Adaptive algorithm for High-Performance FPGA Cores

Shankar Dheeraj Konidena

dheeraj@ittc.ku.edu

KU INFORMATION & TELECOMMUNICATION TECHNOLOGY CENTER
The University of Kansas

- Introduction
- Issue
- Background
- Architecture
- Implementation
- Results
- Conclusion & Future work
- Questions

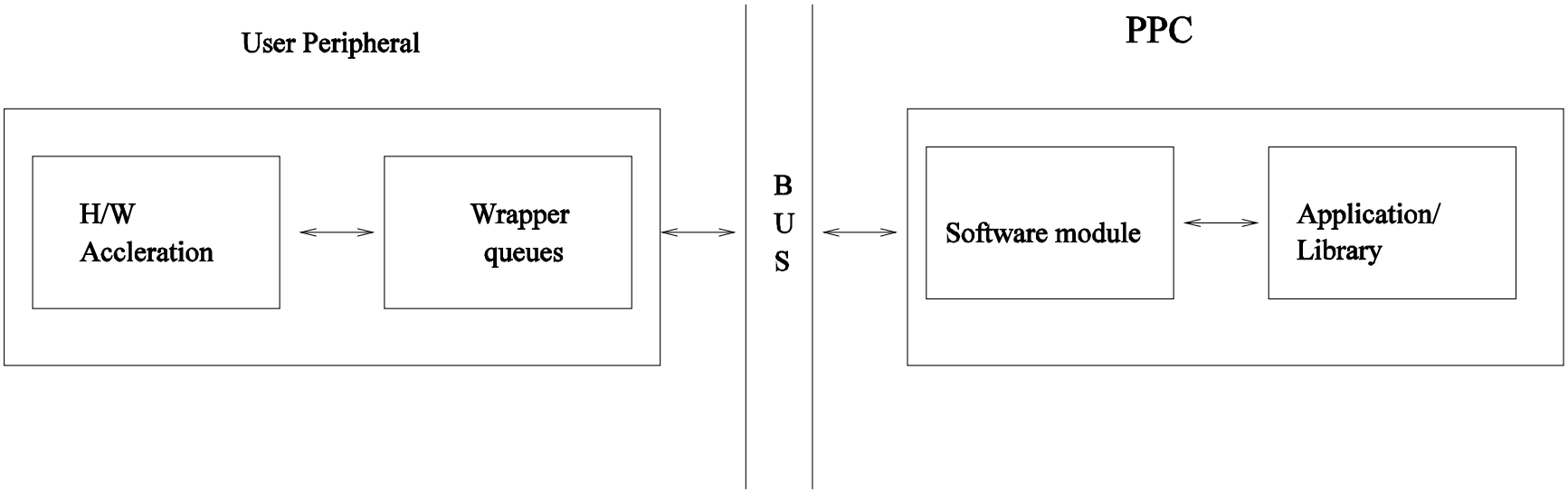
Introduction

- Simulation
- FPGA
  - Input / Output wrapper queues
  - Application core
- Concurrency Scenario

Issue

- FPGA Cores
  - consume data quickly
  - have relatively smaller buffers
  - rates vary over time
- Core Stall can
  - destroy execution rate
  - waste energy due to static power
  - degrade off-chip memory bandwidth

Background


User Peripheral
PPC
H/W
Accleration
Wrapper
queues
B
U
S
Software module
Application/
Library

Background

- Software module
- Application library
- H/W Acceleration
- Stall
  - Conditions for a stall
    - Input / Output wrapper queues

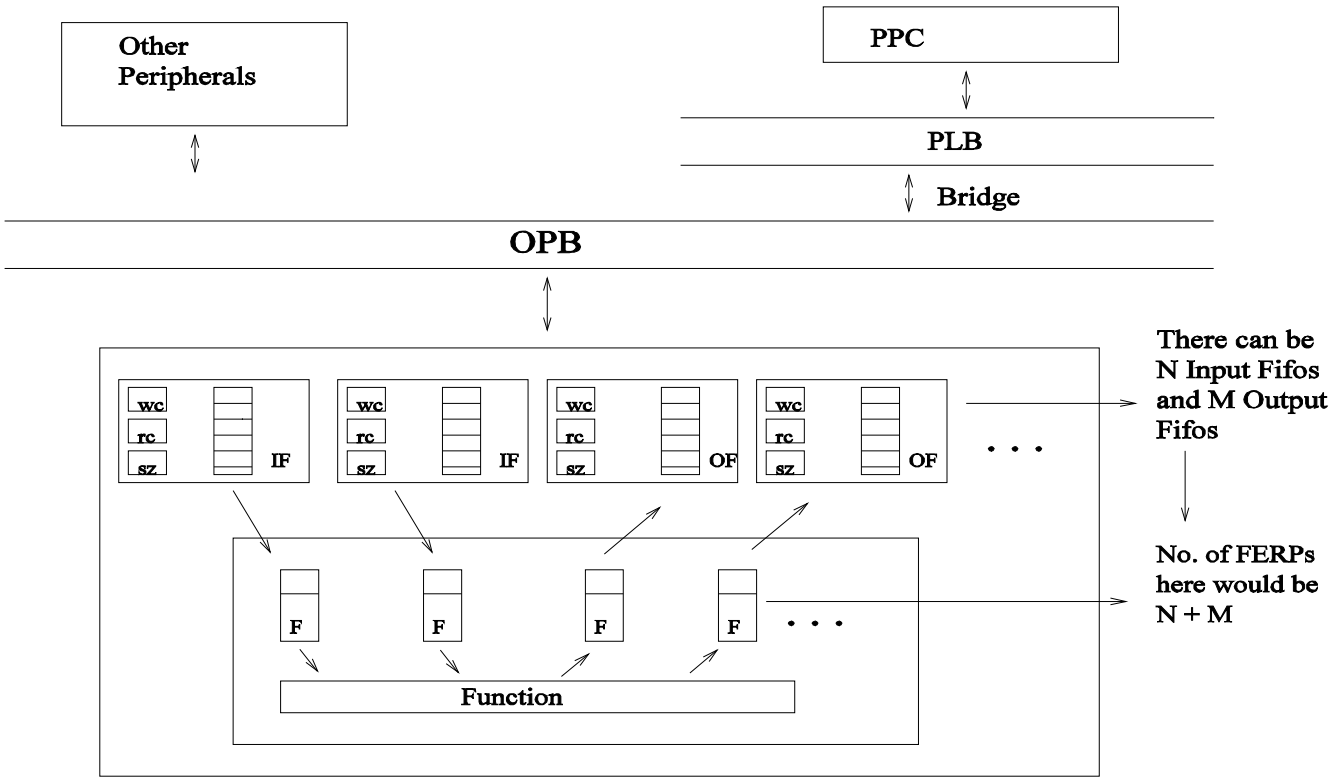
Background

- Discrete event simulation
  - S imulation
    - Ability to model the behaviour of the system as time progresses
    - The process of building simulation models will invariably involve some form of software
    - Key of simulation:
      - simulation executive
        - Responsible for controlling the time advance
        - A central clock is used to keep track of time

Background

- Discrete Event Simulation
  - Mechanisms of Time advance
    - Time slicing: Advances the model forward in time at fixed intervals, e.g. every 5 seconds
    - Next event: the model is advanced to the time of next significant event
      - More discrete
      - In most cases, is more efficient and allows models to be evaluated more quickly
  - Mechanisms for describing logic
    - Event
      - Describes an event as instantaneous change
    - Activity
      - Describe a duration
    - Process
      - Joins a collection of events or entities to describe a life cycle or entity

Architecture



wc - write count
rc - read count
sz - difference of write count and read count
F - FERP

Size of each Fifo(both IF and OF) = 512 entries
Size of each entry in the FIFO = 32 bits

Architecture

- Wrapper queues
  - Size of each queue - 512 entries
  - Size of each entry - 32 bits
  - Each wrapper queue runs at different rates
- Registers
  - Write count register
  - Read count register
  - Difference register
  - Data overflows/ underflows

Implementation

- Simulation environment
  - Multithreading
    - Software Thread
      - Write data from user application to Input wrapper queues
      - Read data from Output wrapper queues to data structures
      - Input wrapper queue lengths are polled before every consumption
      - Output wrapper queue lengths are polled before every production
      - Context switch to hardware thread if not enough space present in the Input queues to write or not enough data present in the Output queues to read

Implementation

- Simulation environment
  - Hardware Thread
    - Hardware Accelerator consumes data from input wrapper queues based on the corresponding Input consumption rates
    - Hardware Accelerator produces data to the Output wrapper queues based on the corresponding Output production rates
    - Finally data is produced to the Output buffers sitting in the Software
    - Context switch

Implementation

- Simulation environment
  - Event Simulation
    - Event
      - Event type
      - Input Consumption
      - Output Production
    - Fifo
      - Input wrapper queue
      - Output wrapper queue
    - Timer
      - Timestamp retrieved from system clock
    - Simulation implemented by a maintaining an event list
    - Event list sorted based on the timestamps of events

Implementation

- Simulation environment
  - The simulation works such that for every consumption / production, the queue lengths are polled.
  - Find a way to accomplish the same result of filling/unfilling the queues without polling queue lengths
  - Adaptive Algorithm
    - Assumption of when exactly the fifo can be empty or full based on the rate at which its consuming or producing data

Implementation

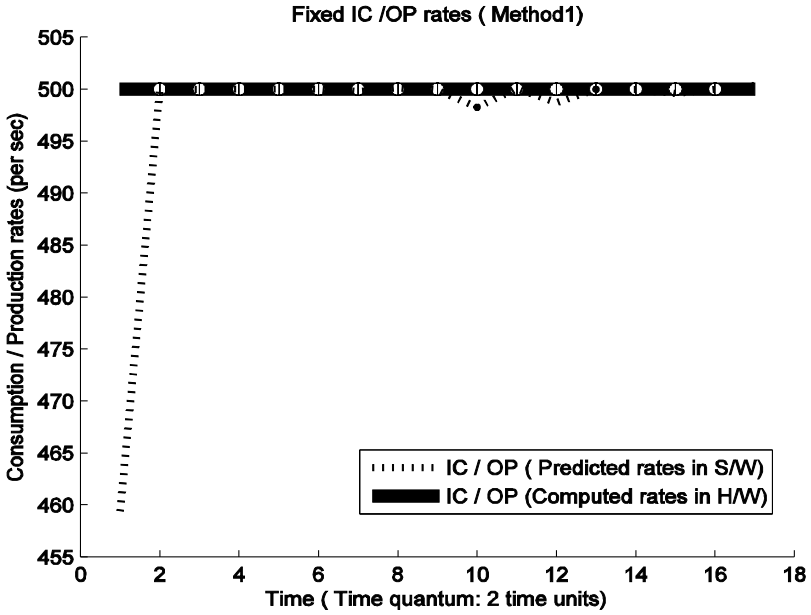
- Simulation environment
  - Adaptive Algorithm
    - Set a Time quantum (T1)
    - For each T1,
      - Note the amount of data consumed / produced after last Time quantum (D)
      - Note the increment/ decrement in fifosizes after last Time quantum (F)
      - Compute the no. of elements consumed / produced based on D, F (C)
      - Consumption / Production rate = C / T1
      - Set new time quantum T2 and repeat

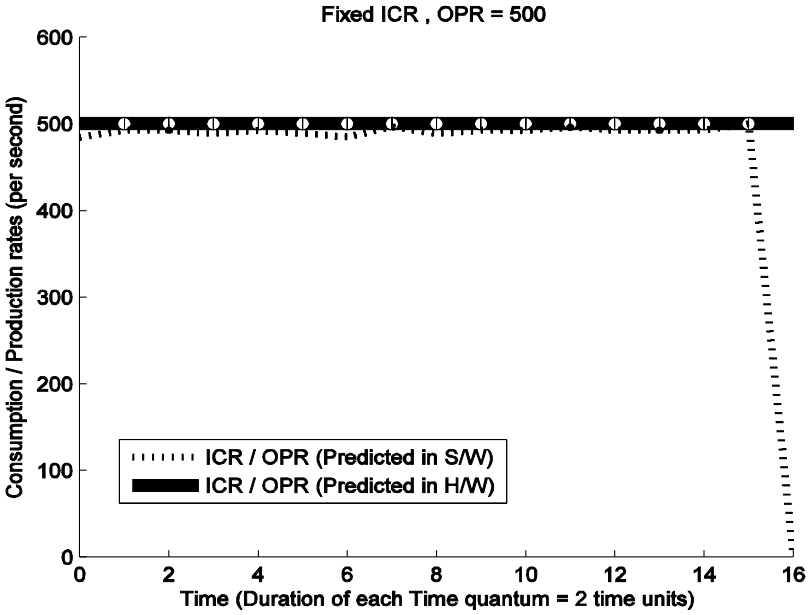
Results

- The simulations:
  - Method1: Polling of queue lengths
  - Method2: Adaptive Algorithm
  - Parameters:
    - FIFOSIZE 128 – 512
    - ICR / OPR
      - Fixed rates – 500
      - Normal , Incremental rates – mean of 500
    - No. of Fifos – 1 Input Fifo, 1 Output Fifo
    - Amount of Data processed – 8 * FIFOSIZE

Results

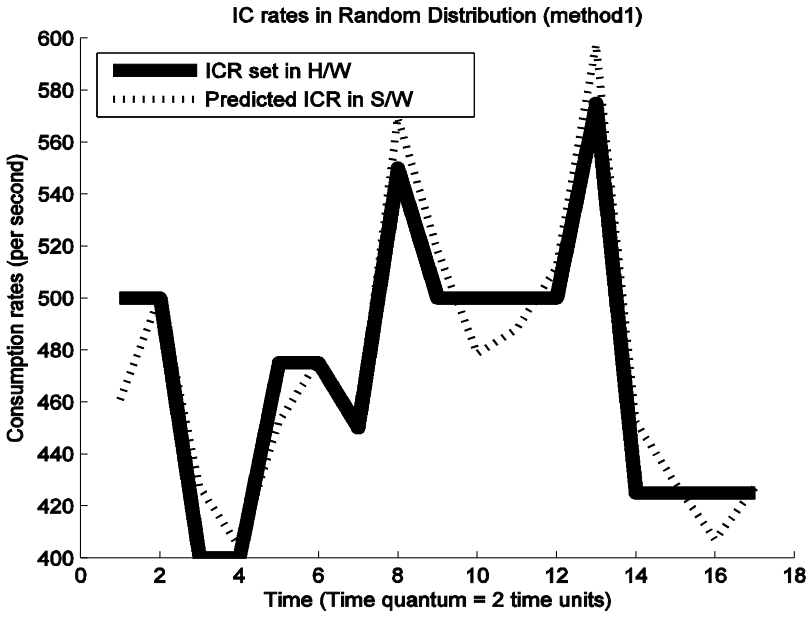
Fixed rates

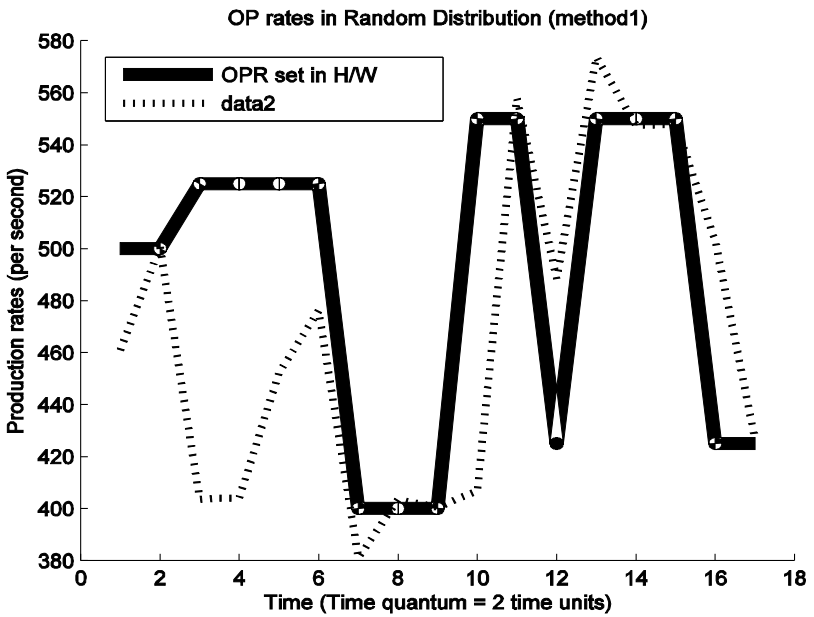




Results

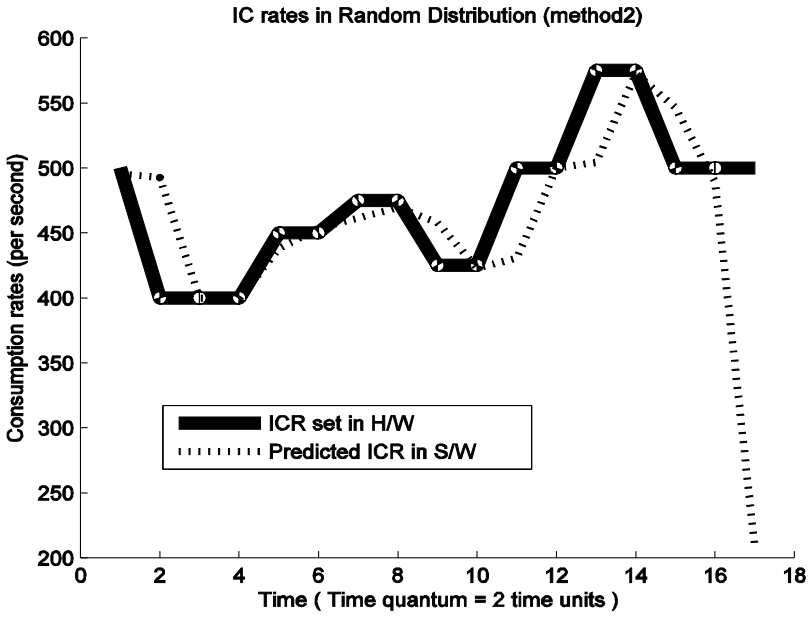
Normal Distribution (Method1)

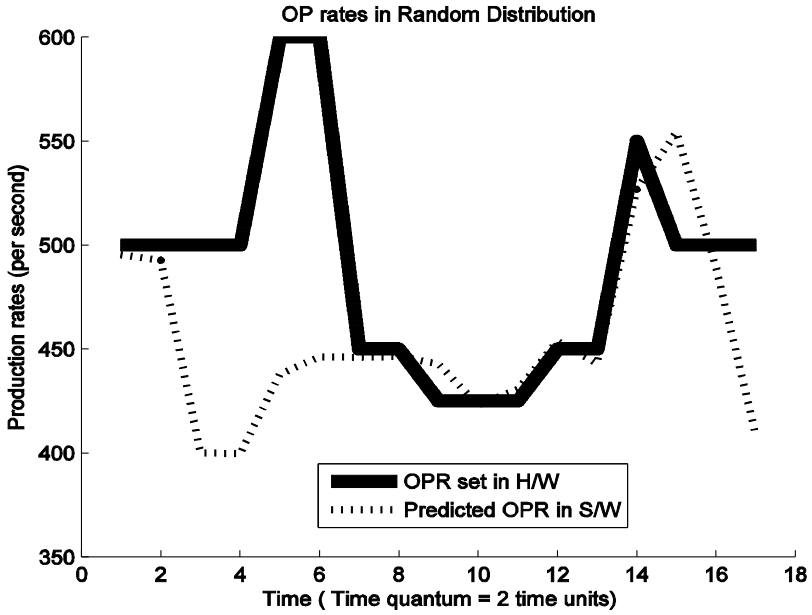




Results

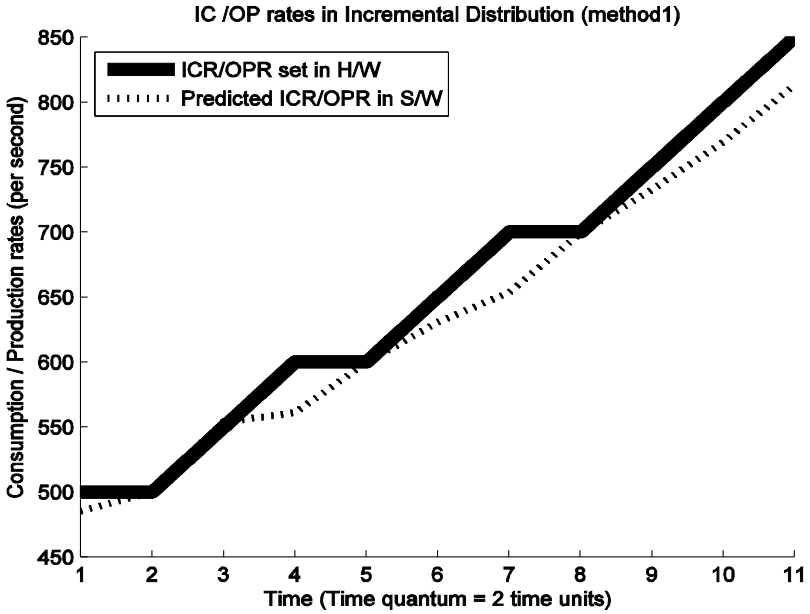
Normal Distribution (Method2)

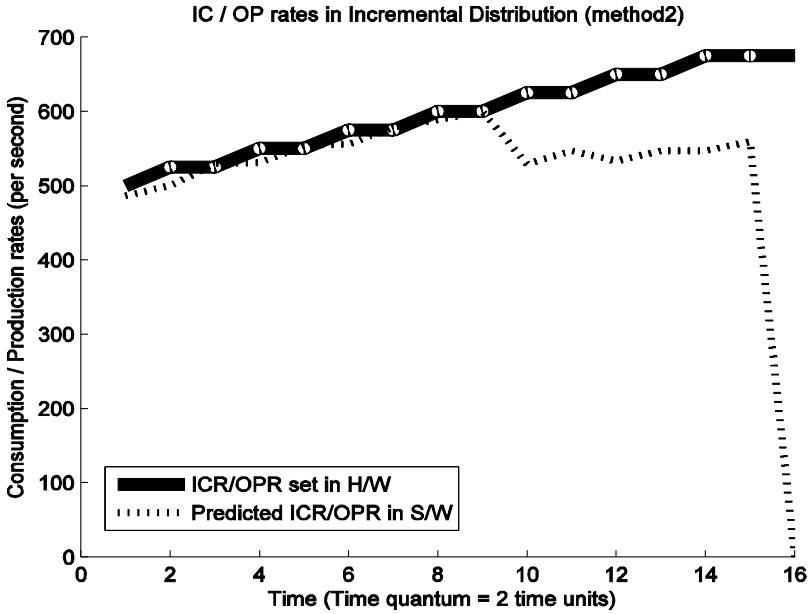




Results

Incremental Distribution





Conclusion & Future work

- Multithreaded environment
  - Hardware, Software threads perform data traversal without a stall
  - Adaptive algorithm implemented
  - Enhancements
    - Needs to be extended to 16 Input / Output wrapper queues
    - The software module now working as the software thread needs to be replaced with a Device driver

Questions