Cairo University

Faculty of Engineering

Systems and Biomedical Department

2nd term - 2022/2023

Digital Electronics Project

“Round Robin Arbiter“

Under the supervision of :

Dr. Mohamed Islam

Eng. Samar Taher

TEAM 13:

Ziyad Hossam Masoud

Omar Abd El-Nasser

Mahmoud Magdy Ismail

Mahmoud Mohamed Ali Ahmed

Project preview

A Round Robin Arbiter is a digital circuit used for arbitration in computer systems, especially in shared resources where multiple requesters contend for access.

The purpose of a Round Robin Arbiter is to allocate access to the shared resource in a fair and deterministic manner. It works by rotating through a sequence of requesters and granting each requester access for a fixed time slice or priority level before moving on to the next requester. This ensures that each requester receives an equal share of the resource over time.

The Round Robin Arbiter typically maintains a state or counter that keeps track of the current requester being serviced. When a new request arrives, it is added to the rotation queue, and the arbiter grants access to the current requester until its time slice expires or until the requester completes its operation. Then, the arbiter moves on to the next requester in the queue, repeating the process.

Circuit applications:

The Round Robin Arbiter has several applications and offers advantages and disadvantages, as outlined below:

Applications:

1. Bus Arbitration: In computer systems with a shared bus, the Round Robin Arbiter can be used to allocate bus access fairly among multiple devices or modules.

2. Shared Memory Controllers: In systems with shared memory, the Round Robin Arbiter can be employed to allocate access to the memory among multiple requesting devices or processors.

3. Task Scheduling: Round Robin scheduling algorithms can utilize Round Robin Arbiters to allocate CPU time among multiple processes or threads in an operating system.

How the circuit work:

Inputs:

clk: Clock signal.

rst: Reset signal.

req: 4-bit input representing requests from four sources.

Outputs:

grant: 4-bit output representing the granted access to each source.

Registers:

state: 3-bit register representing the current state of the arbiter.

next_state: 3-bit register representing the next state of the arbiter.

Parameters:

s_ideal: Ideal state where no request is active.

s0, s1, s2, s3: States representing each source's turn to be granted access.

The module has three always blocks:

- The first always block updates the state based on the positive edge of the clock (clk) or the negative edge of the reset signal (rst). If rst is active low, the state is set to the ideal state (s_ideal). Otherwise, it takes the value of the next_state.

- The second always block is combinational and determines the next_state based on the current state, requests (req), and next_state. It uses a case statement to handle each state and sets the next_state based on the active request. If no request is active, it transitions back to the ideal state.

- The third always block is combinational and sets the grant output based on the current state. It uses a case statement to assign the grant value for each state. The default case sets grant to all zeros when the state is not explicitly defined.

The round-robin arbiter works as follows:

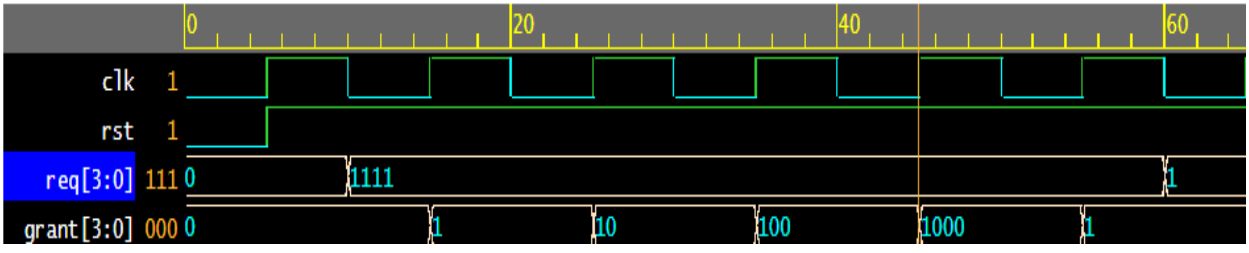
In the ideal state, it checks the requests sequentially and grants access to the source with the lowest index that has an active request. If no requests are active, it remains in the ideal state.

In each of the other states (s0, s1, s2, s3), it checks the subsequent sources and grants access to the source with the lowest index that has an active request. If no requests are active, it transitions back to the ideal state.

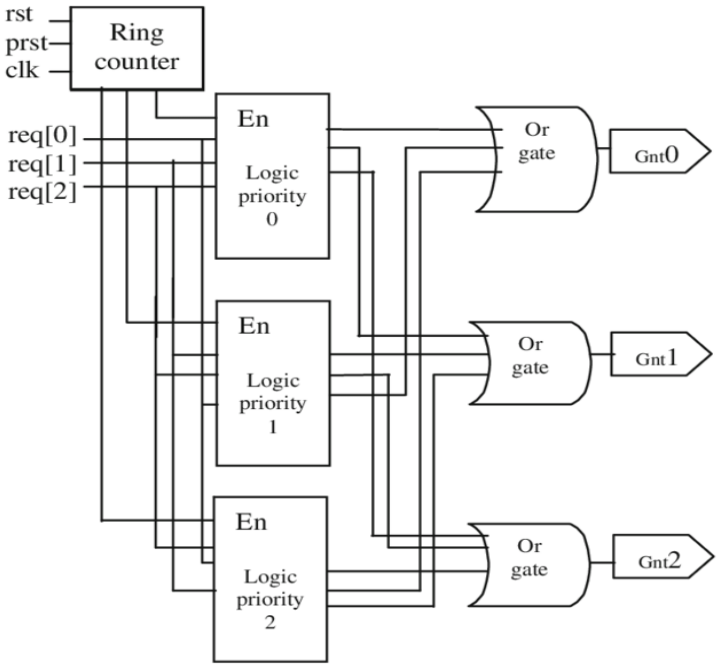
Note: The code assumes that multiple requests can be active simultaneously. If only one request can be active at a time, the code can be simplified.

Simulation result & state diagram :

1. Wave & Results

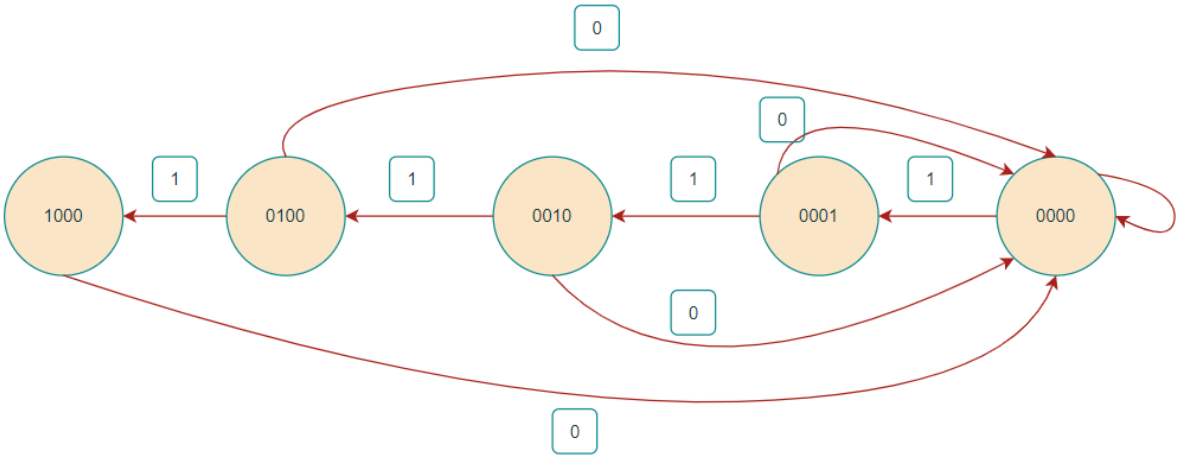


2. Circuit



---

3. State diagram & table



Current state	req[0]	req[1]	req[2]	req[3]
Ideal state	1	0	0	0
	0	1	0	0
	0	0	1	0
	0	0	0	1
S0	0	1	0	0
	0	0	1	0
	0	0	0	1
	1	0	0	0
S1	0	0	1	0
	0	0	0	1
	1	0	0	0
	0	1	0	0
S2	0	0	0	1
	1	0	0	0
	0	1	0	0
	0	0	1	0
S3	1	0	0	0
	0	1	0	0
	0	0	1	0
	0	0	0	1

Challenges & future work:

Challenges and Disadvantages:

- **Priority inversion:** Higher-priority requests delayed by lower-priority ones, leading to inefficient resource allocation.
- **Scalability:** Complexity and efficiency issues as the number of request inputs increases.

- **Clock synchronization:** Challenges in multi-clock domain systems, requiring proper synchronization techniques to ensure correct operation.

Solutions and Future Work:

- **Priority promotion:** Incorporate mechanisms for higher-priority requests to bypass lower-priority ones and gain access to resources.
- **Advanced arbitration algorithms:** Explore sophisticated algorithms like weighted round-robin or fairness-based approaches for improved resource allocation efficiency and reduced latency.
- **Advanced synchronization techniques:** Employ clock domain crossing principles, synchronization circuits, and metastability-aware design methodologies to address clock synchronization challenges.
- **Performance analysis and optimization:** Analyze the arbiter's performance, identify bottlenecks, and optimize the design for better throughput, latency, and power efficiency.
- **Hardware/software co-design:** Investigate combining hardware and software techniques to optimize the arbitration process, offloading tasks to software or implementing intelligent hardware/software co-design for enhanced resource allocation flexibility and adaptability.

References

1. "Sequential Logic Circuits and the SR Flip-flop," Basic Electronics Tutorials, Aug. 29, 2013. https://www.electronics-tutorials.ws/sequential/seq_1.html
2. "VLSICoding: VHDL Code for Round Robin Arbiter with Fixed Time Slices," VLSICoding, Oct. 31, 2013. https://vlsicoding.blogspot.com/2013/10/design-round-robin-arbiter-with-fixed.html
3. "round robin arbitration | RTLery," rtlery.com. https://rtlery.com/articles/round-robin-arbitration
4. "3.1.5.1. Round-Robin Arbitration," Intel. https://www.intel.com/content/www/us/en/docs/programmable/683364/18-1/round-robin-arbitration.html