A Simulator for Integrated Optoelectronic Devices

Ramón Gutiérrez-Castrejón
Institute of Engineering, Univ. Nacional Autonoma de Mexico.

Marcus Duelk and Pietro Bernasconi
Bell Labs, Lucent Technologies, Holmdel, NJ, USA.
Outline

• Simulator for photonic integrated devices

• Programming language

• Application: Wavelength conversion at 40 Gb/s

• Single Gaussian pulse response

• System level simulations: PRBS signal response

• Conclusions
Simulation of new photonic integrated circuits

- New integration technology allows fabrication of sophisticated Photonics Integrated Circuits (PICs).

- Single SW tool must be able to deal with a wide variety of complex optoelectronic integrated circuits consisting of tens or even hundreds of interconnected devices.

- Solution: implementation of building blocks, each one performing a specific data-flow oriented function onto an input optical stream.

- Result: A simulator that is Flexible, Powerful and Intuitive
Solution: Modularity + Data Flowing

- Example of simulation units:

- Desirable characteristics:
  - High modeling accuracy
  - Modular, following I-O approach
  - Short processing time and low memory requirements
Novel semiconductor optical amplifier simulator

• Uni-directional approach: easy integration into system simulator and fast execution.

• Solved via a fast single-marching algorithm and integration of propagation equation in a (1+1)–dimension rectangular grid.

• Time domain representation allows straightforward incorporation of multiple channels. It is flexible in terms of applications.

• Non-linear effects: all carrier dynamical processes relevant for bit rates from 2.5 to 160 Gb/s (and higher) are considered through rate equation approach.

• Impact of amplified spontaneous emission on carrier dynamics is included.
LabVIEW graphical programming language

- Friendly graphical user interface (graphical source code).
- Modularity and interconnectivity of subroutines (VIs).
- Efficient management of libraries.
- Dataflow–oriented: data processing along wires.
- Allows interfacing with other programming languages.
Is LabVIEW fast Enough?  - YES

- Identical SOA simulators written in LabVIEW and Matlab: code optimized to each programming language.
- Benchmark: amplification of 40 Gb/s PRBS of Gaussian pulses 6.25ps FWHM. \( \Delta t=0.0976 \) ps, \( N=2^n \) bits long.

<table>
<thead>
<tr>
<th></th>
<th>LabVIEW (LV)</th>
<th>Matlab (ML)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Runge-Kutta</td>
<td>Runge-Kutta</td>
<td>ODE15s</td>
</tr>
<tr>
<td>Interp1 Library</td>
<td>Interp1</td>
<td>Interp1</td>
</tr>
<tr>
<td>Interp2 Self-W</td>
<td>Interp2 Self-W</td>
<td>Interp2 Self-W</td>
</tr>
</tbody>
</table>

Identical simulators
LabVIEW 7.1 versus Matlab 7.0.1

The graph compares computational time [sec] versus the number of bits for different methods:
- LV RK interp1
- LV RK interp2
- ML RK interp1
- ML RK interp2
- ML ODE15 interp1
- ML ODE15 interp2

The graph shows a linear relationship between computational time and the number of bits for each method.
LabVIEW 7.1 versus Matlab 7.0.1 … cont’d

<table>
<thead>
<tr>
<th>Method</th>
<th>Slope</th>
<th>Slope Error</th>
<th>Rel. Speed factor</th>
</tr>
</thead>
<tbody>
<tr>
<td>LV RK Interp1 ___</td>
<td>0.1608</td>
<td>7.8E-05</td>
<td>1.0</td>
</tr>
<tr>
<td>LV RK Interp2</td>
<td>0.1884</td>
<td>1.0E-05</td>
<td>1.17</td>
</tr>
<tr>
<td>ML RK Interp1 ___</td>
<td>15.298</td>
<td>1.6E-02</td>
<td>95.1</td>
</tr>
<tr>
<td>ML RK Interp2</td>
<td>1.8611</td>
<td>1.7E-03</td>
<td>11.6</td>
</tr>
<tr>
<td>ML ODE15s Interp1</td>
<td>5.6578</td>
<td>8.7E-03</td>
<td>35.2</td>
</tr>
<tr>
<td>ML ODE15s Interp2</td>
<td>3.8703</td>
<td>5.2E-03</td>
<td>24.1</td>
</tr>
</tbody>
</table>

- 10 times outperformance of LV over ML for identical sims.
- LV interpolant speeds up, ML interpolant slows down.
- Best LV simulator is 11 times faster than any ML simulator.
- Fastest ML simulator: entirely self-written (ML RK Interp2).
LabVIEW 8.2 versus Matlab 7.2.0.232 - Update

The diagram compares computational time in seconds for different methods (LV RK interp1, LV RK interp2, ML RK interp1, ML RK interp2, ML ODE15 interp1, ML ODE15 interp2) as a function of the number of bits.

The x-axis represents the number of bits, ranging from 0 to 1100. The y-axis represents computational time in seconds, ranging from 0 to 8000.

Legend:
- LV RK interp1
- LV RK interp2
- ML RK interp1
- ML RK interp2
- ML ODE15 interp1
- ML ODE15 interp2

Graph shows a trend where computational time increases with the number of bits, with the methods differing in their performance characteristics.
LabVIEW 8.2 versus Matlab 7.2.0.232  

<table>
<thead>
<tr>
<th>Method</th>
<th>Slope</th>
<th>Slope Error</th>
<th>Rel. Speed factor</th>
</tr>
</thead>
<tbody>
<tr>
<td>LV RK Interp1</td>
<td>0.1723</td>
<td>2.0E-05</td>
<td>1.0</td>
</tr>
<tr>
<td>LV RK Interp2</td>
<td>0.2011</td>
<td>6.0E-05</td>
<td>1.2</td>
</tr>
<tr>
<td>ML RK Interp1</td>
<td>16.2746</td>
<td>8.7E-02</td>
<td>94.5</td>
</tr>
<tr>
<td>ML RK Interp2</td>
<td>0.862</td>
<td>1.3E-03</td>
<td>5.0</td>
</tr>
<tr>
<td>ML ODE15s Interp1</td>
<td>4.9668</td>
<td>8.5E-03</td>
<td>28.8</td>
</tr>
<tr>
<td>ML ODE15s Interp2</td>
<td>3.0566</td>
<td>3.8E-03</td>
<td>17.7</td>
</tr>
</tbody>
</table>

- Qualitative results remain the same.
- ML shows appreciable improvement in contrast to LV.
- For identical sims. LV still outperforms ML, now by 4.16.
- Apparently ML interpolant is still a bottleneck.
Active Mach-Zehnder interferometer: LabVIEW code

- Upper port: PRBS @ $\lambda_{\text{sig}} = 1565.5 \text{ nm} (~191.5 \text{ THz}), 40 \text{ Gb/s}, \text{RZ-Gaussian puses}, 8.25 \text{ ps FWHM}.
- Lower port: CW signal @ $\lambda_{\text{target}} = 1554.1 \text{ nm} (~192.9 \text{ THz}).
- Identical 1 mm long SOAs with 400 mA of injected current.
Single Gaussian pulse: transmission window

- Short delays: short and narrow AMZI transmission window.
- Long delays: flat-top or even double peaked windows.
Single Gaussian pulse: output power

- Best results when delay equals input pulse width: 8.25 ps.
- For this delay the pulse width is preserved.
Simulated eye diagrams for PRBS 256-bits long

Differential delay = 8.25 ps

I₁ = 400 mA  I₂ = 400 mA
I₁ = 399 mA  I₂ = 400 mA

• Fine-tuning of I₁ reduces noise in marks and spaces.
Tuning of $I_1$ exhibits quasi-periodic behavior

$I_1 = 381 \text{ mA} \quad I_2 = 400 \text{ mA}$

$I_1 = 361 \text{ mA} \quad I_2 = 400 \text{ mA}$

$I_1 = 341 \text{ mA} \quad I_2 = 400 \text{ mA}$

$I_1 = 320.5 \text{ mA} \quad I_2 = 400 \text{ mA}$

$I_1 = 298.5 \text{ mA} \quad I_2 = 400 \text{ mA}$

$I_1 = 277 \text{ mA} \quad I_2 = 400 \text{ mA}$
Simulation results

• The high execution speed of our simulator allowed several system-level simulations with $\Delta I_1=0.5$ mA.
• Eye diagrams with reduced noise found at: $I_1=381, 361, 341, 320.5, 298.5$ and $277$ mA.
• Apparent quasi-periodic behavior, experimentally confirmed.
• The eyes are not identical: lower values of $I_1$ produced weaker pulses with lower amplitude jitter (patterning).
• Varying the splitting ratio of the leftmost MMI while keeping $I_1=I_2=400$ mA also led to noise reduction.
Conclusion

• We presented a novel graphical simulator for multiple photonic integrated circuits. Attractive alternative.

• Modular I-O approach: circuit elements associated with individual simulation units that are intuitively wired.

• Written in LabVIEW: more than 4 times computational speed outperformance over Matlab (SOA simulator).

• Application example: 40 Gb/s wavelength conversion using an active Mach-Zehnder interferometer.

• Simulator shows that best results are obtained when:
  • Differential delay closely matches the pulse width.
  • Unbalanced operation (quasi-periodicity observed).