Hardware RMS Scheduler

Thomas Gaul, Jackson Hafele

CPR E 558 Final Project Fall 2023

IOWA STATE UNIVERSITY

Problem Statement

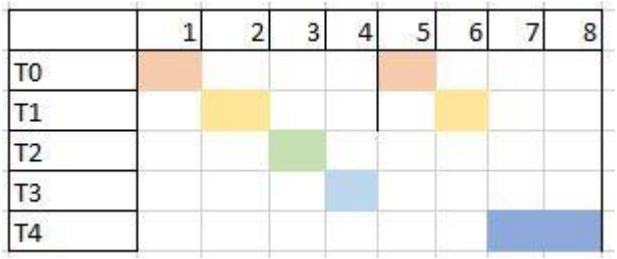
- Software implementation of task schedulers can require large amounts of overhead
- May not be ideal of low level embedded systems application with limited amounts of resources
- Alternative is to implement a hardware scheduler with either FPGA or ASIC for faster context switching or lower amount of power consumption

Solution

Schedule Implementation

- Implement static RMS Schedule
- Tasks refresh based on static period (Pi)
- Tasks take N time units to complete based on computation time (Ci)
- Highest priority goes to task with smallest period
- Sample task set created to model on Hardware

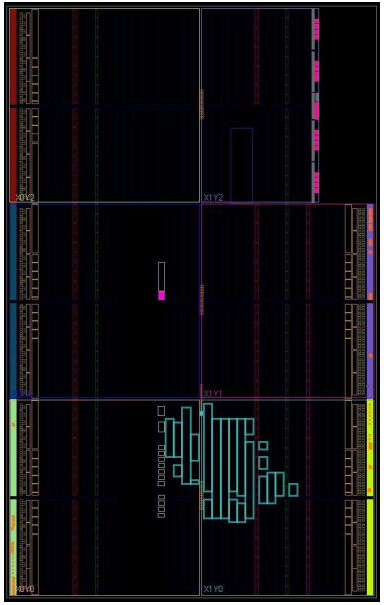
Task	Period (Pi)	Computation Time (Ci)
T0	4	1
T1	4	1
T2	8	1
T3	8	1
T4	8	2



Solution

Hardware Application

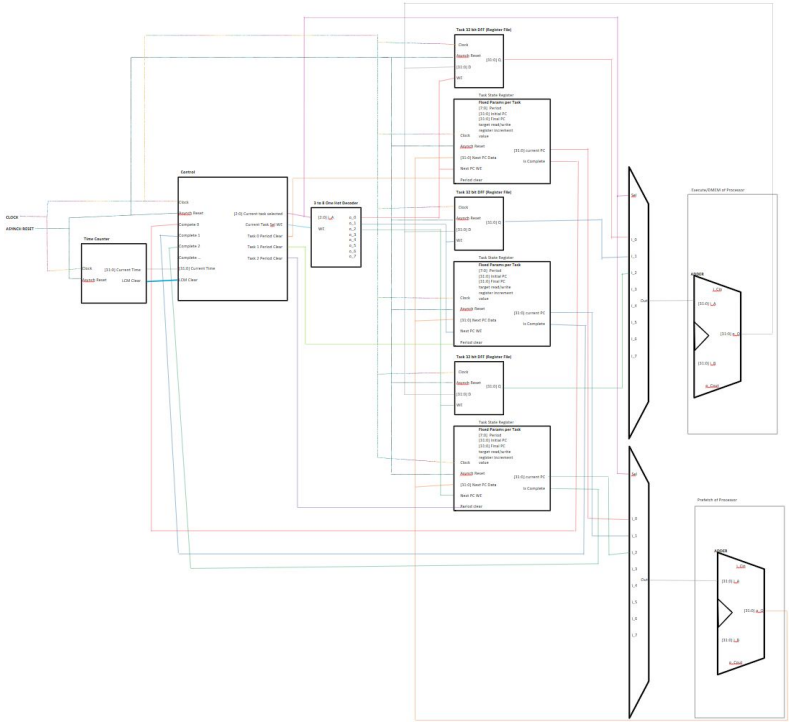
- Model RMS Schedule on Hardware with fixed parameters
- Repeat indefinitely while running periodic task set
- **To create**
  - Implement register to store data of each task
  - Implement control module for task context switching
  - Implement decoding between tasks based on completion and priority



FPGA Utilization Report

Implementation details

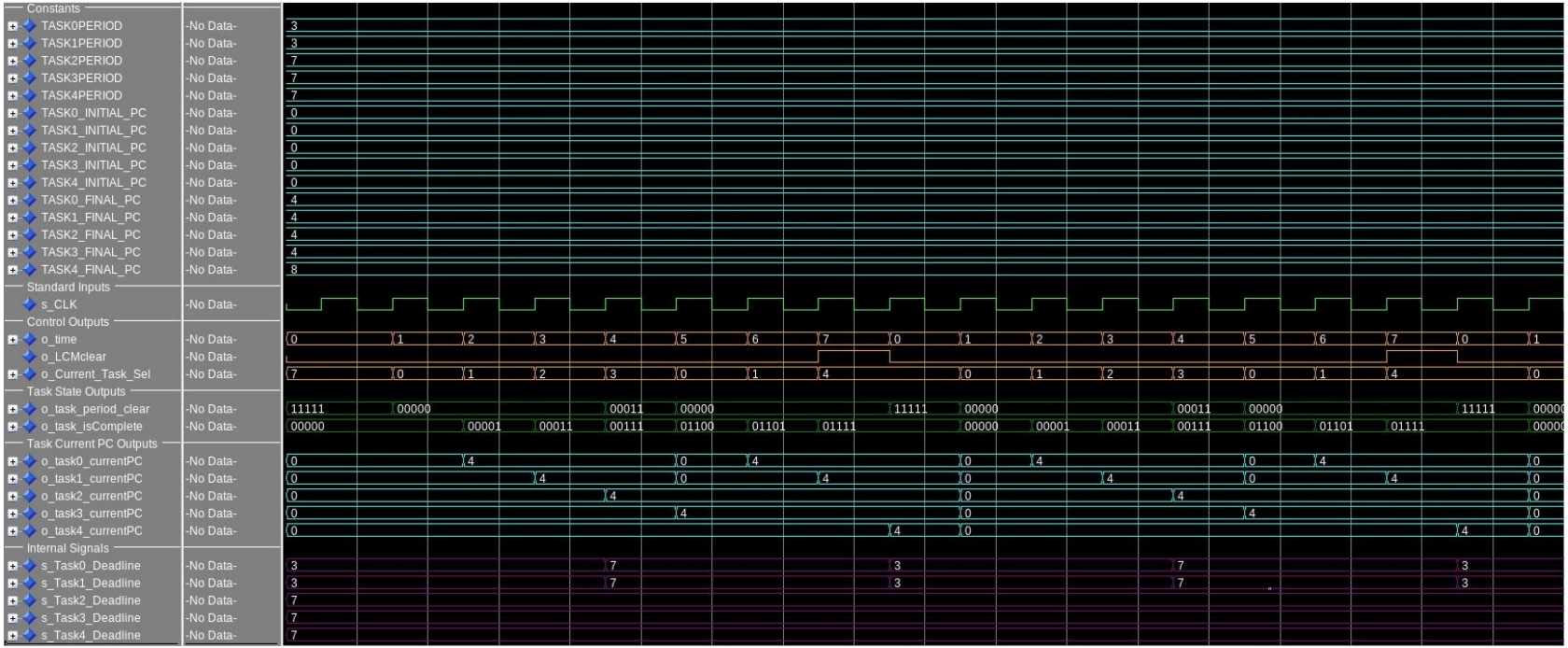
- Tools
  - Verilog/VHDL
  - ModelSim on the Coover Linux VDI servers
  - Vivado
  - Xilinx FPGA Arty Basys 3 Development Board



Implementation details

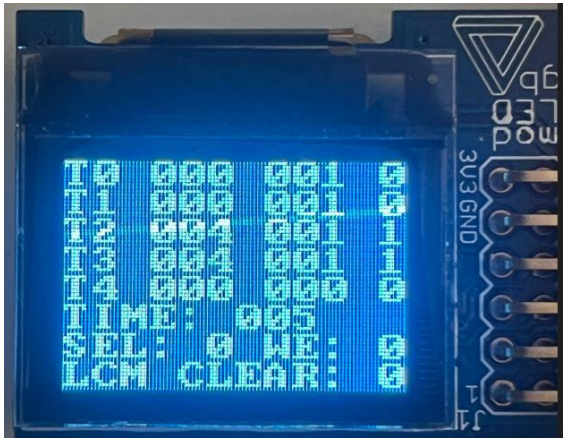
- **Time Counter**
  - tracks of elapsed time to ensure tasks are reset at the correct deadline
- **Control**
  - Selects which task to be active with RMS scheduling based on the period and completion status
- **3 to 8 One Hot Decoder**
  - Controls the write enable for each tasks register file and task state register
- **Task Register File** (32bit DFF)
  - Each task has its own register file that interface with the ALU to allow for fast context switching
- **Task State Register**
  - Tracks each task in terms of completion and its program counter and resets to the beginning of the task at the beginning of its period

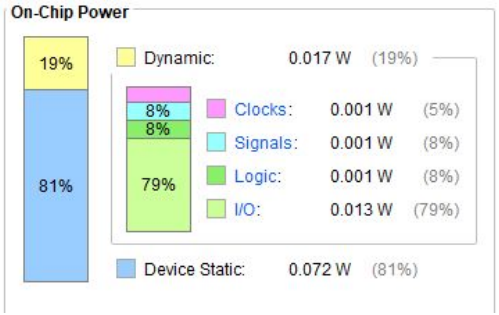
Testing/Evaluation Results



Testing/Evaluation Results

- **Advantages:**
  - **No OS overhead(64 cycles + for context Switching)**
- **Cost (Vivado)**
  - Worst Negative Slack: 3.937ns
  - Total Power Consumption: 0.089W
  - Look Up Tables Used: 1120
  - Flip Flops Used: 1122





Conclusions

- Successfully modeled sample RMS schedule with waveform results and FPGA synthesis
- Applied topics from related courses such as CPRE 281, CPRE 381, and CPRE 581
- Created platform for further work with other real-time schedulers or dynamic properties
- **Future Work**
  - EDF Implementation
  - Aperiodic Events
  - Integration with full processor

Learning Achieved through the project

- Applied RMS schedule to hands-on FPGA application
- Used real-time systems topics in a unique fashion
- Utilized experience gained from other digital design classes for FPGA application in new context
- Strengthened hardware design language skills
- Learned about research in the area of hardware for real time systems