Mixed-Criticality Scheduling upon Varying-Speed Processors

Sanjoy Baruah and Zhishan Guo
Department of Computer Science,
UNC Chapel Hill
Outline

• Motivation
• Model
• Problem Description
• Algorithm
• Simulation and Discussion
• Conclusion and further work
Outline

• Motivation
• Model
• Problem Description
• Algorithm
• Simulation and Discussion
• Conclusion and further work
Mixed-Criticality Systems

- The analysis of **Mixed-Criticality** embedded systems has been identified as one of the core foundational focal areas in the emerging discipline of **Cyber-Physical Systems**.
Mixed-Criticality Systems

• The analysis of **Mixed-Criticality** embedded systems has been identified as one of the core foundational focal areas in the emerging discipline of **Cyber-Physical Systems**.

• Mixed-Criticality arises from...
  
  – WCET Cost
  – Periods

![Diagram showing WCET Cost and Periods](image)
Mixed-Criticality Systems

• The analysis of **Mixed-Criticality** embedded systems has been identified as one of the core foundational focal areas in the emerging discipline of *Cyber Physical Systems*.

• Mixed-Criticality arises from...
  - WCET Cost
  - Periods
  - Processing Speeds
Varying-Speed Processors

- Hardware Design
- Ambient Temperature
- Work Load + Battery Strength

Varying Processor Speed
Varying-Speed Processors

Recover late signals (at circuit level) by delaying the next clock tick.
Varying-Speed Processors

Hardware Design

Ambient Temperature

Temperature Changing
Linux: cpuspeed

Varying Processor Speed
Varying-Speed Processors

- Hardware Design
- Ambient Temperature
- Work Load + Battery Strength

Dynamic Freq. Scaling
- Lightly-Loaded Processors
- Clock Rates (voltage) Reducing

Varying Processor Speed
Varying-Speed Processors

- Hardware Design
- Ambient Temperature
- Work Load + Battery Strength

Applications

Varying Processor Speed

Wireless Network
Outline

• Motivation
• Model
• Problem Description
• Algorithm
• Simulation and Discussion
• Conclusion and further work
Model - Varying-Speed Processors

Processor speed $s(t)$

Time $t$
Model - Varying-Speed Processors

• Computing capacity within interval \([a,b)\):

\[ \int_a^b s(t) \, dt \]
Model - Varying-Speed Processors

• Normal mode vs. Degraded mode

- Degraded mode: Computing capabilities are diminished

\[
\text{Process speed} \geq s_n \quad \text{Process speed} < s_n, \text{ but } \geq s_d
\]
Model - Varying-Speed Processors

- Normal mode vs Degraded mode

- Degraded mode: Computing capabilities are diminished
Model - Varying-Speed Processors

- Normal mode vs Degraded mode

Processor speed $s(t)$

$t$

$S_n$

$S_d$

Normal mode

Degraded mode
Model - Varying-Speed Processors

- Normal mode vs Degraded mode

May switch mode at any time
Model - Varying-Speed Processors

- It is not a priori known when, or whether, such degradation will occur (non-clairvoyant).
- We do however assume that the system is capable of self-monitoring: the processor “immediately” knows if and when such degradation occurs.

\[ S_n \]
\[ S_d \]

0 \hspace{1cm} t

Normal mode
Degraded mode
Outline

• Motivation
• Model
• Problem Description
• Algorithm
• Simulation and Discussion
• Conclusion and further work
The Problem

• Given MC instance

HI-Criticality Jobs

LO-Criticality Jobs

Job set

Independent, Preemptive
One shot job model
Mixed-Criticality systems

• A job $J_i$ - $(a_i, d_i, c_i, \chi_i)$
  – Release time $a_i$
  – Deadline $d_i$
  – Worst Case Execution Time $c_i$
  – Criticality level $\chi_i \in \{\text{LO, HI}\}$.

• An Example
  
  $J_{\text{LO}}=(0,1,2,\text{Lo}), J_{\text{HI}}=(0,2,4,\text{Hi})$
The Problem

• Given MC instance + varying-speed processor

HI-Criticality Jobs

LO-Criticality Jobs

$S_n$  $S_d$  Uni-processor
The Problem

• Given MC instance + varying-speed processor,
• Construct a correct scheduling strategy that
The Problem

• Given MC instance + varying-speed processor,
• Construct a correct scheduling strategy that
Outline

• Motivation
• Model
• Problem Description
• Algorithm
• Simulation and Discussion
• Conclusion and further work
Algorithm - Overview

• Construct a scheduling table $S$ for all jobs (with Linear Programming) prior to run-time
Algorithm - Overview

• Construct a scheduling table $S$ for all jobs (with Linear Programming) prior to run-time
• Use $S$ when processor is in normal mode
Algorithm - Overview

• Construct a scheduling table $S$ for all jobs (with Linear Programming) prior to run-time
• Use $S$ when processor is in normal mode
• If at any instant processor is detected in degraded mode:
  – Table $S$ is no longer used
  – “Discard” all LO-criticality jobs immediately
  – Execute the (remaining) HI-criticality jobs according to EDF
Algorithm - Overview

• Construct a scheduling table $S$ for all jobs (with Linear Programming) prior to run-time
• Use $S$ when processor is in normal mode
• If at any instant processor is detected in degraded mode:
  – Table $S$ is no longer used
  – “Discard” all LO-criticality jobs immediately
  – Execute the (remaining) HI-criticality jobs according to $EDF$
• Whenever processor returns to normal mode, dropped jobs are recovered according to $S$
Algorithm - An Example

<table>
<thead>
<tr>
<th>I</th>
<th>a_i</th>
<th>c_i</th>
<th>d_i</th>
<th>( \chi_i )</th>
</tr>
</thead>
<tbody>
<tr>
<td>J_{LO}</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>LO</td>
</tr>
<tr>
<td>J_{HI}</td>
<td>0</td>
<td>2</td>
<td>4</td>
<td>HI</td>
</tr>
</tbody>
</table>

\[ s_n = 1 \]
\[ s_d = 0.5 \]
Algorithm - An Example

<table>
<thead>
<tr>
<th></th>
<th>I</th>
<th>a_i</th>
<th>c_i</th>
<th>d_i</th>
<th>x_i</th>
</tr>
</thead>
<tbody>
<tr>
<td>J_LO</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>LO</td>
<td></td>
</tr>
<tr>
<td>J_HI</td>
<td>0</td>
<td>2</td>
<td>4</td>
<td>HI</td>
<td></td>
</tr>
</tbody>
</table>

Intervals

<table>
<thead>
<tr>
<th></th>
<th>[0,2)</th>
<th>[2,4)</th>
</tr>
</thead>
<tbody>
<tr>
<td>J_LO</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>J_HI</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

\( s_n = 1 \)
\( s_d = 0.5 \)

\( s(t) \)

0   2   4

THE UNIVERSITY of NORTH CAROLINA at CHAPEL HILL
Algorithm - An Example

<table>
<thead>
<tr>
<th></th>
<th>$l$</th>
<th>$a_i$</th>
<th>$c_i$</th>
<th>$d_i$</th>
<th>$\chi_i$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$J_{LO}$</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>LO</td>
<td></td>
</tr>
<tr>
<td>$J_{HI}$</td>
<td>0</td>
<td>2</td>
<td>4</td>
<td>HI</td>
<td></td>
</tr>
</tbody>
</table>

$s_n = 1$
$s_d = 0.5$

- **Intervals**
  
  - **$[0,2)$**
    - $J_{LO}$
    - $J_{HI}$

  - **$[2,4)$**
    - $J_{LO}$
    - $J_{HI}$
Algorithm - An Example

<table>
<thead>
<tr>
<th>( I )</th>
<th>( a_i )</th>
<th>( c_i )</th>
<th>( d_i )</th>
<th>( \chi_i )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( J_{LO} )</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>LO</td>
</tr>
<tr>
<td>( J_{HI} )</td>
<td>0</td>
<td>2</td>
<td>4</td>
<td>HI</td>
</tr>
</tbody>
</table>

\[ s_n = 1 \]
\[ s_d = 0.5 \]

Intervals:
- \([0,2)\]
- \([2,4)\]

| \( J_{LO} \) | 1 | 0 |
| \( J_{HI} \) | 1 | 1 |

Graphical representation:
- \( s(t) \)
- \( t \)
- \( J_{HI} \)
- \( J_{LO} \)
Algorithm - An Example

<table>
<thead>
<tr>
<th></th>
<th>$a_i$</th>
<th>$c_i$</th>
<th>$d_i$</th>
<th>$\chi_i$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$J_{LO}$</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>LO</td>
</tr>
<tr>
<td>$J_{HI}$</td>
<td>0</td>
<td>2</td>
<td>4</td>
<td>HI</td>
</tr>
</tbody>
</table>

$s_n = 1$
$s_d = 0.5$

<table>
<thead>
<tr>
<th>Intervals</th>
<th>[0,2)</th>
<th>[2,4)</th>
</tr>
</thead>
<tbody>
<tr>
<td>$J_{LO}$</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>$J_{HI}$</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

$J_{HI}$

$J_{LO}$

$J_{HI}$
Algorithm - An Example

<table>
<thead>
<tr>
<th>I</th>
<th>$a_i$</th>
<th>$c_i$</th>
<th>$d_i$</th>
<th>$\chi_i$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$J_{LO}$</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>LO</td>
</tr>
<tr>
<td>$J_{HI}$</td>
<td>0</td>
<td>2</td>
<td>4</td>
<td>HI</td>
</tr>
</tbody>
</table>

$s_n = 1$
$s_d = 0.5$

Intervals

<table>
<thead>
<tr>
<th>Intervals</th>
<th>$[0,2)$</th>
<th>$[2,4)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$J_{LO}$</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>$J_{HI}$</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

$J_{HI}$

$s(t)$

$0 \quad 2 \quad 4$
Algorithm - An Example

<table>
<thead>
<tr>
<th>I</th>
<th>$a_i$</th>
<th>$c_i$</th>
<th>$d_i$</th>
<th>$\chi_i$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$J_{LO}$</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>LO</td>
</tr>
<tr>
<td>$J_{HI}$</td>
<td>0</td>
<td>2</td>
<td>4</td>
<td>HI</td>
</tr>
</tbody>
</table>

$s_n = 1$
$s_d = 0.5$
Algorithm - An Example

<table>
<thead>
<tr>
<th>I</th>
<th>a_i</th>
<th>c_i</th>
<th>d_i</th>
<th>( \chi_i )</th>
</tr>
</thead>
<tbody>
<tr>
<td>J_{LO}</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>LO</td>
</tr>
<tr>
<td>J_{HI}</td>
<td>0</td>
<td>2</td>
<td>4</td>
<td>HI</td>
</tr>
</tbody>
</table>

\( s_n = 1 \)
\( s_d = 0.5 \)

<table>
<thead>
<tr>
<th>Intervals</th>
<th>[0,2)</th>
<th>[2,4)</th>
</tr>
</thead>
<tbody>
<tr>
<td>J_{LO}</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>J_{HI}</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
Algorithm - An Example

<table>
<thead>
<tr>
<th>I</th>
<th>$a_i$</th>
<th>$c_i$</th>
<th>$d_i$</th>
<th>$\chi_i$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$J_{LO}$</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>LO</td>
</tr>
<tr>
<td>$J_{HI}$</td>
<td>0</td>
<td>2</td>
<td>4</td>
<td>HI</td>
</tr>
</tbody>
</table>

Intervals

- $[0,2)$
- $[2,4)$

<table>
<thead>
<tr>
<th>Intervals</th>
<th>$J_{LO}$</th>
<th>$J_{HI}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$[0,2)$</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>$[2,4)$</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

$s_n = 1$
$s_d = 0.5$
## Algorithm - An Example

### Table

<table>
<thead>
<tr>
<th>I</th>
<th>$a_i$</th>
<th>$c_i$</th>
<th>$d_i$</th>
<th>$\chi_i$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$J_{LO}$</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>LO</td>
</tr>
<tr>
<td>$J_{HI}$</td>
<td>0</td>
<td>2</td>
<td>4</td>
<td>HI</td>
</tr>
</tbody>
</table>

### Intervals

- $[0,2)$
- $[2,4)$

<table>
<thead>
<tr>
<th>$S$</th>
<th>Intervals</th>
<th>[0,2)</th>
<th>[2,4)</th>
</tr>
</thead>
<tbody>
<tr>
<td>$J_{LO}$</td>
<td>$1$</td>
<td>$0$</td>
<td></td>
</tr>
<tr>
<td>$J_{HI}$</td>
<td>$1$</td>
<td>$1$</td>
<td></td>
</tr>
</tbody>
</table>

$S_n = 1$

$S_d = 0.5$
Algorithm - An Example

<table>
<thead>
<tr>
<th>$J$</th>
<th>$a_i$</th>
<th>$c_i$</th>
<th>$d_i$</th>
<th>$\chi_i$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$J_{LO}$</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>LO</td>
</tr>
<tr>
<td>$J_{HI}$</td>
<td>0</td>
<td>2</td>
<td>4</td>
<td>HI</td>
</tr>
</tbody>
</table>

$s_n = 1$
$s_d = 0.5$

Intervals

<table>
<thead>
<tr>
<th>Intervals</th>
<th>[0,2)</th>
<th>[2,4)</th>
</tr>
</thead>
<tbody>
<tr>
<td>$J_{LO}$</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>$J_{HI}$</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Normal mode => recovery

$J_{LO}$

$J_{HI}$

s(t)

0 2 4
Algorithm - An Example

<table>
<thead>
<tr>
<th>l</th>
<th>$a_i$</th>
<th>$c_i$</th>
<th>$d_i$</th>
<th>$\chi_i$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$J_{LO}$</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>LO</td>
</tr>
<tr>
<td>$J_{HI}$</td>
<td>0</td>
<td>2</td>
<td>4</td>
<td>HI</td>
</tr>
</tbody>
</table>

$s_n = 1$
$s_d = 0.5$

Intervals

<table>
<thead>
<tr>
<th>Intervals</th>
<th>[0,2)</th>
<th>[2,4)</th>
</tr>
</thead>
<tbody>
<tr>
<td>$J_{LO}$</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>$J_{HI}$</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

$J_{LO}$ may not meet its deadline
Algorithm - An Example

<table>
<thead>
<tr>
<th>l</th>
<th>a_i</th>
<th>c_i</th>
<th>d_i</th>
<th>x_i</th>
</tr>
</thead>
<tbody>
<tr>
<td>J_{LO}</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>LO</td>
</tr>
<tr>
<td>J_{HI}</td>
<td>0</td>
<td>2</td>
<td>4</td>
<td>HI</td>
</tr>
</tbody>
</table>

\[ s(t) \]

<table>
<thead>
<tr>
<th>Intervals</th>
<th>[0,2)</th>
<th>[2,4)</th>
</tr>
</thead>
<tbody>
<tr>
<td>J_{LO}</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>J_{HI}</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

\[ s_n = 1 \]
\[ s_d = 0.5 \]
Algorithm - Linear Programming

• To construct the table $S := x_{i,j} \geq 0$
  – amount of execution assign to job $J_i$ in interval $I_j$
Algorithm - Linear Programming

- Constraints to construct $S := x_{i,j} \geq 0$
  - Each job receives adequate execution under normal circumstances

  - For each $i$, $1 \leq i \leq n$,

    $$\sum_{j \mid t_j \geq a_i \land d_i \geq t_{j+1}} x_{i,j} \geq c_i$$
Algorithm - Linear Programming

• Constraints to construct $S := x_{i,j} \geq 0$
  – Each job receives adequate execution (normal)
  – The capacity of each interval is respected

  - For each $j$, $1 \leq j \leq k$,
    $\left( \sum_{i=1}^{n} x_{i,j} \right) \leq t_{j+1} - t_j$
Algorithm - Linear Programming

• Constraints to construct $S := x_{i,j} \geq 0$
  – Each job receives adequate execution (normal)
  – The capacity of each interval is respected
  – Degradation at any time should not cause a HI-criticality job miss its deadline

\[
\sum_{i : (\chi_i = \text{HI}) \land (d_i \leq t_m)} \left( \sum_{j=\ell}^{m-1} x_{i,j} \right) \leq s_d(t_m - t_\ell)
\]
Algorithm - Linear Programming

• Variables $S(I) := x_{i,j} \geq 0$

• Constraints
  
  For each $i$, $1 \leq i \leq n$,
  
  $$\left( \sum_{j \mid t_j \geq a_i \land d_i \geq t_{j+1}} x_{i,j} \right) \geq c_i \quad (1)$$

  For each $j$, $1 \leq j \leq k$,
  
  $$\left( \sum_{i=1}^{n} x_{i,j} \right) \leq t_{j+1} - t_j \quad (2)$$

  For each $\ell$, $1 \leq \ell \leq k$, for each $m$, $\ell < m \leq (k + 1)$
  
  $$\left( \sum_{i \mid (x_i = \text{HI}) \land (d_i \leq t_m)} \left( \sum_{j=\ell}^{m-1} x_{i,j} \right) \right) \leq s_d^{(t_m - t_\ell)} \quad (3)$$

\[ n \text{ - Number of jobs} \]
\[ \text{Variables: } O(n^2), \]
\[ \text{Constraints: } O(n^2) \]
\[ \text{Polynomial-time solvable} \]
Outline

• Motivation
• Model
• Problem Description
• Algorithm
• Simulation and Discussion
• Conclusion and further work
Algorithm - Optimization Version

MC job set $J$
- Normal speed $s_n$
- Degraded speed $s_d$

Y/N

Correctness?
Algorithm - Optimization Version

MC job set $J$
Normal speed $s_n$
Degraded speed $s_d$

Y/N

Correctness?

Minimize

$s_d$
Correctness

$s_n = 1$
Algorithm - Optimization Version

\[ \text{minimize } S_d \]

S.t.

- For each \( i, 1 \leq i \leq n \),
  \[ \left( \sum_{j|t_j \geq a_i \land d_i \geq t_{j+1}} x_{i,j} \right) \geq c_i \] \hspace{1cm} (1)

- For each \( j, 1 \leq j \leq k \),
  \[ \left( \sum_{i=1}^{n} x_{i,j} \right) \leq t_{j+1} - t_j \] \hspace{1cm} (2)

- For each \( \ell, 1 \leq \ell \leq k \), for each \( m, \ell < m \leq (k + 1) \),
  \[ \left( \sum_{i:(x_i=HI) \land (d_i \leq t_m)} \left( \sum_{j=\ell}^{m-1} x_{i,j} \right) \right) \leq S_d (t_m - t_\ell) \] \hspace{1cm} (3)

- For each \( i,j \), \( x_{i,j} \geq 0 \)
Simulation

• Random Independent Job Set Generator
  – $n$: Number of jobs,
Simulation

• Random Independent Job Set Generator
  – $n$: Number of jobs,
  – $f_{HI}$: Expected fraction of HI-criticality jobs,
Simulation

- Random Independent Job Set Generator
  - $n$: Number of jobs,
  - $f_{HI}$: Expected fraction of HI-criticality jobs,
  - $u_{all}$: Overall computational load
Simulation

• Random Independent Job Set Generator
  – $n$: Number of jobs,
  – $f_{HI}$: Expected fraction of HI-criticality jobs,
  – $u_{all}$: Overall computational load
  – $\zeta$: expected number of jobs with scheduling windows that overlap (cover) each time instant.

$$\zeta = 1 \quad a_1 \downarrow d_1 = a_2 \quad d_2 = a_3 \quad d_{n-1} = a_n \quad d_n \downarrow t$$

$$\zeta = n \quad a_i \downarrow d_i \downarrow t$$
Simulation

**load** is equal to the speed of the smallest processor upon which such a collection can be scheduled using preemptive EDF.

\[
\text{load}_{HI} = \max \left\{ \sum_{i \in [t_k, t_l)} \frac{c_i}{t_1 - t_k} \right\}
\]
Simulation

\[ S_d \]

\( \text{load}_{HI} \)
Simulation

A generated example

\[ s_d = 1 \]

\[ \text{load}_{HI} = 0.5 \]

\[ \text{Load} = 1, \quad \text{Load}_{HI} = 0.5 \]

\[ S(I) = [0,2) \quad [2,4) \]

<table>
<thead>
<tr>
<th>I</th>
<th>( a_i )</th>
<th>( c_i )</th>
<th>( d_i )</th>
<th>( \chi_i )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( J_{LO} )</td>
<td>0</td>
<td>2</td>
<td>2</td>
<td>LO</td>
</tr>
<tr>
<td>( J_{HI} )</td>
<td>0</td>
<td>2</td>
<td>4</td>
<td>HI</td>
</tr>
</tbody>
</table>

\[ s(t) \]

\[ 0 \quad 2 \quad 4 \]
Theoretical bound for clairvoyant algorithm: \( S_d \) is equal to the speed of the smallest processor upon which such a collection can be scheduled using preemptive EDF.
Outline

• Motivation
• Model
• Problem Description
• Algorithm
• Simulation and Discussion
• Conclusion and further work
Conclusion

- Model
  - Platforms with *varying-speed* performance during run-time
Conclusion

- **Model**
  - Platforms with *varying-speed* performance during run-time

- **Correct Algorithm**
  - MC job set on *self-monitoring* varying-speed processor
  - Based on Linear Programming
  - Job set generator + Simulation
Further Work

• Efficiency improvements
  – Linear Programming
  – Polynomial -> $O(n^2)$ or $O(n \log n)$
Further Work

• Efficiency improvements
• Theoretical analysis for worst case
  – Upper bound for speedup factor
Further Work

- Efficiency improvements
- Theoretical analysis for worst case
- Multiple levels of criticality
  - More than two thresholds for processor speeds
Further Work

- Efficiency improvements
- Theoretical analysis for worst case
- Multiple levels of criticality
- Combination with previous models
  - Schedule MC instance upon varying-speed processors

<table>
<thead>
<tr>
<th>l</th>
<th>a_i</th>
<th>c_i</th>
<th>d_i</th>
<th>χ_i</th>
</tr>
</thead>
<tbody>
<tr>
<td>J_{LO}</td>
<td>0</td>
<td>[1,1]</td>
<td>2</td>
<td>LO</td>
</tr>
<tr>
<td>J_{HI}</td>
<td>0</td>
<td>[1,2]</td>
<td>4</td>
<td>HI</td>
</tr>
</tbody>
</table>

s_n = 1
s_d = 0.5
Thank you!

Zhishan Guo
zsguo@cs.unc.edu
Thank you!!!

Zhishan Guo
zsguo@cs.unc.edu