	3448	0	63400	5.44
LUT as Logic	3448	0	63400	5.44
LUT as Memory	0	0	19000	0.00
Slice Registers	402	0	126800	0.32
Register as Flip Flop	402	0	126800	0.32
Register as Latch	0	0	126800	0.00
F7 Muxes	0	0	31700	0.00
F8 Muxes	0	0	15850	0.00

Figure 3: Utilization report for task 1.

A 3D visualization of the design space with design vectors from our colleagues (crosses) and our design vector (red circle) is shown in figure 4.

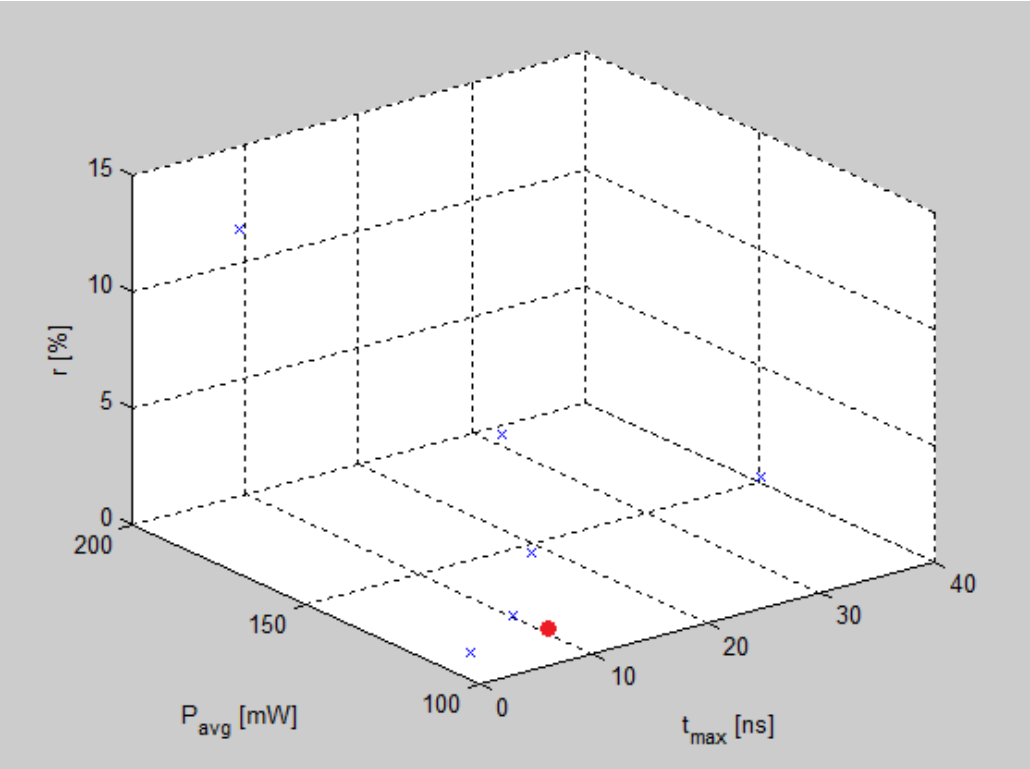

Figure 4: Design space task 1.

4 AMBA APB Interface

4.1 Design Decisions

We have implemented a general AMBA ABP bus master and slave module. Each sub-component of our task 1 design includes an instance of the slave. In the top level design a state machine controls the bus via the master.

The AMBA ABP bus clock is set to 25 MHz.

The state machine in the top level design consits of the 6 following sequential states:

1. **IDLE:** This state is just used as an initial state for the system after power-up or resets.
2. **INPUT:** The user input for the size of the averaging window is read from the Input Module.
3. **SENSOR:** The 16 bit 2's complement vector from the Sampling Module is read.
4. **FEEDMAVG:** The temperature value and window size are concatenated and written to the Moving Average Module.
5. **MAVG:** The averaged temperature is read.
6. **OUTPUT:** The averaged temperature vector is written to the Output Module. This state is followed by the INPUT state.

A state transition only occurs when the *m_ready* flag is set by the bus master (Except for the IDLE state).

All slaves have to share the PRDATA and PREADY port of the master. In order to read from the slaves we demultiplex these signals.

We have planned to use bi-directional bitvectors for the communication between top level design and bus master, respectively bus slaves and their corresponding sub-components. (Supported read and write operations). However, the synthetization of the design led to critcal warnings. Therefore we changed the communication between the slaves and their sub-components to uni-directional bitvectors.

The AMBA ABP specification intends the usage of an adress bus. Although, we have implemented this feature we

don't need it, since the 32 bit data bus is wide enough.

4.2 Testing

Besides testbenches for bus master and bus slave, we have built a testbench, which simulates the state machine of the top level design including all AMBA ABP bus communication. This testbench shows satisfying results. The waveform diagram is shown in figure 5 & 6.

Figure 5: Waveform from our testbench part 1.

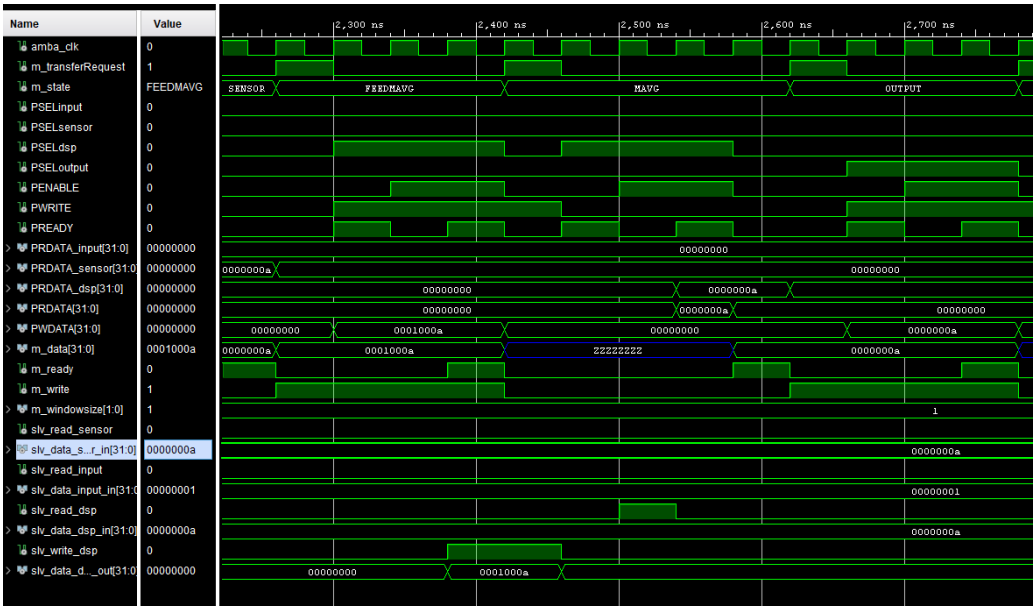

Figure 6: Waveform from our testbench part 2.

Waveforms for reading and writing operations from the AMBA ABP specification are shown in figure 7 & 9. A waveform from a reading and a writing operation of the master are shown in figure 8 & 10.

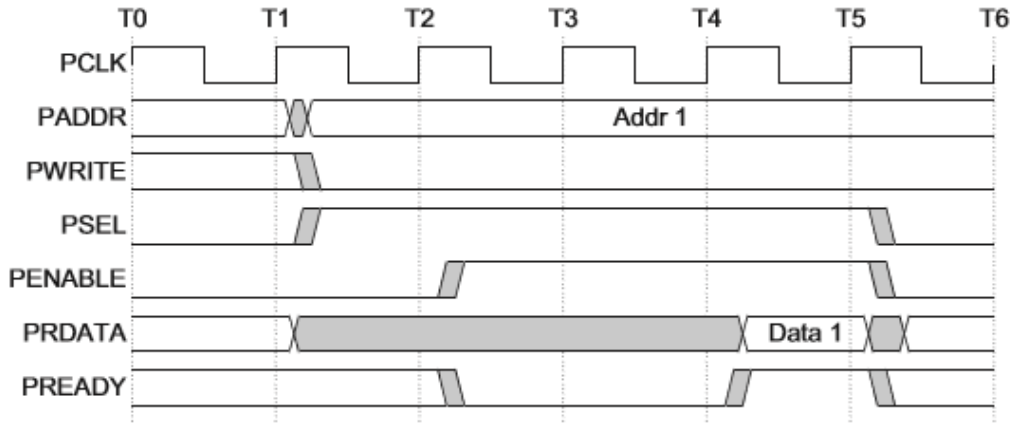

Figure 7: Reading specification for AMBA ABP.

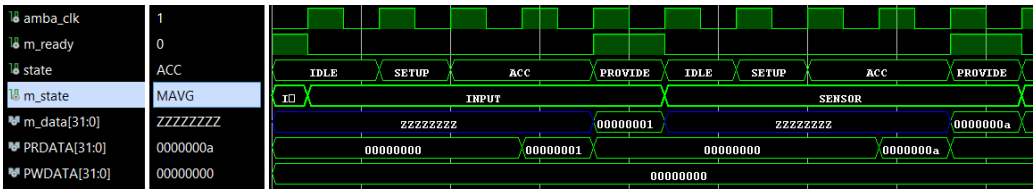

Figure 8: Waveform for master reading from slave.

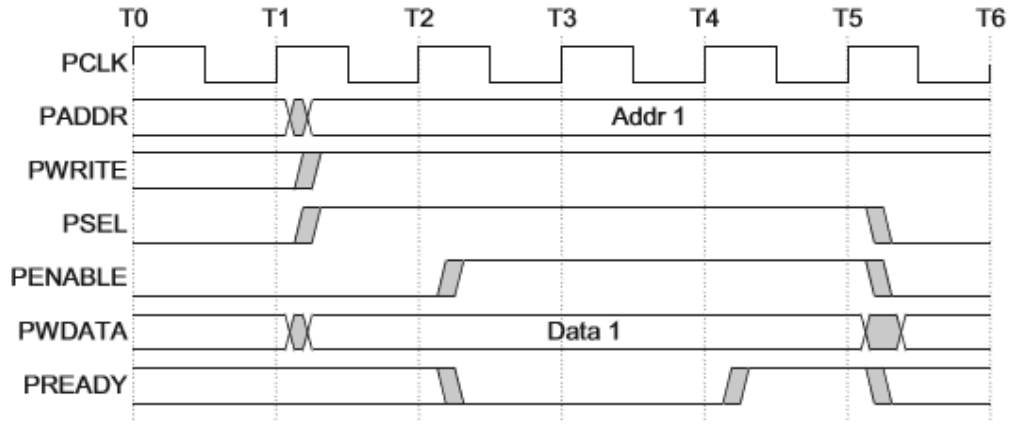

Figure 9: Writing specification for AMBA ABP.

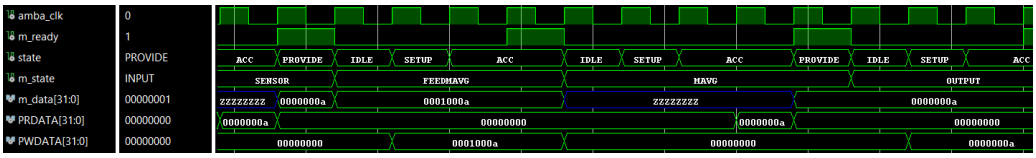

Figure 10: Waveform for master writing to slave.