Modelling Optical Devices and Systems in MAPP

Tianshi Wang
Jaijeet Roychowdhury

Electrical Engineering and Computer Sciences
University of California at Berkeley

Technical Report No. UCB/EECS-2017-160
http://www2.eecs.berkeley.edu/Pubs/TechRpts/2017/EECS-2017-160.html

October 17, 2017

Modelling Optical Devices and Systems in MAPP

Tianshi Wang and Jaijeet Roychowdhury
Department of Electrical Engineering and Computer Sciences, University of California, Berkeley, CA, USA
Email: {tianshi, jr}@berkeley.edu

***Abstract*—In this technical report, we illustrate how optical devices and systems are formulated and implemented in the Berkeley Model and Algorithm Prototyping Platform (MAPP). With examples and their code snapshots, this report documents the current structure and capabilities of MAPP's optical modules. It is also useful as a quick-start guide for new users working with optical modelling and simulation in MAPP.**

I. Concepts behind Optical Modelling in MAPP

Similar to electrical circuits, an optical system is modelled as a set of optical devices interacting with each other through node connections. One simple example is shown in Fig. 1 (from [1]) — a communication link where an optical fiber connects a laser diode and a photodiode.

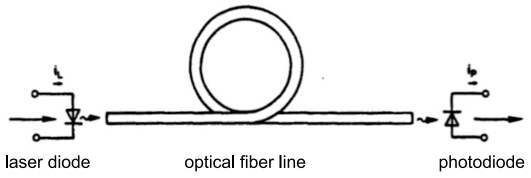


Fig. 1. A simple intensity-modulated optical fiber communication link, adapted from Fig 1.1 in [1].

The link has connections to electrical circuits around it, which is typical for most practical optical systems. But to illustrate our ideas on optical modelling, we consider only the optical parts in this report. We can then abstract the laser diode as an ideal light source and photodiode an ideal light sink which doesn't emit or reflect any light. The abstracted system diagram of the communication link is shown in Fig. 2.

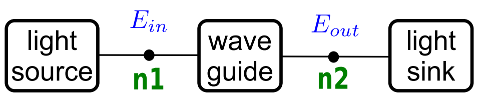


Fig. 2. System diagram for the purely optical parts in the simple optical fiber communication link in Fig. 1.

While the block diagram in Fig. 2 seems straightforward, with optical devices represented by blocks and connections drawn as line segments in between, several questions need to be answered before the system is fully defined. For example, what does an optical connection mean? Is such a connection directional? Furthermore, what is inside of the blocks? In the remainder of this section, we answer these questions by clarifying a few concepts related to optical modelling.

A. Optical Connection

To begin with, the modelling of an optical connection should not be confused with the physical device of an optical fiber connector. Proper modelling of a physical optical connector requires taking into account a lot of factors. Its performance depends on the alignment of the centers of the connected fibers, the angle between them, as well as the quality of the assembly and polishing operations. All these factors affect the attenuation and reflection of light. Moreover, different models have to be deployed in the cases of multimode and single-mode fibers, or in the context of long-haul communication and nano-scale silicon photonics.

Therefore, it becomes clear to us that a physical optical connection, *e.g.*, a fiber connector, is better modelled as a device rather than an optical connection used in diagrams like Fig. 2. Instead, we define a simpler, artificial connection to use in system diagrams. We define that when two optical ports are connected with each other, all the optical properties on one side of the connection are equal to the corresponding properties on the other side. Under this definition, an optical port can be thought of as the left or right side of a cross-section of a fiber or waveguide, and an optical connection of two ports merely describes the *continuation of light* through the media. Note that this definition of an optical connection allows only two optical ports to be connected. In the case of connecting three fibers, a three-port device like a splitter or a joiner is needed and all the physical factors are considered in the modelling of such devices.

B. Optical IOs at an Optical Port

In this subsection, we define the optical quantities modelled at an optical connection of two ports, and explain our reasons. In electrical circuits, the consensus is that the inputs/outputs (IOs) at an electrical port are voltages and currents; Kirchhoff's current law and voltage law (KCL and KVL) is used to write the system-level equations. However, no such standard is agreed upon for optical modelling at this moment. Instead, people often use different optical quantities in their models for different applications.

In laser modelling, people often use only the intensity of light as the IO to model at the optical port of a laser. But it quickly becomes insufficient for applications in optical communication, where the phase of light is also required for the model to be meaningful. Moreover, in cases like Wavelength-Division Multiplexing (WDM) fiber-optic communication, intensities and phases of light with multiple frequencies or spatial modes all need to be taken into consideration. But even all these may still not be sufficient. While in optical communication we often assume that light waves with multiple frequencies won't interfere with each other during transmission and the number of frequencies is fixed, this is not true in many other applications. When light passes through nonlinear photonic devices, *e.g.*, second-harmonic generators or optical mixers, new frequency components can be generated. To make situation more complicated, frequencies used in the system can also change with time. All of these considerations add up to the difficulty of the design of a general framework for the modelling of optical devices and systems.

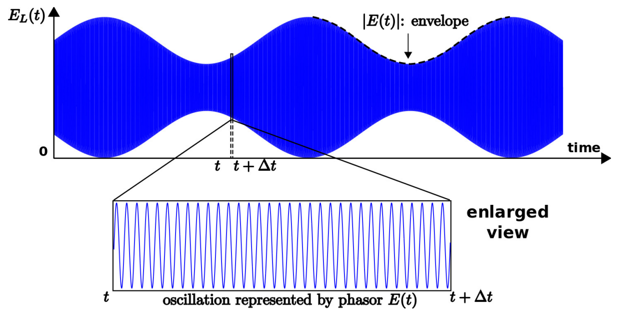


Fig. 3. Illustration of electric field of light with modulation. As the envelope is changing at a much slower speed, at any time, the electric field of light can be considered to be oscillating at a steady state.

In MAPP[2], [3],[1] we model the optical properties at an optical connection using both magnitude $A(t)$ and phase $\Phi(t)$ information of the light. Specifically, in the implementation, we model a beam of light at a certain frequency as a complex number, or a phasor $E(t)$. Note that a phasor by definition represents the steady state of an oscillation. Then what does a time-varying phasor $E(t)$ even mean?

Normally we model a light wave by considering its electric field. Fig. 3 depicts the changing electric field of light $E_L(t)$ in time domain [2]. This electric field of light $E_L(t)$ is oscillating at high frequencies (around 200 THz assuming wavelength is around 1550 nm). Its value is difficult to measure in reality, nor is it normally directly used in practice. Instead, we usually modulate the light using external electrical signals, whose frequencies are normally not higher than hundreds of GHz. A typical waveform of a modulated light in optical systems is shown in Fig. 3. As the frequencies of light and electrical signals are widely separated by a factor of thousands, the waveform as in Fig. 3 develops an envelope and it is this envelope of light that is used in power or signal transmission applications. At any time t, within a small time interval Δt, we consider the electric field of light to have reached its periodic steady state. The magnitude and phase of the oscillation can be assumed to be constant during the time interval Δt and we denote them as $A(t)$ and $\Phi(t)$ respectively. In simulation, as $\Phi(t)$ often has multiple solutions separated by $k \cdot 2\pi, k \in \mathbf{Z}$, it is not suitable as a system unknown for non-linear solvers. Instead, we use a complex number $E(t)$ or its real and imaginary parts as the terminal properties of an optical device. In this way, the time constant of $E(t)$ is comparable with that of the electric circuitry around the optical device.

Note that the complex number $E(t)$ used to represent the envelope of light is more like a frequency-domain property compared with voltages or currents in electrical circuits. So the simulation of optoelectronic systems is similar to mixed-time-frequency-domain simulation, such that light with very high frequency and electrical signals with much slower dynamics can both be efficiently simulated at the same time.

II. Optical Device Models

Using the mixed-time-frequency-domain property $E(t)$ to represent optical IOs, we take a waveguide as an example and develop optical models at different levels.

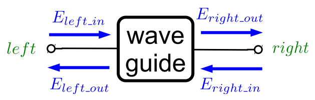


Fig. 4. Diagram of a waveguide model at Level I.

Level I: A diagram of a waveguide is shown in Fig. 4 with two optical ports $left$ and $right$. At this level we consider the simplest scenario with a single frequency and single mode of light transmitting in the waveguide. Even in this case, because of the nature of light, at one optical port there can be two light waves — the incoming and outgoing waves superimposed on each other. As is shown in Fig. 4, four complex numbers constitute all the IOs of the waveguide model: E_{left_in}, E_{left_out}, E_{right_in} and E_{right_out}.

Then we can write waveguide equations using these IOs. The simplest equations describing the magnitude decay and the phase shift caused by the waveguide are shown as follows.

$$\begin{aligned} E_{right\_out} &= E_{left\_in} \cdot e^{-\alpha \cdot L} \cdot e^{j \cdot 2\pi \cdot f \cdot \frac{L \cdot n}{c}}, & (1) \\ E_{left\_out} &= E_{right\_in} \cdot e^{-\alpha \cdot L} \cdot e^{j \cdot 2\pi \cdot f \cdot \frac{L \cdot n}{c}}, & (2) \end{aligned}$$

where L is the length of the waveguide, n is the group index, α is the decay factor of the waveguide and c is the speed of light. f in (1) and (2) is the frequency of light. But unlike the parameters or constants above, f is not a model parameter. It is determined by the optical system and passed to the ModSpec object of waveguide through its Network Interface Layer (NIL) [5].

Similar to the electrical domain scenario, for the equation engine to associate IOs with their meanings, the NIL of this waveguide model will return IOs' names $\{E_{left\_in}$, E_{left_out}, E_{right_in}, $E_{right\_out}\}$; IOs' corresponding nodes $\{left, right, left, right\}$; IOs' directions $\{in, out, in, out\}$. Apart from these properties, NIL also has a field that returns frequency f for the ModSpec functions to use. This field is set up by the system before evaluating waveguide equations (1) and (2).

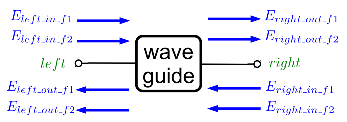


Fig. 5. Diagram of a waveguide model at Level II.

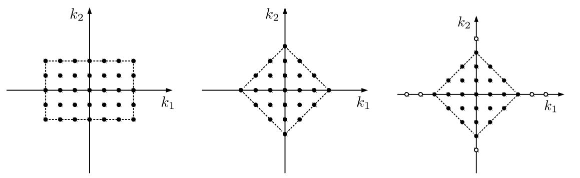
Level II: As mentioned before, in many applications such as WDM communication, light waves at multiple frequencies are superimposed at the optical ports of

[1]MAPP is open-source released at [4].

[2]$E_L(t)$ is the electric field of the light at z direction, which is along the guiding direction

waveguides. In this scenario, instead of using one complex number to represent the incoming or outgoing wave, we need a vector of such complex numbers. As shown in Fig. 5, when two frequencies f_1 and f_2 are present in the system, $E_{left\_in\_f1}$ and $E_{left\_in\_f2}$ are used to represent the frequency components at f_1 and f_2 of the incoming light wave at port $left$. In the code implementation of such a ModSpec device, NIL is designed not to return a single frequency value, but a vector of frequency values and an array of their names. With the help of this frequency information, the IOs and waveguide equations can then be set up in a similar way as in Level I.

However, there is a practical concern when we introduce multiple frequencies into the system. A nonlinear optical device can often generate second and higher harmonics from an incoming wave with a certain frequency. This is to say, if f_2 is equal to $2f_1$, the device will need to know this information in order to correctly assign outgoing waves to other ports. If only names and values of the frequencies are available, the device will have to check the values numerically to determine the relationship between frequencies, *e.g.*, which frequency seems like the second harmonic of f_1. Alternatively, we can enforce certain naming conventions for such relationships to be inferred from frequencies' names. But neither of these schemes is robust. So in MAPP, instead of using frequency names and values, we represent *system frequencies* as harmonics or intermodulations of a set of *fundamental frequencies*. In this way, NIL is designed to return fundamental frequency names and values and a list of indices to construct all the frequencies used in the optical system. For example, there may be two fundamental frequencies in the system, namely f_1 and f_2. If freq_index is set to be $[1, 0; 0, 1; 1, 1]$, then three frequencies will be considered in the simulation: f_1, f_2 themselves as well as one intermodulation term $f_1 + f_2$. To include the second harmonics, we can add entries $[2, 0]$ and $[0, 2]$ to `freq_index` so that $2f_1$, $2f_2$ will also be available in the system. In general, if we have d fundamental frequencies with values f_1 to f_d, then a row $[k_1, k_2, \cdot, k_d]$ will represent the frequency $\sum_i^d f_i \cdot k_i$. Using `freq_index`, the ModSpec object of device can then be able to relate different harmonic and intermodulation terms inside its equations.



(a) Box truncation: $M_1 = 3, M_2 = 2$. (b) Diamond truncation: $M = 3$. (c) Hybrid truncation: $M = 3, M_1 = 5, M_2 = 4$.

Fig. 6. Three types of frequency truncations when the system has two-fundamental frequencies.

Note that `freq_index` doesn't have to be set up completely manually. When the system has only one fundamental frequency, `freq_index` is normally a column vector with entries from 1 to the highest harmonic order of interest to designers. In cases with multiply fundamental frequencies, we can use box truncation, diamond truncation or more complex methods to select frequencies from the multi-frequency space.

$$\text{box truncation: } |k_1| \leq M_1, \cdots, |k_d| \leq M_d. \quad (3)$$

$$\text{diamond truncation: } |k_1| + \cdots + |k_d| \leq M. \quad (4)$$

Fig. 6(a) and Fig. 6(b) illustrate box and diamond truncations respectively. Fig. 6(c) shows a hybrid truncation which combines the shapes of a cross and a diamond. Hybrid truncation is suitable for moderately non-linear optical systems where intermodulation is relatively weak. In this case, both M_i and M have to be specified to define the number of harmonics for each tone and the maximum order of intermodulation respectively.

At this level, we have split one electric wave into a vector of them at different frequencies. Similar separation can be performed for spatial modes, polarizations of light, *etc.*, expanding the scope of optical modelling when needed.

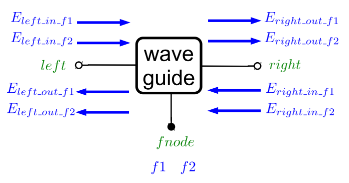


Fig. 7. Diagram of a waveguide model at Level III.

Level III: In the previous levels, we treat frequencies or fundamental frequencies as system-level parameters specified externally by users. This approach is valid under the assumption that fundamental frequency values do not change over time, which is true for most optical simulation applications. However, there are exceptions. For example, the frequency of the output of a laser may vary as temperature changes and temperature may rely on the heat emitted by the laser as well as the rest of the system. Therefore, without simulating the system as a whole we cannot determine the frequency *a priori*. If we take this scenario into consideration, fundamental frequencies then become system unknowns instead of parameters. Therefore, we introduce a frequency node into the system. When optical devices connect to this node, their vector of IOs will also contain the system's fundamental frequencies. The devices' equations can use or assign values to these frequencies.

We illustrate this idea also using the waveguide example. As shown in Fig. 7, we connect the model's $fnode$ port to the system's frequency node. The ModSpec object can get the time-varying fundamental frequency values, use `freq_index` to calculate all the frequencies of interests, then use them to evaluate its functions.

MAPP implements *Level III* for optical device modelling. At the same time, MAPP also supports devices without a connection to a frequency node, in which case the models become compatible with the other two lower levels. In the remainder of this section, we illustrate the actual implementation of MAPP's optical device models in more detail.

In MAPP, device models are specified as differential algebraic equations (DAEs) in the ModSpec format [2], [6]:

$$\vec{z} = \frac{d}{dt}\vec{q}_e(\vec{x}, \vec{y}) + \vec{f}_e(\vec{x}, \vec{y}, \vec{u}) \quad (5)$$

$$0 = \vec{w} = \frac{d}{dt}\vec{q}_i(\vec{x}, \vec{y}) + \vec{f}_i(\vec{x}, \vec{y}, \vec{u}) \quad (6)$$

where the variables $\vec{x}$, $\vec{z}$ are IOs, with $\vec{z}$ being those that can be expressed explicitly in equations; $\vec{y}$ contains non-IO, internal unknowns in the model; functions $\vec{f}_e$, $\vec{q}_e$, $\vec{f}_i$, $\vec{q}_i$ are the differential and algebraic parts of the model's explicit and implicit DAEs that describe the relationship between all the variables.

In optical models, IOs consist of two parts: the incoming/outgoing light waves at optical ports, and the fundamental frequencies at the frequency node [3]. The optical model's role is to describe the relationship between these IOs (frequencies and incoming/outgoing waves) with equations.

Unlike in electrical devices, the number of IOs in an optical device depends on the amount of frequencies of interest to the system-level simulation. An optical device gets such information through its NIL. As illustration, assume the system has two fundamental frequencies, namely `ff1` and `ff2`, and two operating frequencies `f1=ff1`, `f2=ff2`. Then the NIL of the waveguide model has the following fields.

```
NIL
  .NodeNames      {'port1', 'port2'}
  .FreqNodeNames  {'freq'}

  .FreqNames      {'f1', 'f2'}
  .FundFreqNames  {'ff1', 'ff2'}
  .LocalFundFreqNames {}
  .FundFreqVals   [200e15; 201e15]
  .FreqIdx        [1, 0; 0, 1]

  .IOnames {'port1_f1_in','port1_f1_out','port1_f2_in',...
            'port1_f2_out','port2_f1_in','port2_f1_out',...
            'port2_f2_in','port2_f2_out','ff1_scale',...
            'ff2_scale'}
  .IOtypes      {'E','E','E','E','E','E','E','E',...
                 'fscale','fscale'}
```

From the NIL of the waveguide, we can see that IOs consist of in/out waves (at each port, at each frequency), and also variables named fscales. An fscale is the ratio between a fundamental frequency and its nominal value (specified in NIL.FundFreqVals)[4]. Part of the NIL is set up by the system (*e.g.*, FreqNames, FundFreqNames, FreqIdx, *etc.*), while part of it is set up by the device (*e.g.*, NodeNames, FreqNodeNames, *etc.*). To know about the full definition of optical NIL, users can run the following command in MAPP.

```
help attach_optical_NIL
```

With the IOs specified, we can then set up the device's ModSpec APIs by specifying explicit outputs ($\vec{z}$), the other IOs ($\vec{x}$), and model functions ($\vec{f}_e$, $\vec{q}_e$, $\vec{f}_i$, $\vec{q}_i$). Selected fields of the waveguide ModSpec object are shown below:

```
MOD
.ModelName: waveguideModSpec
.parmnames: {'Alpha', 'L', 'n'}
.parmdefaults: {288, 0, 4.1963}
.ExplicitOutputNames: outgoing waves (based on NIL)
.OtherIONames: incoming waves and fscales
.InternalUnkNames: {}
.uNames: {}

.fe:
    fundfreqnames = feval(MOD.opticalNIL.FundFreqNames, MOD);
    nfundfreqs = length(fundfreqnames);
    fundfreqvals = feval(MOD.opticalNIL.FundFreqVals, MOD);
    freqidx = feval(MOD.opticalNIL.FreqIdx, MOD);

    % separate E-field and frequency parts from vecX
    nvecX = length(vecX);
    Es = vecX(1:nvecX-nfundfreqs);
    nE = nvecX-nfundfreqs;
    fscales = vecX((nvecX-nfundfreqs+1):end);

    % get freqs from fundfreq and freqidx
    freqs = freqidx * (fscales .* fundfreqvals);

    % calculate outgoing waves (explicit outputs)
    feout(1:nE/2) = Es(nE/2+1:end) .* exp(-Alpha*L)...
                .* exp(2*pi*(0+1i) .* freqs .* (L*n/c));
    feout(nE/2+1:end) = Es(1:nE/2) .* exp(-Alpha*L)...
                .* exp(2*pi*(0+1i) .* freqs .*

.qe: return all zeros
.fi: []
.qi: []
```

[3]MAPP currently supports only one frequency node in the system, which is to say all devices share the same frequencies in simulation.

[4]Such design improves the numerical properties of the system DAE compared with using fundamental frequency values directly as unknowns.

Note that the model function $\vec{f}_e$ is written to handle in/out waves and fscales as vectors. And the exact size of waves and frequencies modelled in the device comes from its NIL, which depends on the system settings of operating frequencies. When implemented in this low-level method by setting up ModSpec APIs directly, most optical devices have to be written in this vector form. To make it more convenient for users, we are currently developing higher-level ModSpec wrappers for optical devices that will come in later releases.

III. Optical Netlists

After defining optical models, the connection between them can be described using a netlist similar to those used for electrical circuits. The netlist is a Matlab structure. It contains a number of data fields that are normally set up as follows:

1) The first two fields in the netlist are:

```
.name % A name for the optical system (string).
.nodenames % Names of all the optical nodes
           % (cell array of strings).
```

They are normally set up directly. For example:

```
netlist.name = 'example';
netlist.nodenames = {'n1', 'n2', 'n3'};
```

2) The optical netlist also contains a frequency node. Associated with this frequency node there are a few data fields describing the frequencies used in the simulation of the optical system (*e.g.*, `freq_names`, `freq_idx`, `fund_freq_names`, *etc.*). However, users don't have to deal with these fields directly. They can specify all of them by calling a utility function `add_freq_node`. To learn about the definitions of these data fields as well as the usage of `add_freq_node`, see

```
help add_freq_node
```

3) Then devices/elements are added to the netlist by specifying the field `netlist.elements`. Similar to adding electrical devices in circuits, users can use `add_element` to enter a device in an optical system. While the utility function `add_element` for optical systems is the same as the one used in circuits, there is one difference in its usage. When specifying the nodes of the element, because there are both optical

nodes and frequency node in the system, these two types of nodes can be specified together. To be exact, the following arguments are all acceptable as nodes in `add_element`:

```
{'n1', 'n2'} or {{'n1', 'n2'}}
    % when the device doesn't connect to freq node
{{'n1', 'n2'}, {'fn1'}}
    % when the device connects to freq node, optical
    % nodes and freq node are organized in separated
    % cell arrays with freq node in the end
{{'n1', 'n2'}, {'fn1', {'f1', 'f2'}}}
    % when the device has to provide equations for
    % some or all of the fund freqs at the freq node,
    % the names of these fund freqs are specified in
    % a cell array after freq node name.
```

The following example code illustrates the above procedures by specifying a netlist structure for the system in Fig. 2.

```
% 1. name and nodenames:
netlist.name = 'LS-WG-Sink';
netlist.nodenames = {'n1', 'n2'};

% 2. frequency node:
netlist = add_freq_node(netlist, 'fn1',...
            'fundfreqnames', {'f1', 'f2'}, ...
            'nominal_vals', [200e12; 201e12], ...
            'truncation', 'fund_only');

% 3. elements:
netlist = add_element(netlist, lightsourceModSpec(), 'LS',...
                              {{'n1'}, {'fn1', {'f1', 'f2'}}});
netlist = add_element(netlist, waveguideModSpec(), 'WG',...
                        {{'n1', 'n2'}, {'fn1'}}, {{'L', 1e-5}});
netlist = add_element(netlist, lightsinkModSpec(), 'Sink',...
```

Such a netlist contains all the information about device models and their connection; an optical equation engine can then use it to generate system DAEs.

IV. Optical Equation Engine

With optical models defined and their connection specified in netlists, an optical equation engine can formulate the system equations just by equating the incoming and outgoing waves of two connected optical ports.

When the outgoing wave of a port and the corresponding incoming wave of the connected port are equal to each other, they can be represented with only one unknown variable in the final equation system. In this way, for the system in Fig. 2 with 4 optical ports and 1 frequency, instead of having 8 optical quantities, only 4 unknown variables are needed to model the system:

$$\begin{aligned}\vec{x} =& [E\_n1\_forward, E\_n1\_backward, \\ & E\_n2\_forward, E\_n2\_backward]^T\end{aligned} \quad (7)$$

where the subscription $forward$ indicates the light is transmitting to the right of the diagram [5]. As the equality between incoming and outgoing waves are implicit in the choice of unknowns when building system DAEs, the optical equation engine can then simply call each device's equations to set up system DAEs:

$$\begin{aligned}
\text{LS:}\quad & E\_n1\_forward = \vec{u} && (8)\\
\text{WG-1:}\quad & E\_n2\_forward = E\_n1\_forward \cdot \\
& e^{-\alpha \cdot L + j \cdot 2\pi \cdot f \cdot \frac{L \cdot n}{c}} && (9)\\
\text{WG-2:}\quad & E\_n1\_backward = E\_n2\_backward \cdot \\
& e^{-\alpha \cdot L + j \cdot 2\pi \cdot f \cdot \frac{L \cdot n}{c}} && (10)\\
\text{Sink:}\quad & E\_n2\_backward = 0 && (11)
\end{aligned}$$

With this simple optical example, we have illustrated the modelling of optical connection, model, system netlist and the equation engine used to connect them together and create system DAEs — all within the MAPP framework and in a way intended to be most appropriate and natural for the optical domain. The use of equation engine allows optical devices to be connected in a way as convenient as putting together circuit components, separating the routines for individual device modelling and integrated system construction in a modular way, enabling the modelling of more complicated optical systems.

V. Optical System Examples

In this section we provide a couple of optical system examples currently implemented in MAPP.

A. Mach-Zehnder Modulator

In a Mach-Zehnder modulator, a waveguide is split up into two waveguide arms. If the two arms have slightly different group indices (*e.g.*, by applying a voltage to one arm), a phase shift difference is induced for the waves passing through them. When the two arms are recombined, the phase difference is converted to an amplitude modulation. The diagram of a Mach-Zehnder modulator is shown in Fig. 8.

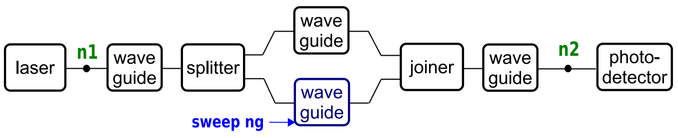


Fig. 8. Diagram of a Mach-Zehnder Modulator. Laser is modelled as an ideal light source, while the photodetector an ideal light sink.

Similar to the example in Sec. III, the optical system in Fig. 8 can be modelled in MAPP by specifying a netlist structure.

```
netlist.name = 'Mach Zehnder modulator';
netlist.nodenames = {'n1', 'rl', 'b1l', 'b2l', ...
                    'b1r', 'b2r', 'rr', 'n2'};

netlist = add_freq_node(netlist, 'fn1', ...
            'fundfreqnames', {'f1', 'f2'}, ...
            'nominal_vals', [200e12; 210e12], ...
            'truncation', 'fund_only');

netlist = add_element(netlist, lightsourceModSpec(), 'LS',...
        {{'n1'}, {'fn1', {'f1', 'f2'}}}, {});
netlist = add_element(netlist, waveguideModSpec(), 'WG1',...
        {{'n1', 'rl'}, {'fn1'}}, {{'L', 1e-5}});
netlist = add_element(netlist, splitterModSpec(), 'S1',...
        {{'b1l', 'b2l', 'rl'}, {'fn1'}});
netlist = add_element(netlist, waveguideModSpec(), 'WGup',...
        {{'b1l', 'b1r'}, {'fn1'}}, {{'L', 1e-5}});
netlist = add_element(netlist, waveguideModSpec(), ...
        'WGdown', {{'b2l', 'b2r'}, {'fn1'}}, ...
```

[5]This is a convention set up in the optical equation engine. Our implementation sets up $forward$ and $backward$ based on the order in which devices are added into the netlist. The outgoing wave of the device connected to the system first is named forward wave.

```
        {{'L', 1e-5}, {'n', 4}});
netlist = add_element(netlist, splitterModSpec(), 'S2',...
        {{'b1r', 'b2r', 'rr'}, {'fn1'}});
netlist = add_element(netlist, waveguideModSpec(), 'WG2',...
        {{'rr', 'n2'}, {'fn1'}}, {{'L', 1e-5}});
netlist = add_element(netlist, lightsinkModSpec(), 'Sink',...
       {{'n2'}, {'fn1'}});
```

After specifying the netlist, we can then convert it into system DAEs using optical equation engine by issuing the following commands.

```
DAE = opticalEqnEngine(netlist);
DAE.unknames(DAE)
% unk names displayed in comments below:
% 'n1_f1_forward' 'n1_f1_backward' 'n1_f2_forward'
% 'n1_f2_backward' 'r1_f1_forward' 'r1_f1_backward'
% 'r1_f2_forward' 'r1_f2_backward' 'b1l_f1_forward'
% 'b1l_f1_backward' 'b1l_f2_forward' 'b1l_f2_backward'
% 'b2l_f1_forward' 'b2l_f1_backward' 'b2l_f2_forward'
% 'b2l_f2_backward' 'b1r_f1_forward' 'b1r_f1_backward'
% 'b1r_f2_forward' 'b1r_f2_backward' 'b2r_f1_forward'
% 'b2r_f1_backward' 'b2r_f2_forward' 'b2r_f2_backward'
% 'rr_f1_forward' 'rr_f1_backward' 'rr_f2_forward'
% 'rr_f2_backward' 'n2_f1_forward' 'n2_f1_backward'
% 'n2_f2_forward' 'n2_f2_backward'
% 'fn1_f1_scale' 'fn1_f2_scale'
```

The netlist contains 8 optical nodes and 1 frequency node. From the add_freq_node statement in the netlist we can see that the system considers two frequencies in simulation. Then we know that each node has forward, backward waves at two frequencies, and the system unknowns then consist of $8 \times 2 \times 2 = 32$ waves and 2 fscale variables representing the ratios between fundamental frequencies in the system and their nominal values.

With the optical DAE, various analyses can be performed on it. The code below uses `dot_dcsweep` to run DC sweep on one of the arm's group index, and plot the magnitude modulations at the two frequencies. Results are plotted in Fig. 9.

```
DAE = DAE.set_uQSS('LS:::f1_scale_u', 1, DAE);
DAE = DAE.set_uQSS('LS:::f2_scale_u', 1, DAE);
DAE = DAE.set_uQSS('LS:::port_f1_u', 1, DAE);
DAE = DAE.set_uQSS('LS:::port_f2_u', 1, DAE);

swp = dot_dcsweep(DAE, [], 'WGdown:::n', 4, 4.2, 70);
[ns, sols] = swp.getSolution(swp);

idx = unkidx_DAEAPI('n2_f1_forward', DAE);
plot(ns, abs(sols(idx, :)), '.-r');
hold on;
idx = unkidx_DAEAPI('n2_f2_forward', DAE);
plot(ns, abs(sols(idx, :)), '.-b');
legend('n2\_f1\_forward', 'n2\_f2\_forward');
xlabel('WGdown:::n'); ylabel('magnitudes'); grid on;
```

B. Ring Resonator

Similar to the Mach-Zehnder modulator example, a silicon ring resonator [7] can also be modelled in MAPP as a network of several optical components, as shown in Fig. 10.

The netlist is constructed in the same way as the Mach-Zehnder modulator example, by the following script.

```
netlist.name = 'ring resonator demo';
netlist.nodenames = {'n1','n2','in1','in2','out1','out2'};

netlist = add_freq_node(netlist, 'fn1',...
               'fundfreqnames', {'f1'}, ...
               'nominal_vals', 193e12, ...
               'truncation', 'fund_only');
```

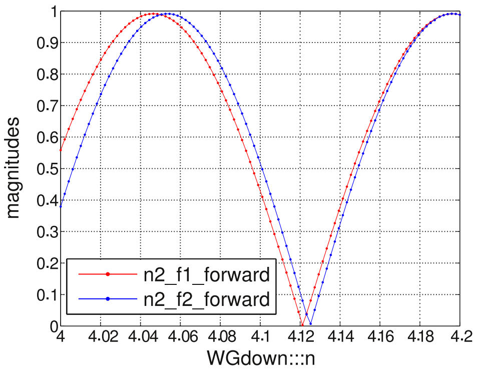


Fig. 9. Results from DC sweep on the Mach-Zehnder Modulator.

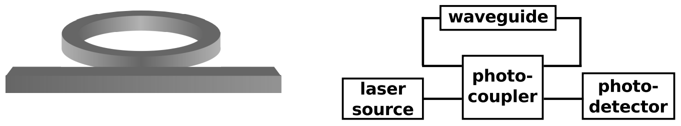


Fig. 10. A 3D illustration of a silicon micro-ring resonator (left) and its system diagram used in MAPP. The coupling region is modelled as a photo-coupler.

```
netlist = add_element(netlist, lightsourceModSpec(), 'LS',...
               {{'n1'}, {'fn1', {'f1'}}}, {});
netlist = add_element(netlist, waveguideModSpec(), 'WG1',...
               {{'n1', 'in1'}, {'fn1'}}, {{'L', 1e-4}});
netlist = add_element(netlist, waveguideModSpec(), 'WG2',...
               {{'out1', 'n2'}, {'fn1'}}, {{'L', 1e-4}});
netlist = add_element(netlist, photocouplerModSpec(),...
               'Coupler', {{'in1','in2','out1','out2'},...
               {'fn1'}}, {{'cc', 0.5}});
netlist = add_element(netlist, ...
               waveguideModSpec_effective_ng(), 'ring',...
               {{'out2', 'in2'}, {'fn1'}}, {{'L', 5e-4}});
netlist = add_element(netlist, lightsinkModSpec(), 'Sink',...
               {{'n2'}, {'fn1'}});
```

When we set the outgoing wave from the laser source to 1, the corresponding incoming wave arriving at the photo-detector is a complex number indicating the decay in magnitude and shift in phase of the light. We perform a DC sweep with frequency as the sweeping parameter, and plot the magnitude of Eout as in Fig. 11. From Fig. 11 designers can then observe the resonant frequencies of the modulator.

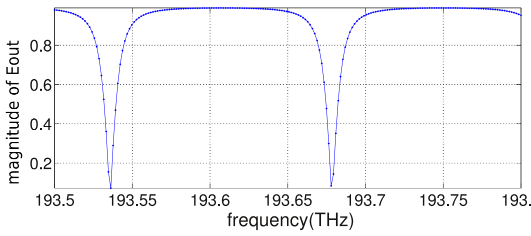


Fig. 11. DC sweep of laser frequency on the ring resonator in Fig. 10. The relative magnitude of the electric field of the light arriving at the photo-detector with respect to that emitted by the source is plotted.

References

[1] K. Petermann, Laser diode modulation and noise. Springer, 1991, vol. 3.

[2] T. Wang, K. Aadithya, B. Wu, J. Yao, and J. Roychowdhury, "MAPP: The Berkeley Model and Algorithm Prototyping Platform," Sep. 2015, pp. 461–464, DOI link.

[3] T. Wang, A. V. Karthik, B. Wu and J. Roychowdhury, “MAPP: A platform for prototyping algorithms and models quickly and easily,” in IEEE MTT-S International Conference on Numerical Electromagnetic and Multiphysics Modeling and Optimization (NEMO). IEEE, 2015, pp. 1–3.

[4] “MAPP: The Berkeley Model and Algorithm Prototyping Platform,” web site: https://github.com/jaijeet/MAPP.

[5] T. Wang and J. Roychowdhury, “Multiphysics Modelling and Simulation in Berkeley MAPP,” in Numerical Electromagnetic and Multiphysics Modeling and Optimization (NEMO), 2016 IEEE MTT-S International Conference on. IEEE, 2016, pp. 1–3.

[6] D. Amsallem and J. Roychowdhury, “ModSpec: An open, flexible specification framework for multi-domain device modelling,” in Computer-Aided Design (ICCAD), 2011 IEEE/ACM International Conference on. IEEE, 2011, pp. 367–374.

[7] E. Kononov, “Modeling photonic links in Verilog-A,” Ph.D. dissertation, Massachusetts Institute of Technology, 2013.