An Introduction to the DES (Discrete Event System) Analyzer: A Performance Analysis and Timing Verification Tool for Concurrent Digital Systems

Peggy B. McGee Steven M. Nowick

\{pmcgee,nowick\}@cs.columbia.edu

Department of Computer Science Columbia University

This work was partially supported by NSF ITR Award No. NSF-CCR-0086036, an Initiatives in Science and Engineering (ISE) grant from the Office of the Executive Vice President for Research of Columbia University, and a subcontract to Boeing under the DARPA CLASS program
Developers and documentation

► Developers (2005 - present)
  • Peggy B. McGee: design and implementation
  • Steven M. Nowick: project management

► Documentation
  • Peggy B. McGee, Steven M. Nowick and E.G. Coffman Jr.,
    “Efficient Performance Analysis of Asynchronous Systems Based on Periodicity,"
    in Proceedings of the 3rd IEEE/ACM/IFIP International Conference on

  • Peggy B. McGee and Steven M. Nowick,
    “An Efficient Algorithm for Time Separation of Events in Concurrent Systems,"
    in Proceedings of the 2007 IEEE/ACM International Conference on
    Computer-Aided Design (ICCAD ’07), Nov. 2007.
Download site

• Accessible on the web from:
  http://www1.cs.columbia.edu/~nowick/asynctools

• Package includes:
  • Tool binaries
    ▶ Currently, Linux version only
  • Introduction and tutorial slides (this document)
  • Benchmark examples
  • Other documentation
    ▶ Tool setup instructions (README)
    ▶ Related conference publications
    ▶ Related conference presentation slides
Outline

► The DES Analyzer:
  • Introduction
  • Tool flow overview
► Background on modeling
► Overview of analysis methods
► Tool features
► Tutorial: Design examples and hands-on tutorial
  • Using des-tse: Time separation of events (TSE) analysis
    ► Example 1a: FIFO ring
    ► Example 1b: Micropipeline
  • Using des-perf: Performance analysis
    ► Example 2: Micropipeline
► Conclusions
The DES Analyzer:  
*Introduction & tool flow overview*
The DES Analyzer: Goals and Applications

▶ Overall goal:
  • A CAD package for analyzing the timing behavior of digital concurrent systems
    ▶ Asynchronous systems
    ▶ Mixed-timing systems, e.g. GALS

▶ Applications
  • Performance analysis
    ▶ Finds average-case system latency and throughput
    ▶ Finds worst and best-case system latency and throughput
  • Timing verification
    ▶ Identifies violations of system-level timing constraints
  • Optimization
    ▶ Finds system performance bottlenecks
    ▶ Identifies impossible ordering of events
      • Increases don’t-care space for synthesis
The DES Analyzer: Scope

- **Scope:**
  - Assumes repetitive systems
    - System interacts with environment continuously
  - Assumes systems modeled with concurrent graphs
    - Currently supports *marked graphs*, a sub-class of Petri nets
  - Handles two types of delay models
    - Bounded delays = lower and upper bounds (*for des-tse*)
      - special case: Fixed delays = single delay number
    - Exponential distributions (*for des-perf*)
  - Currently only handles choice-free systems
    - Support for systems with choice planned in future releases
The DES Analyzer: Tool package

- Two analysis tools under the package:
  1. des-tse
     = Time Separation of Events analysis
     - For bounded-delay systems = min/max delay bounds
     - Special case: fixed-delay systems = single delay number
     - Applications:
       - Timing verification
       - Best- and worst-case performance analysis
       - Average-case performance analysis
       - for fixed-delay systems only
  2. des-perf
     = Performance analysis
     - For stochastic-delay systems (exponential distributions)
     - Applications:
       - Average-case performance analysis
The DES Analyzer: Tool flow overview

Input Specification (Timed Marked Graph)

DES Analyzer

Graphical Display of Input Specification

Max or Min TSE (Also average TSE for fixed-delay systems)

Average-case performance metrics

des-tse  des-perf
Background on modeling
Background on modeling: Marked graphs

Background on modeling: Marked graphs


**node:** an event in the system
Background on modeling: Marked graphs

edge: captures a pre-condition to an event

node: an event in the system

**Background on modeling: Marked graphs**

- **edge**: captures a pre-condition to an event
- **token**: pre-condition is satisfied
- **node**: an event in the system

Background on modeling: Marked graphs

**edge:** captures a pre-condition to an event

**token:** pre-condition is satisfied

**enabling of node:** a token on each input edge

**node:** an event in the system

Background on modeling: Marked graphs

- **edge**: captures a pre-condition to an event

- **token**: pre-condition is satisfied

- **firing of node**: occurrence of an event
  - a token deposited onto each output edge

- **node**: an event in the system

Background on modeling: **Timed marked graphs**

*Timed marked graphs* = An extension of marked graphs to include timing information

Each *edge* or *node* in the marked graph assigned a *delay*

- Types of delay models:
  - Probabilistic distribution, e.g. exponential distribution
  - Bounded delay = lower and upper bounds
    - Special case: fixed delay = single delay number
Background on modeling: *Timed marked graphs*

For the DES Analyzer:

**For des-tse:**

- Bounded delays on edges
  - a→b: [1,2]
  - b→c: [1,3]
  - c→d: [1,2]
  - d→a: [1,2]

**For des-perf**

- Exponentially-distributed delays on nodes
  - λ = 5
  - λ = 2
  - λ = 2
  - λ = 5

(λ = Mean of delay distribution)
Overview of analysis methods
**Key concept:** Capture *exact timing behavior* of system for timing verification

**Example:** Find maximum TSE between two consecutivefirings of nodes $b$ and $d$
des-tse: TSE analysis overview

- Evaluates entire time evolution of system analytically
  - System operates in two phases: "ramp-up" and "steady state"
  - Tool considers timing behavior in both phases

- For fixed-delay systems
  - Critical cycles drive asymptotic timing behavior
  - Critical paths = longest paths from critical cycle to each node
    ▶ Determine relative firing time of system events
    ▶ Find TSE from relative firing time of events

- For bounded-delay systems
  - Re-cast as two fixed-delay problems
  - Solve individually and combine results

For details, see accompanying ICCAD’07
- Conference publication
- Presentation slides
des-perf: Performance analysis overview

Key concept: Derive asymptotic timing behavior of system using Markovian analysis

\[ \lambda = \text{Mean delay at node (assumes exponential distribution)} \]

\[
\begin{align*}
\lambda &= 5 \\
\lambda &= 2 \\
\lambda &= 2 \\
\lambda &= 5 \\
a &\rightarrow b \\
b &\rightarrow c \\
c &\rightarrow d \\
d &\rightarrow a
\end{align*}
\]

Given timed marked graph:

- Gives asymptotic state distribution
- Can be further processed to give performance metrics
- \textit{Example:} average delay(d,b) = 4.8 time units
**des-perf: Performance analysis overview**

- Evaluates asymptotic timing behavior of system analytically
  - Gives average performance metrics of system at steady-state

- System state transition dynamics captured in a *Markov chain*
  - Markov transition probabilities derived from delay distributions

- Efficient method based on *periodic properties* of system for:
  - Constructing the Markov chain
  - Solving the Markov chain

*For details, see accompanying CODES’05*
- *Conference publication*
- *Presentation slides*
Tool features
Tool features: Command line input

- Commands to run the tools:
  - des-tse [input_filename] [options]
  - des-perf [input_filename] [options]

  Input file format and tool options *same* for des-tse and des-perf

- Input file =
  - Text description of timed marked graph

- Outputs
  - Analysis results
    - Printed onto the standard output
    - Can be piped to a text file for further analysis
  - (Optional) graphical display of input specification
**Tool features: Tool options**

- **"-o output_filename"**
  - Optional feature: displaying input specification
  - Given input specification, generates a graphical display
    - Graphical display described in text format
    - Viewable in a third-party tool: dotty
      - Viewer can be downloaded from the AT&T website: [http://www.research.att.com](http://www.research.att.com)

- **"-no_processing"**
  - Overrides tool default by performing no analysis
    - Useful when used together with the "-o" option
      - For generating graphical display only

- **"-help"**
  - Prints "help" information of the commands
Tool features: Input format

- Format of input specification = text file

- Each line in input text file prefixed with an identifier:
  - 
    - `#`
      - The rest of the line is ignored by tool front-end
      - Used for comments
  - `.node_list`
    - Declares list of all nodes in the marked graph
      *example:*. `.node_list a b c d`
    - Must be the first line in the input files
      - Excluding comments
Tool features: Input format (cont’d)

- Each line in input text file prefixed with an identifier (cont’d)
  - `.edge`:
    - Specifies an edge
    - For des-perf: followed by input and output nodes of edge
      Example: `.edge a b`
    - For des-tse: followed by input and output nodes of edge
      - Plus three additional arguments:
        - Lower delay bound
        - Upper delay bound
        - 1 (if there is a token on the edge), or 0 (otherwise)
      Example: `.edge a b 3.5 5.2 1`
Tool features: Input format (cont’d)

- Each line in input text file prefixed with an identifier (cont’d)
  - `.init`
    - Used in des-tse only
    - Specifies the firing time of enabled nodes at initialization
      *Example:* `.init a 0`
  - `.check`
    - Used in des-tse only
    - Specifies two nodes to check TSE for
      *Example:* `.check a b`
    - Alternatively, specifies all nodes
      *Example:* `.check all`
  - `.node`
    - Used in des-perf only
    - Specifies the mean of the delay distribution of a node
      *Example:* `.node a 3.5`
Tutorials

0. Getting started

1. TSE analysis with des-tse
   Example 1a: FIFO ring
   Example 1b: Micropipeline

2. Performance analysis with des-perf
   Example 2: Micropipeline
Tutorial 0: Getting started
Tutorial 0: Getting started

Step 1: Making sure the tool is set up

- Make sure the tool and path for the DES Analyzer are set up:
  - Follow the instructions from the README file

- Test the set-up by running the tool with the “-help” option:
  - `des-tse -help`
  - `des-perf -help`

You should see the following output display:

```
Usage: des-tse [input_file] [-o output_file] [-no_processing]

input_file Filename of input marked graph specification.

-o output_file Graphical display option. Converts input specification to ".dot" format for display with the dotty viewer and writes to output filename.

-no_processing Option to perform no analysis. When used with the "-o" option, prints graphical display only.
```
Tutorial 0: Getting started

Step 2: Setting up the dotty viewer (Optional)

▸ Check if “dotty” is already installed in your environment:
  > which dotty

▸ If the tool is not found in your path, download the tool from:
  http://www.research.att.com

▸ Follow the instruction from the tool website to setup the tool.
Tutorial 0: Getting started

Step 3: Copying tutorial files

- Make a new directory for running the tutorials:
  For example:
  > mkdir DES

- Go to it:
  > cd DES

- Create a subdirectory for each of the two tutorials:
  > mkdir tutorial1
  > mkdir tutorial2

- Copy the example input files to the tutorial directories:
  > cp $DES_HOME/examples/des-tse/micropipeline.txt tutorial1/.
  > cp $DES_HOME/examples/des-tse/fifo_ring_run1.txt tutorial1/.
  > cp $DES_HOME/examples/des-tse/fifo_ring_run2.txt tutorial1/.
  > cp $DES_HOME/examples/des-perf/micropipeline.txt tutorial2/.

$DES_HOME = location of the downloaded DES Analyzer CAD Package
Tutorial 1: TSE analysis with des-tse

Example 1a: FIFO ring
Tutorial 1: TSE analysis with des-tse
- Example 1a: FIFO ring

▶ In this tutorial we shall learn how to:

- **Step 1**: Specify a marked graph input for the des-tse tool
- **Step 2**: Display the input specification graphically
- **Step 3**: Run TSE analysis
- **Step 4**: Specify initial conditions of the system
  - and learn how initial conditions affect TSE results
- **Step 5**: Perform different TSE queries on the system
FIFO ring: marked graph model [McGee et al., ICCAD’07]

This example has *fixed delay* on all edges
(e.g. \([1,1]\) = fixed-delay of 1)
Tutorial 1: TSE analysis with des-tse
- Example 1a: FIFO ring

Step 1: Specifying the marked graph input

► Go to the ‘tutorial1’ directory created in Step 0

► Take a look at the file fifo_ring_run1.txt

```plaintext
# list of nodes in graph
.node_list a b c d

# edge specification: .edge <input node> <output node> <min delay> <max delay> <has token?>
.edge a b 1 1 1
.edge b a 3 3 0
.edge b c 1 1 0
.edge c b 3 3 1
.edge c d 1 1 1
.edge d c 3 3 0
.edge d a 1 1 0
.edge a d 3 3 1

# initial firing time of enabled nodes
.init b 0
.init d 0

# TSE pairs to check
.check b d
```
Step 2: Displaying the input specification

- Generate a graphical output:
  - `des-tse fifo_ring_run1.txt -o fifo_ring.dot -no_processing`

- Display it:
  - `dotty fifo_ring.dot`
  
  A window should pop up to display the following:
Step 3: Running TSE analysis

- Run the tool:
  ```
  des-tse fifo_ring_run1.txt
  ```
- Look at the output:

<table>
<thead>
<tr>
<th>Event pair</th>
<th>Max TSE</th>
<th>Min TSE</th>
</tr>
</thead>
<tbody>
<tr>
<td>(b, d)</td>
<td>6.0</td>
<td>0.0</td>
</tr>
</tbody>
</table>

The result table shows the maximum and minimum TSE between all consecutive firings of events \( b \) and \( d \). From initialization to steady-state.
Tutorial 1: TSE analysis with des-tse
- Example 1a: FIFO ring

Step 4: Specifying different initial conditions

▶ Take a look at both files:
  fifo_ring_run1.txt
  fifo_ring_run2.txt

▶ The two files specify the same design
  ▶ with **same initial marking** = placement of tokens
  ▶ but **different initial firing times** of enabled nodes
    ▶ tokens can have different “lag” times at initialization
      = time before it contributes to the firing of nodes
    ▶ node fires only when all input tokens arrive
      → initial firing time of node = Max. of lag times of input tokens
    ▶ user specifies actual firing time of enabled nodes at initialization
      • system time starts at \( t = 0 \)
Tutorial 1: TSE analysis with des-tse
- Example 1a: FIFO ring

Step 4: Specifying different initial conditions

► Note the difference between the two specifications:

Run 1

... # initial firing time of # enabled nodes
.init b 0
.init d 0
...

Run 2

... # initial firing time of # enabled nodes
.init b 0
.init d 2
...

System time $t = 0$ at startup
Tutorial 1: TSE analysis with des-tse
- Example 1a: FIFO ring

Step 4: Specifying different initial conditions

► Run des-tse on both files and note the difference in results:
  > des-tse fifo_ring_run1.txt
  > des-tse fifo_ring_run2.txt

► Result of Run 1:

<table>
<thead>
<tr>
<th>Event pair</th>
<th>Max TSE</th>
<th>Min TSE</th>
</tr>
</thead>
<tbody>
<tr>
<td>(b, d)</td>
<td>6.0</td>
<td>0.0</td>
</tr>
</tbody>
</table>

► Result of Run 2:

<table>
<thead>
<tr>
<th>Event pair</th>
<th>Max TSE</th>
<th>Min TSE</th>
</tr>
</thead>
<tbody>
<tr>
<td>(b, d)</td>
<td>2.0</td>
<td>2.0</td>
</tr>
</tbody>
</table>

Note the significant difference in TSE results: caused by different initial conditions
**Tutorial 1: TSE analysis with des-tse**

- **Example 1a: FIFO ring**

Step 5: Performing different TSE queries

- Modify input files to perform TSE queries on different event pairs
- **Two options:**
  1. Use ".check all" to query TSE for all event pairs
  2. Use multiple ".check" lines

```
Original specification
...
# TSE pairs to check
.check b d
...
```

```
Modify with option 1
...
# TSE pairs to check
.check all
...
```

```
Modify with option 2
...
# TSE pairs to check
.check a a
.check a b
.check b c
...
```
Step 5: Performing different TSE queries

- Try out different options, run des-tse and observe results

- Example output from using “.check all” with fifo_ring_run2.txt:

<table>
<thead>
<tr>
<th>Event pair</th>
<th>Max TSE</th>
<th>Min TSE</th>
</tr>
</thead>
<tbody>
<tr>
<td>(a, a)</td>
<td>8.0</td>
<td>4.0</td>
</tr>
<tr>
<td>(a, b)</td>
<td>5.0</td>
<td>1.0</td>
</tr>
<tr>
<td>(a, c)</td>
<td>2.0</td>
<td>2.0</td>
</tr>
<tr>
<td>(a, d)</td>
<td>3.0</td>
<td>3.0</td>
</tr>
<tr>
<td>(b, a)</td>
<td>3.0</td>
<td>3.0</td>
</tr>
<tr>
<td>(b, b)</td>
<td>8.0</td>
<td>4.0</td>
</tr>
<tr>
<td>(b, c)</td>
<td>5.0</td>
<td>1.0</td>
</tr>
<tr>
<td>(b, d)</td>
<td>2.0</td>
<td>2.0</td>
</tr>
<tr>
<td>(c, a)</td>
<td>2.0</td>
<td>2.0</td>
</tr>
<tr>
<td>(c, b)</td>
<td>3.0</td>
<td>3.0</td>
</tr>
<tr>
<td>(c, c)</td>
<td>8.0</td>
<td>4.0</td>
</tr>
<tr>
<td>(c, d)</td>
<td>5.0</td>
<td>1.0</td>
</tr>
<tr>
<td>(d, a)</td>
<td>5.0</td>
<td>1.0</td>
</tr>
<tr>
<td>(d, b)</td>
<td>2.0</td>
<td>2.0</td>
</tr>
<tr>
<td>(d, c)</td>
<td>3.0</td>
<td>3.0</td>
</tr>
<tr>
<td>(d, d)</td>
<td>8.0</td>
<td>4.0</td>
</tr>
</tbody>
</table>
Tutorial 1: TSE analysis with des-tse

Example 1b: Micropipeline
In this tutorial we shall:

- Look at a bounded-delay system
- Run TSE analysis using the same steps as in Tutorial 1
Micropipeline design [Sutherland, Comm. of the ACM, '89]

Circuit diagram

Marked graph model (Delays not shown)
**Tutorial 1: TSE analysis with des-tse**

- Example 1b: Micropipeline

Step 1: Specifying the marked graph input

- Go to the tutorial1 directory created in Step 0:
  
  ```
  cd tutorial1
  ```

- Look at the DES input specification file:
  
  ```
  less micropipeline.txt
  ```

---

```plaintext
# list of nodes in graph
.node_list in c1 d1 c2 d2 c3 d3 out

# edge specifications
.edge in c1 5 10 1
.edge c1 d1 1 1 0
.edge d1 in 4 5 0
.edge c2 d2 1 1 0
.edge d2 c1 4 8 1
.edge c3 d3 1 1 0
.edge d3 out 4 5 0
.edge out c3 5 10 1

# initial firing time of enabled nodes
.init c1 0

# TSE pairs to check
.check in out
```
**Tutorial 1: TSE analysis with des-tse**

- Example 1b: Micropipeline

Step 2: Displaying the input specification

- Generate graphical output:
  > des-tse micropipeline.txt -o micropipeline.dot -no_processing

- Display it:
  > dotty micropipeline.dot

A window should pop up to display the following:
Tutorial 1: TSE analysis with des-tse
- Example 1b: Micropipeline

Step 3: Running TSE analysis

▶ Run the tool:
  > des-tse micropipeline.txt

▶ Look at the output:

<table>
<thead>
<tr>
<th>Event pair</th>
<th>Max TSE</th>
<th>Min TSE</th>
</tr>
</thead>
<tbody>
<tr>
<td>(in, out)</td>
<td>13.0</td>
<td>10.0</td>
</tr>
</tbody>
</table>

Step 4: Specifying different initial conditions

▶ Modify the initial conditions in input file as in Tutorial 1

▶ Run des-tse and observe results

Step 5: Performing different TSE queries

▶ Modify the TSE query section in input file as in Tutorial 1

▶ Run des-tse and observe results
Tutorial 2: Performance analysis with des-perf

Example 2: Micropipeline
In this tutorial we shall learn how to:

- **Step 1:** Specify a marked graph input for the des-perf tool
- **Step 2:** Display the input specification graphically
- **Step 3:** Run performance analysis
Step 1: Specifying the marked graph input

► Go to the tutorial2 directory created in Step 0:
  > cd tutorial2

► Look at the input specification file:
  > less micropipeline.txt

```plaintext
# list of nodes in graph
.nodes IN C1 D1 C2 D2 C3 D3 OUT

# edge list:
# .edge <input node> <output node>
.edge IN C1
.edge C1 D1
.edge D1 IN
.edge C2 D2
.edge D2 C1
.edge C3 D3
.edge D3 OUT
.edge OUT C3
```

```plaintext
# node list:
# .node <mean delay>
.node IN 10
.node C1 1
.node D1 5
.node C2 1
.node D2 5
.node C3 51
.node D3 5
.node OUT 10
```
Step 2: Displaying the input specification

- Generate a graphical output:
  ```
  des-perf micropipeline.txt -o micropipeline.dot -no_processing
  ```

- Display it:
  ```
  dotty micropipeline.dot
  ```

A window should pop up to display the following:
Tutorial 2: Performance analysis with des-perf
- Example 2: Micropipeline

Step 3: Run des-perf
▶ Run the tool:
  > des-perf micropipeline.txt
▶ Look at the output:

<table>
<thead>
<tr>
<th>SYMBOLIC STATE TABLE</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
</tr>
<tr>
<td>1</td>
</tr>
<tr>
<td>2</td>
</tr>
<tr>
<td>3</td>
</tr>
<tr>
<td>4</td>
</tr>
<tr>
<td>5</td>
</tr>
<tr>
<td>6</td>
</tr>
<tr>
<td>7</td>
</tr>
<tr>
<td>8</td>
</tr>
<tr>
<td>9</td>
</tr>
<tr>
<td>.....</td>
</tr>
<tr>
<td>.....</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>STATIONARY STATE DISTRIBUTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
</tr>
<tr>
<td>1</td>
</tr>
<tr>
<td>2</td>
</tr>
<tr>
<td>3</td>
</tr>
<tr>
<td>4</td>
</tr>
<tr>
<td>5</td>
</tr>
<tr>
<td>6</td>
</tr>
<tr>
<td>7</td>
</tr>
<tr>
<td>8</td>
</tr>
<tr>
<td>9</td>
</tr>
<tr>
<td>.....</td>
</tr>
<tr>
<td>.....</td>
</tr>
</tbody>
</table>
Tutorial 2: Performance analysis with des-perf
- Example 2: Micropipeline

Step 3: Run des-perf

- Two sections in the results table
  - Symbolic state table
    - State = a marking in the marked graph
      = placement of tokens on graph edges
    - Output representation:
      - Column 1: symbolic state
      - Column 2: edges with tokens in the state
  - Stationary state distribution
    - Output representation:
      - Column 1: symbolic state
      - Column 2: asymptotic probability of state

- Results can be further processed to give other useful results:
  - Average latency, throughput, etc.
Conclusions

- Two analysis tools under the DES Analyzer CAD package
  - des-tse
  - des-perf

- Used in the design flow for concurrent digital systems for
  - Verifying timing correctness
  - Measuring system performance
  - Getting feedback on performance bottlenecks for optimization