


Handbook
for
Generic Photonic IC Design

Editors: Meint Smit and Xaveer Leijtens

4-4-2026

 *Handbook for generic photonic IC design*, by the *Photonic Integration group*, Technische Universiteit Eindhoven, is licensed under a Creative Commons “Attribution-NonCommercial-NoDerivatives 4.0 International” license.

We traced the ownership of all figures used as far as we could. However, if you are a copyright owner and believe we used your work without permission, please contact us at coordinator@jeppix.eu.

Chapter 5

Simulation Methods

DOMINIC GALLAGHER, HANNES LÜDER, MEINT SMIT

This chapter describes the methods and tools that are available to the designer for calculating the properties of his or her components or circuits. In the first section a brief explanation is given of a number of numerical methods used by the software tools described in this chapter. Section 2 describes a significant number of computation examples using different software tools. The examples are taken from the chapters in this book. Section 3 provides short descriptions of the tools provided by the design software companies which are contributing to this chapter.

5.1 Numerical Methods

5.1.1 Introduction

Modern photonic integrated circuits (PIC) pose several challenges to the designer. Particularly if there are active components in the PIC, there are potentially many physical processes occurring in the chip, in addition to the electro-magnetics governing the propagation of light along PIC waveguides, including current flows in the presence of electrical fields (drift), electrical diffusion, thermal diffusion, carrier confinement in quantum wells, microwave-like propagation of high speed signals over the chip to name just a few. A simulation of an active PIC must take all these effects into account without requiring a super-computer to achieve reasonable accuracy. This section will introduce the reader to the methods that are available to achieve this and provide him or her with the information to decide which method is appropriate for a given problem, when to use a detailed or rigorous model and when a simple model will suffice.

There are many methods available to simulate PICs. Some of these are highly efficient but are capable of solving only a small subset of problems that photonics designers encounter. But even the methods capable of solving general problems are very effective at solving some problems and very inefficient at solving others. We introduce the reader to some of the most successful methods and explain the advantages and disadvantages of each. It does not attempt to present a complete list of available methods.

Dominic Gallagher is CEO of Photon Design Ltd., Hannes Lüder is with VPIphotonics.

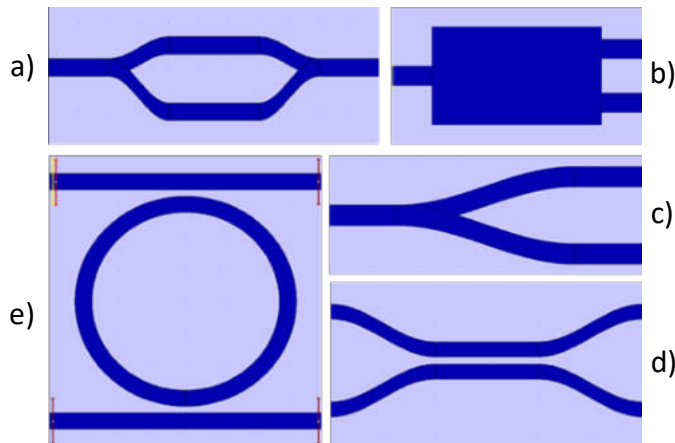


Figure 5.1: Waveguide based building blocks making up a modern PIC. Clockwise from top-left: a) Mach-Zehnder interferometer, b) multi-mode interference coupler (MMI), c) Y-junction, d) directional coupler, and e) ring-resonator.

A photonic circuit may consist of hundreds of smaller building blocks such as directional couplers, Y-junctions, bends, optical amplifiers and the like. The methods discussed in the following subsection (Section 5.1.2) can simulate these individual building blocks but are, however, not appropriate for modelling anything but the smallest, simplest PIC consisting of two or three building blocks. To model a PIC of greater complexity one will need an optical circuit simulator, as described in Section 5.1.3. An optical circuit simulator aims to be able to simulate a PIC that potentially extends to hundreds of components, such as an optical transceiver, or a LIDAR chip.

5.1.2 Electromagnetic modelling methods

The modelling of the electromagnetic propagation of light in the building block of a PIC is predominantly concerned with modelling the propagation along waveguides and waveguide components such as directional couplers, Y-junctions and the like (Fig. 5.1). This will have an important bearing on the choice of a simulation method. Some building blocks important for modern PICs do not however fall into this description. This includes the surface grating coupler (Fig. 5.2 left) and the AWG (Fig. 5.2 right)

Electromagnetic simulation methods can generally be categorised into:

Frequency domain methods

1. Frequency domain (FD) methods – only one frequency (wavelength) of optical field is assumed to be present in the simulated test – so that all fields have a simple $e^{j\omega t}$ time dependence. In this section we will discuss three FD methods: BPM (beam propagation method), FEFD (finite element frequency domain), EME (eigenmode expansion) method.

Time domain methods

2. Time domain (TD) methods – the evolution of the electromagnetic fields in time is modelled. Typically a short pulse is input into the device so that the response to a wide spectrum of wavelengths (frequencies) is obtained. In this chapter we will discuss two methods: FDTD (finite difference time domain) and FETD (finite element time domain).

There are many other methods for the solution of Maxwell's equations besides the ones listed above. Some of these will provide much higher performance in specific applica-

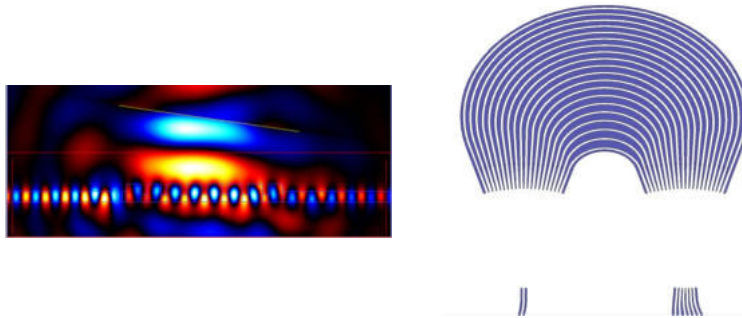


Figure 5.2: Complex building blocks found in modern PICs. Left: a surface grating coupler. Light entering along a waveguide from the left encounters a second-order grating which causes the light to be scattered up into an optical fibre. Right: an arrayed-waveguide grating (AWG), used to separate different wavelengths into different waveguides.

tions but the ones mentioned are more widely used because they can work well over a wide range of applications.

When choosing an algorithm for a particular application, you should consider the following:

- speed – obvious but crucial for efficient design work
- memory usage – Important for the simulation to fit in your computer
- numerical aperture – the range of angles that can be accurately propagated. Ideally the algorithm would be completely agnostic to angle. However for waveguide components light might be propagating over a narrow range of angles and so an algorithm with a small NA may still work well. *numerical aperture*
- Δn – the refractive index contrasts in the device. Ideally the algorithm would deal well with any contrast – InP to air is ~ 2.2 .
- polarisation – it should model all polarisations of light equally well.
- lossy materials – it should be able to model absorbing materials, even metals.
- reflections – can it deal with reflections in the device?
- non-linearity – it might need to model non-linear effects such as Kerr effect.
- dispersive materials – in general the refractive index is varying with wavelength.
- arbitrary geometries – some algorithms can model circular structures well, others rectangular. The ideal algorithm would model all geometries equally well.
- ABCs: a good algorithm should be capable of preventing light incident on the outside of the simulation box from being reflected. An absorbing boundary condition (ABC) is one way of achieving this. *absorbing boundary condition*
- PML: another way of preventing reflections from the walls of a simulation box is to add an absorbing layer just inside the walls of the box which will absorb the light travelling through it while minimizing the reflection of light from the inner surface of the absorbing layer. A perfectly matched layer (PML) allows introduction of absorption with negligible reflection from the inner surface. *perfectly matched layer*

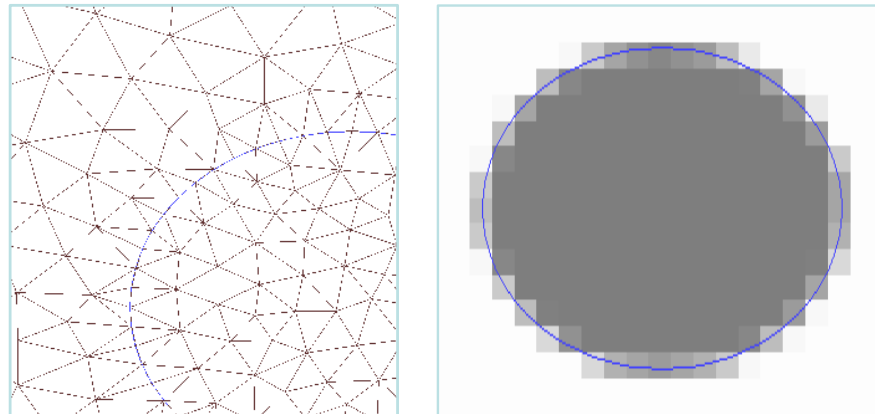


Figure 5.3: (left) A finite element mesh conforming to the boundary of the circle. (right) A rectangular mesh must create a staircase approximation of the circle.

5.1.2.1 Finite Element Frequency Domain Methods (FEFD)

Finite Element Frequency Domain FEFD methods [168] divide the simulation domain up into elements. Elements can in principle be of almost any shape. Typically they are of tetrahedral or cuboid form, but elements with curved faces are also popular. With each element the solution is represented by a set of basis functions. These could be simply polynomials. The larger the set of basis functions the higher the order of the finite elements and the more accurate the solution, at the cost of computation time.

Finite Element conformal mesh FE methods have the big advantage of using a conformal mesh – where the mesh (elements) follows the boundaries of the structure as shown in Fig. 5.3 for a circle.

Advantages and Disadvantages of FEFD. Aside from the conformal mesh mentioned above, FEFD methods do generally provide a very efficient, accurate approach for modelling small volume wide-angle problems. Wide-angle refers to light which is assumed to be propagating over all directions. The downside of FEFD is that it scales poorly with volume compared to other techniques. The technique reduces to inverting a linear matrix whose row size is equal to the number of unknown fields N_u in the whole volume. The best FEFD methods scale as approximately $N_u^{1.25}$, however even this 1.25 exponent means that for large volumes the computation time will become excessive, as will the memory usage. So great for wide-angle problems in smaller volumes; not suitable for larger volumes where it becomes very slow and memory intensive.

An example for the case of a T-junction is shown in Fig. 5.4.

5.1.2.2 Eigenmode Expansion Methods (EME)

Eigenmode Expansion Method The EME [169][170] method is very different to FEFD, FDTD and FETD methods presented elsewhere in this chapter. In particular it assumes that light is travelling largely along waveguides, or at least a narrow range of directions. On the one hand this means that it is not suitable for wide-angle problems. On the other hand it can take advantage of this assumption to dramatically increase the computational efficiency for suitable problems.

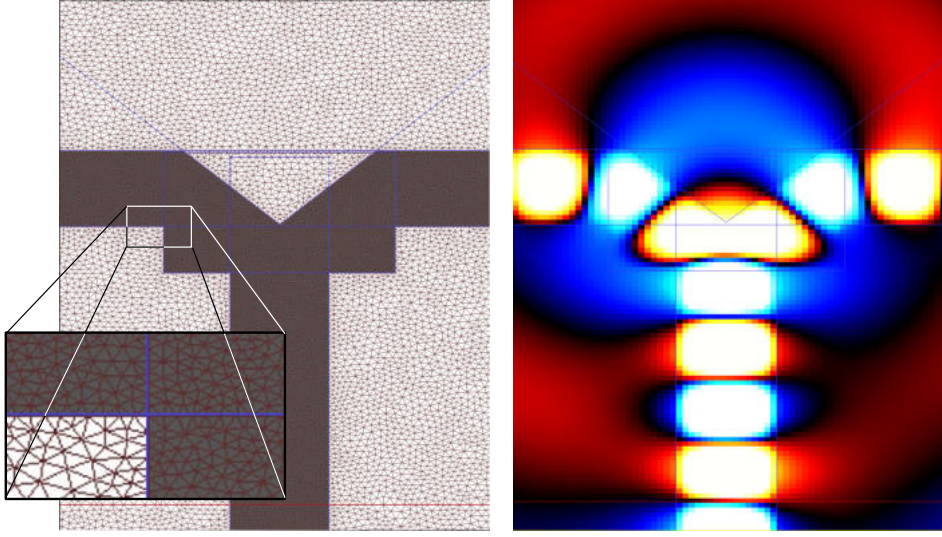


Figure 5.4: (left) Triangular mesh of a 2D-FEFD simulation of a waveguide T-junction. (right) The computed magnetic field. Note that the staircasing seen in the field image is just the plot resolution, not the resolution of the underlying calculation.

In a region of a photonic component where the refractive index is z -invariant (a waveguide), we can have solutions of Maxwell's Equations of the form $U(x, y)e^{-j\beta z}$. These are the modes of the waveguide and $U(x, y)$ is the mode profile. These modes are not the only solutions. The general solution can be expressed:

$$\begin{aligned} E(x, y, z) &= \sum_m E_m(x, y) \cdot \{c_m^+ e^{j\beta_m z} + c_m^- e^{-j\beta_m z}\} \\ H(x, y, z) &= \sum_m H_m(x, y) \cdot \{c_m^+ e^{j\beta_m z} - c_m^- e^{-j\beta_m z}\} \end{aligned} \quad (5.1)$$

i.e. as a superposition of forward and backward travelling modes. E_m and H_m are the fields of the waveguide modes and c_m^+ , c_m^- are the amplitudes of the modes. The math gets more complicated when the cross-section varies, for example in a tapered waveguide or where there is an abrupt change in cross-section, for example where a PIC is butt coupled to an optical fibre. However the essence of the method is that it is able to remove quick variations along the z -axis, permitting a grid spacing far in excess of what FE and FD methods can achieve.

Advantages and disadvantages of EME. Light can be propagated along a straight waveguide or a waveguide of constant curvature in one step. When the cross-section is varying slowly, EME can still take very long steps. Thus it is ideal for the waveguide-like devices typically found in a PIC – directional couplers, Y-junctions, bends, MMIs... However, it works well only when it needs to consider a relatively small number of modes in the cross-section. As the cross-section increases, for example in an array of many coupled waveguides, the number of modes N_m also needs to increase and the compute time increases as N_m^3 . This cubic exponent is very severe and quickly prevents EME from being used in highly multi-mode applications.

One huge advantage of EME over FD and FE methods is that it is a scattering matrix *scattering matrix* method:

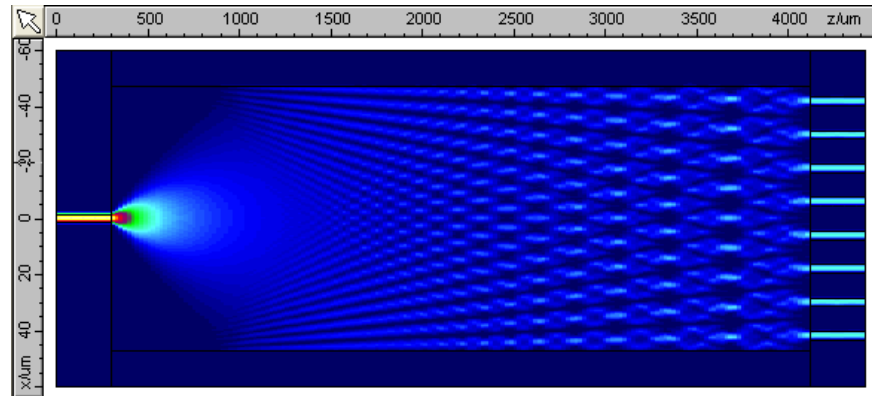


Figure 5.5: EME simulation of a 1-to-8 MMI coupler. EME needs to compute the modes of just 3 cross-sections, making MMIs an ideal application for the technique, even in 3D this simulation takes less than 10s on a modest 4-core PC.

$$U_{\text{out}} = S \cdot U_{\text{in}} \quad (5.2)$$

where S is the scattering matrix of the component, U_{in} and U_{out} are vectors of the mode amplitudes incident on and exiting the component. FD and FE methods all compute a response to one given input. S -matrix methods like EME in contrast solve the problem for an arbitrary input (arbitrary U_{in}). This means that EME can compute the response of a component once and re-use the response for every instance of that component in a large PIC.

An overview of the most important properties of the EME is given in Table 5.1

5.1.2.3 The Beam Propagation Method (BPM)

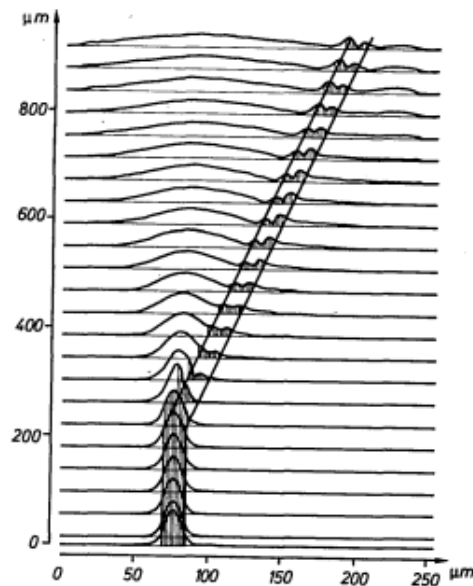
Beam Propagation Method The BPM method is one of the oldest methods used for modelling photonics. It is a frequency domain method and is an approximate solution of Maxwell's equations that is aimed at waveguide-like applications, i.e. where the light is travelling predominantly in one direction. There are two dominant variants – the FD-BPM (Finite Difference BPM) [171] and the FT-BPM (Fourier Transform BPM). The key idea of BPM is to remove the fast varying term $\exp(-j\beta z)$ from the fields, leaving hopefully a function that varies slowly with z . This allows BPM to take much larger z -steps than e.g. FDTD.

Advantages and disadvantages of BPM. The simulation time of FD-BPM scales linearly with cross-section area which is almost as good as it gets. Its long z -step also speeds up the algorithm. It is therefore a very fast algorithm. It is also much more memory efficient than, e.g. FDTD. The big limitation of BPM is that it is an approximate solution of Maxwell's Equations so that it does not model high index contrast problems very accurately. It is thus great for doped silica-like waveguides but not for e.g. silicon photonics.

Table 5.2 summarises different aspects of its performance, with scores out of 10 for each aspect. Speed and memory performance are not given scores since these depend too much on the application – BPM might be fastest for one application and FDTD for another.

Table 5.1: EME Summary Table

Aspect	Performance	Score/10
Speed	EME scales poorly with cross-section area – as A^3 (A is cross-section area). However it can efficiently model very long structures especially if their cross-section changes only slowly or occasionally. Periodic structures scale as $\log(\text{number of periods})$ – so can compute efficiently. S-matrix approach allows a set of similar simulations to be done very quickly – parts of previous calculations can be reused.	-
Memory	Memory increases at rate between A^2 and A^3 (A is cross-section area), but very efficient for long or periodic devices.	-
NA	Can model wide-angle beams by increasing the number of modes in the basis set at expense of speed and memory.	7
Delta-n	Rigorous Maxwell Solver can accurately model high delta-n,.	8
Polarisation	Rigorous Maxwell Solver is polarisation agnostic.	10
Lossy materials	Depends on mode solver used.	7
Reflections	Yes – easy and stable even when there are many reflecting interfaces.	10
Non-linearity	Difficult – have to iterate, and then only modest non-linearity levels will converge.	3
Dispersive	Being a frequency-domain algorithm this is easy.	10
Geometries	Depends on the mode solver used. Can use different structure discretisations in different cross-sections, so solver can better adapt to the geometry.	7
ABCs	Depends on the mode solver used. E.g., a finite-difference solver can be readily constructed to implement PMLs. However, PMLs are more difficult to use with EME than with BPM or FDTD.	7



BPM simulation of a bent waveguide. Credit: Roey et al. [172].

Table 5.2: A score-chart setting out the strengths and weaknesses of FD-BPM.

Aspect	Performance	Score/10
Speed	FD-BPM scales linearly with area and can take fairly long steps in propagation direction.	-
Memory	Usage scales linearly with c/s area.	-
NA	Best with low NA simulations. Versions using Pade approximants can model a beam travelling at a large angle but still cannot deal well with light simultaneously travelling at a wide range of angles.	4
Delta-n	Best with low delta-n simulations.	5
Polarisation	Semi-vectorial versions work best. Still problems modelling mixed or rotating polarisation structures accurately.	5
Lossy materials	Can model modest losses efficiently. Most versions cannot deal well with metals.	7
Reflections	Some success in implementing reflecting/bi-directional BPM but generally avoided due to slow speed and stability problems.	3
Non-linearity	FD-BPM can model non-linearity.	9
Dispersive	Being a frequency-domain algorithm this is easy.	10
Geometries	The BPM grid allows diffuse structures to be modelled easily. Problems modelling non-rectangular structures accurately on the rectangular grid.	7
ABCs	PMLs available and work well.	9

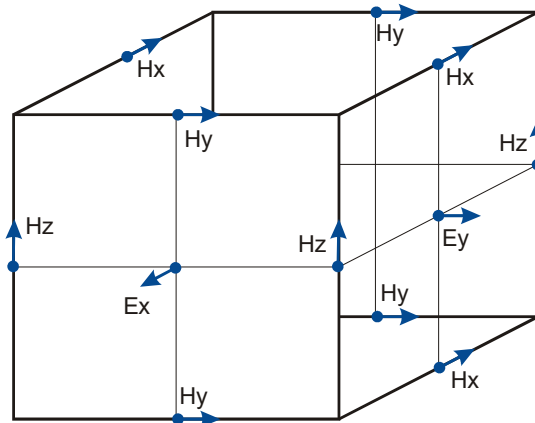


Figure 5.6: The Yee cell of FDTD, showing the position of the 6 EM fields on the cell surface. This staggered grid makes the algorithm more accurate.

5.1.2.4 The Finite Difference Time Domain Method (FDTD)

The FDTD method [173] is now one of the most widely used simulation methods in electromagnetics. It is a brute force finite-difference discretisation of Maxwell's Equations in time and space. It is in principle capable of solving almost every electromagnetic problem. In practice its brute force approach means that it is not practical for simulation of large volumes. It is however very simple to implement – the basic algorithm can be written in 30 lines of code.

*Finite Difference
Time Domain
Method*

The dominant FDTD algorithm dates back to Kane Yee in 1966 [174] and employs the cuboid unit cell indicated in Fig. 5.6 where the 6 EM fields each occupy a different location on the cell. This allows the curl equations of Maxwell's Equations to be readily evaluated. For example the front face shown in the figure gives us (L,R,B,T denoting left, right, bottom, top edge of the face):

$$\frac{\partial D}{\partial t} = \nabla \times H \quad (5.3)$$

$$\frac{\partial D}{\partial t} \simeq \mu_0(H_{z,L} - H_{z,R} + H_{y,B} - H_{y,T}) \quad (5.4)$$

The Yee cell based FDTD algorithm looks like it should be first-order convergent (i.e. the error should go down proportionately to the cell size. In fact in a uniform medium the first order errors cancel out so making it 2nd order convergent. It becomes 1st order convergent again when a surface crosses the unit cell.

Variants with e.g. triangular grids [175] have appeared more recently though these have not proved advantageous – probably because they are 1st order convergent even in a uniform medium.

Advantages and Disadvantages of FDTD. The FDTD algorithm is a rigorous time domain solution of Maxwell's Equations, so can in theory do everything. If you need a response over a range of wavelengths then a time domain method like FDTD is advantageous. It can also relatively easily be extended to model non linear and dispersive materials as well as possessing good ABCs. In fact it does well on all the considerations presented above apart from speed and memory.

*absorbing
boundary
conditions*

Table 5.3: FDTD Score Chart.

Aspect	Performance	Score/10
Speed	Scales as V (device volume) but grid size is small so not as good as BPM or EME for long devices.	-
Memory	Scales as V (device volume) but grid size is small so not as good as BPM or EME for long devices.	-
NA	Omni-directional algorithm is agnostic to direction of light – great when light is travelling in all directions	10
Delta-n	Rigorous Maxwell solver, happy with high delta-n, but slows down somewhat with high index.	9
Polarisation	Rigorous Maxwell Solver is polarisation agnostic	10
Lossy materials	Can model even metals accurately with a fine enough grid and small modifications to the algorithm.	
Reflections	Yes – easy and stable even when there are many reflecting interfaces.	10
Non-linearity	Yes – non-linear algorithm relatively easy to do	9
Dispersive	Have to approximate the dispersion spectrum with one or more Lorentzians but exact fit to the spectrum over a wide wavelength is difficult and the algorithm also slows down.	7
Geometries	Fine rectangular grid can do arbitrary geometries easily, though there are problems approximating diagonal metal surfaces	8
ABCs	Yes – very effective and easy to use	9

The computation time of an FDTD simulation is proportional to the number of grid points times the number of time steps taken. This linear scaling with number of grid points makes FDTD better than many other methods for devices of large volume. In practise it cannot keep up with methods such as EME which can take advantage of features often encountered in photonics such as the long waveguides. Note also that as the volume of a problem increases then in general it will need more time-steps for the fields to propagate around. So, FDTD simulation time scales typically as simulation-volume \times max-dimension. If you need to model a high-Q cavity then the light might need time to travel a distance many times max-dimension in order for the simulation to achieve the spectral resolution needed.

FDTD requires a dense simulation grid of at least 12 Yee cells per wavelength, preferably 15-20. This makes it relatively memory hungry.

ring resonator **Example 1 - Ring resonator** The ring resonator, illustrated in Figure 5.7, is a good application for FDTD, at least for small rings. The ring has discrete resonant frequencies corresponding to integer numbers of wavelengths around the ring. At these resonances light is coupled from the top waveguide to the bottom one. In this example a pulse of light has been injected top left and resonant wavelengths are quasi-trapped in

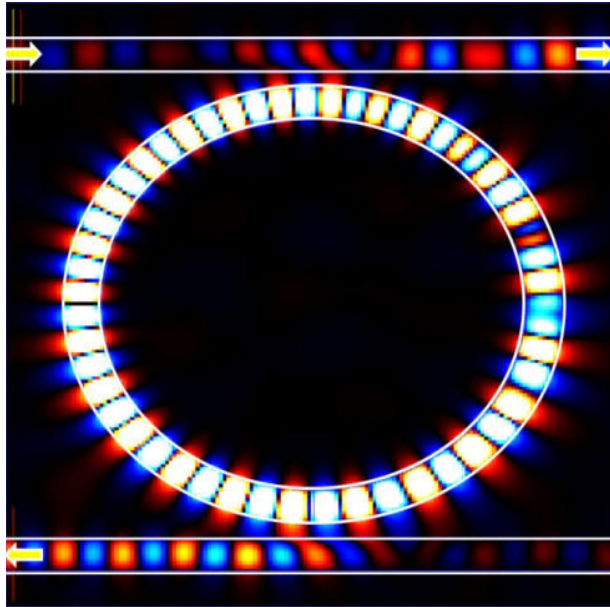


Figure 5.7: A ring resonator simulated using FDTD.

the ring, gradually coupling out to the bottom waveguide. This example represents a silicon-on-silica structure. The mode in the waveguide has an effective index of ~ 2.5 and we used a Yee cell size of $\lambda/30$ – which translates to ~ 12 cells per wavelength in the medium. Such resolution is good for only a quick initial simulation and generally something closer to 25 cells per medium-wavelength would be recommended depending on what accuracy you need.

Example 2 - Surface grating coupler The surface grating coupler, illustrated in Figure 5.8, is used to couple light into or out of a PIC from or to an optical fibre approaching the chip from above. This component is useful for providing optical access to any part of a PIC, rather than just the chip edge where the waveguides terminate. A waveguide injects light from the left which gets scattered upwards by a second-order diffraction grating. The curved grating lines ensure that the light focusses correctly into the mode of an optical fibre (yellow) placed above the grating.

*surface grating
coupler*

The surface grating coupler is an ideal application for FDTD since it combines

- Light travelling in almost all directions – ruling out BPM and EME
- A relatively modest volume – typically $20\ \mu\text{m} \times 10\ \mu\text{m}$ by 1 or $2\ \mu\text{m}$.
- High index contrast.

An overview of the most important properties of the FDTD is given in Table 5.3.

5.1.2.5 The Finite Element Time Domain Method (FETD)

FETD

The Finite Element Time Domain Method is a relatively recent development. Finite element methods for solving Maxwell's equations have been around for a long time

*Finite Element
Time Domain
Method*

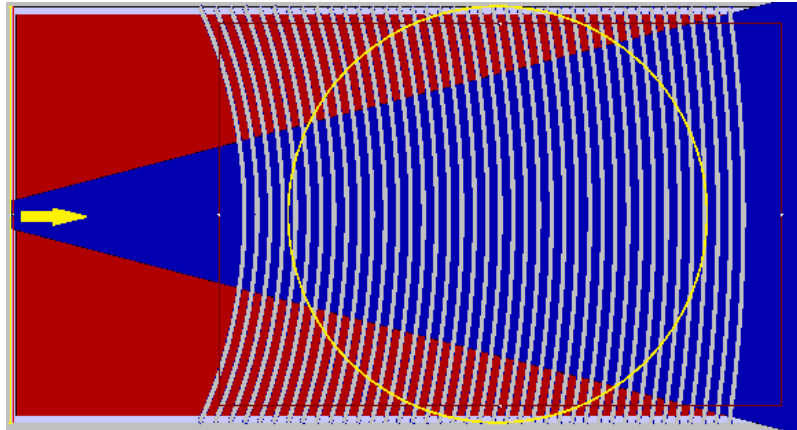


Figure 5.8: An FDTD simulation showing the light travelling along a surface grating coupler. The yellow line denotes the position of a fibre facet that the coupler is designed to excite.

but even the time-domain versions of these had poor scaling as problem size increased. The adoption of Galerkin methods to Maxwell's equations [176] allowed FETD to achieve the same scaling as FDTD – i.e. the compute time is proportional to the number of field nodes times the number of time steps taken.

Advantages and Disadvantages of FETD. FETD and FDTD have similar scaling as a function of volume and time steps:

$$\text{compute time} \propto (\text{number of elements}) * (\text{number of time steps})$$

However, whereas FDTD accuracy ($1/\text{error}$) scales with $(\text{grid spacing})^{-2}$ (in uniform space) or $(\text{grid spacing})^{-1}$ near structural discontinuities, FETD finite elements can be of high order p with accuracy scaling as $(\text{element size})^{-p}$. Element order p can in theory be of arbitrary size but would typically be from 3 to 5. Although higher order elements take more compute time to evaluate each element, the elements can be much bigger so that FETD can converge to high accuracy much more quickly than FDTD. This essentially means that FDTD tends to be faster if you need only modest accuracy but FETD always wins if you demand higher accuracy. This is illustrated in Fig. 5.10.

5.1.2.6 Time Domain vs Frequency Domain Methods

Time-domain and frequency-domain methods each have their own pros and cons. Even if we limit ourselves to exploring just linear solutions of Maxwell's Equations, to compare the two categories we need to ask ourselves a few questions:

1. Do we want one wavelength, a few or many (a spectrum)?
2. Is the structure highly resonant? I.e. does it have sharp spikes in its response spectrum?
3. Is the light predominantly propagating close to one axis?
4. Does the device have a large range of refractive indices?

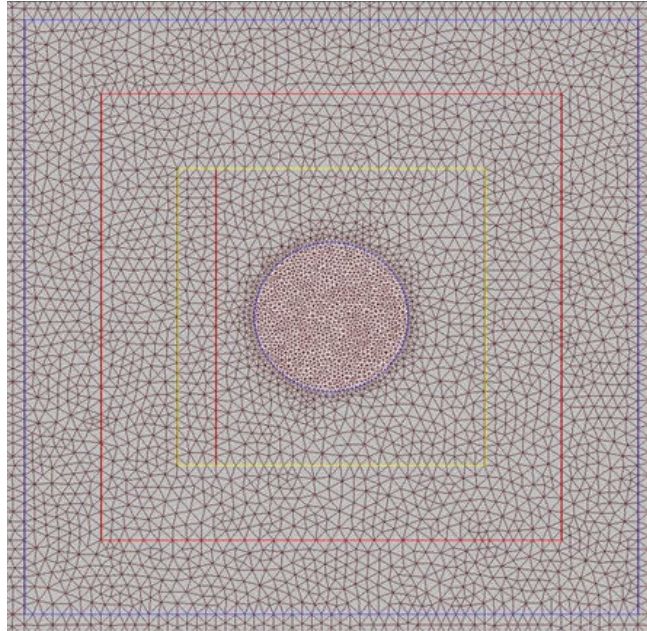


Figure 5.9: A 2D-FETD mesh of a gold disk. The elements a) conform to the boundary of the disk and b) adapt in size according to the refractive index.

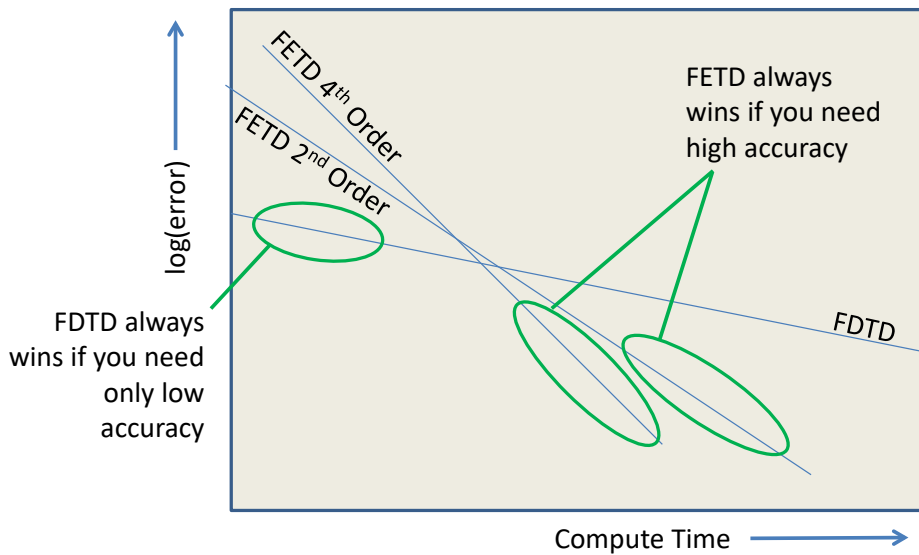


Figure 5.10: Relative accuracy convergence of FETD and FDTD.

Here is the formula for FDTD:

$$\begin{aligned} T_{sim} &= \text{const} \cdot N_x \cdot N_y \cdot N_z \cdot N_t \\ &= \text{const}' \cdot V / (\delta s)^4 / \delta f, \end{aligned} \quad (5.5)$$

where N_x, N_y, N_z are the number of Yee cells in each axis, V is the device volume, δs is the cell size and δf is the frequency resolution required. The smaller δf , the more time-steps are needed. Notice the δs^4 – the 4th power is due to the fact that the time step must obey $\delta t < \delta s / v_g$ where v_g is the highest group velocity in the simulated device. This δs^4 is problematic for FDTD – if we need to halve the grid size to attain the required accuracy then the simulation is now $16\times$ slower. However FDTD gets us a whole spectrum, not just one wavelength.

For EME, using a finite difference mode solver we have:

$$T_{sim} = (c_1 \cdot N_m \cdot N_z \cdot (N_x \cdot N_y)^{1.3} + c_2 \cdot N_m^2 \cdot N_x \cdot N_y) \cdot N_\lambda \quad (5.6)$$

where N_λ is the number of wavelengths needed, N_x, N_y are the number of grid cells needed for the finite difference mode solver, N_z is the number of unique cross-sections and N_m is the number of modes needed per cross-section. Note that N_z is in general much less than the N_z of the FDTD case – EME can make long steps even covering multiple wavelengths. On the other hand note the $(N_x \cdot N_y)^{1.3}$ in EME – this means that EME is not good for things with a large cross-section.

Machine Scaling

In choosing an algorithm for simulating a photonic device it is helpful to understand how well the algorithm can make use of the computational capabilities of modern computers. The relevant technologies are:

Multi-core CPUs: all modern CPUs now contain multiple computational cores, each of which can run independently of the others and is almost like an independent computer – except that it shares the same memory space as other cores. A particular code has to be re-written to take advantage of a multi-core CPU – a process usually referred to as CPU parallelisation. Some simulation codes parallelise better than others.

Vectorisation: an individual core of a modern CPU can execute instructions which can process a vector of floating point numbers at a time. Intel and AMD CPUs support the AVX/AVX2/AVX512 vector instructions which can operate on 4/8/16 single-precision floating point numbers at a time. In practice few codes can speed up by these numbers – in the author's experience AVX2 can get to a factor 5-6 (FETD case).

Clustering: this denotes the connection of many computers together, typically using Ethernet connections, so that they can all work together on a problem. The release of MPI (Message Passing Interface) in 1994 greatly facilitated the support of clusters. Sending data across an Ethernet connection causes substantial delays compared to the equivalent in GPUs and multi-core CPUs. This limits the utility of clustering.

GPUs: GPUs (Graphics Processing Units) are a development of vector instructions. A good GPU can execute thousands of floating point operations/second though each is rather slower than modern CPU vector instructions. GPUs are increasing in popularity due to their raw floating point potential. A mid-range CPU can reach 1 TFlops (single precision, Xeon W5-3435X). A mid range GPU can in theory reach 50 TFlops (single precision, Gforce RTX 4080). The reader should however beware of some authors' reports on obtaining speedup factors of hundreds for their FDTD GPU codes vs CPU codes –

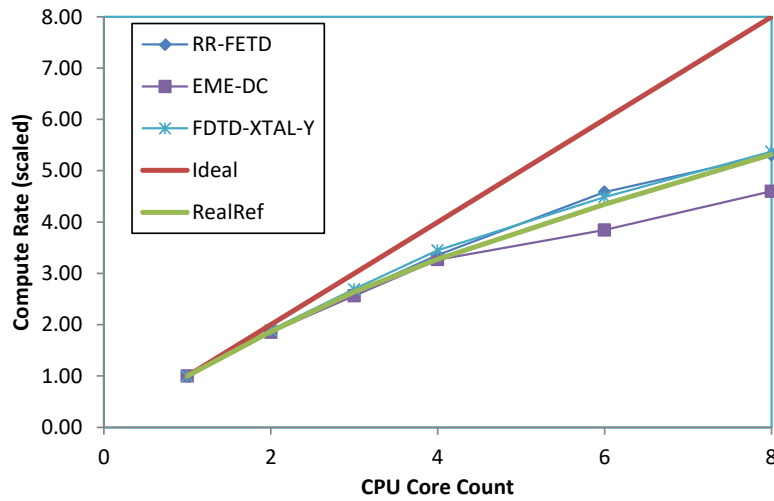


Figure 5.11: Scaling of FDTD, FETD and EME vs core count (Ryzen7-3700x – an 8 core CPU).

some of these appear to be comparing state-of-the-art GPU codes to old CPU codes that for example do not use CPU vectoring.

Some algorithms are more amenable to parallelisation than others. FDTD is particularly suited to parallelisation since a small calculation (update of a Yee cell) is repeated 1000s or even millions of times. FETD is also readily parallelisable – again each element can be updated independently of others. FEFD does not parallelise so well – although it has a large number of elements like FETD, it creates a large matrix which it must invert. The matrix is very sparse so the operation can be sped up but compute time still scales as something like $N_{dof}^{1.25}$ where N_{dof} is the number of degrees of freedom in the simulation – the number of finite elements (tetrahedra) multiplied by the number of field values in each element. This 1.25 exponent is the Achilles heel in the FEFD algorithm – the compute time grows super-linearly with simulation size, making it impractical for large problems.

EME can be parallelised but not as readily as FDTD. A directional coupler might be able to split into 8 parts. A larger circuit might be able to be split into 20-40 parts but rarely the 1000s that FDTD can do. But bear in mind that an FDTD could never tackle such a large circuit on an office workstation.

Fig. 5.11 below shows the scaling of FDTD, FETD and EME with increasing CPU core count. In an ideal world 8 cores would be 8x faster than one core (red line). FDTD and FETD both get $\sim 5\times$ speed up with 8 cores whereas EME only gets $\sim 4\times$. Note however that a modern CPU will throttle if many cores are used. This is indicated by the *RealRef* line and both FDTD and FETD get close to this practical limit.

5.1.3 Modelling of Photonic Integrated Circuits

Simulators using the methods outlined in the previous chapter are generally only able to deal with relatively small photonic components – perhaps a directional coupler or a surface grating coupler. When tasked with simulating a whole photonic integrated circuit (PIC), a different approach is needed. This is the role of the circuit simulator. The PIC is divided into smaller building blocks, each of which is simulated separately

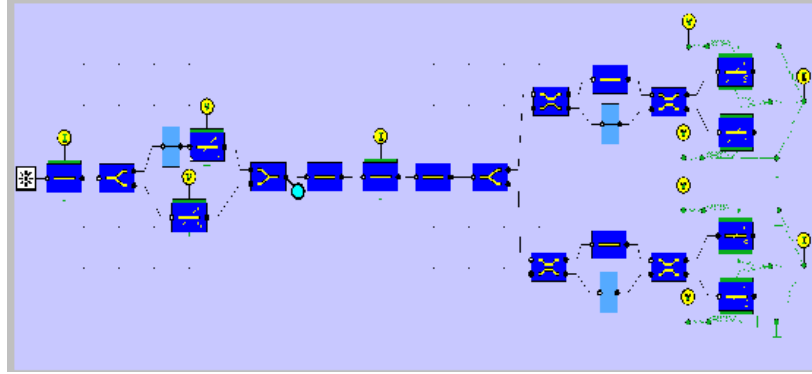


Figure 5.12: Layout of an optical transceiver PIC. Each building block of the PIC is analysed using a full electromagnetic solver, a surrogate model of each building block is then generated and fed into the circuit simulator.

surrogate model using one of the methods outlined above. A surrogate model of each building block is then generated. The surrogate model is a much simplified model that matches the behaviour computed by the full electromagnetic simulator but runs orders of magnitude more quickly. The circuit simulator then combines the individual surrogate models into a simulation of the whole PIC.

Travelling Wave Time Domain method As for electromagnetic solvers, circuits solvers can be divided into frequency domain and time domain solvers. We will present here one of the most powerful and elegant time-domain methods – the Travelling Wave Time Domain method (TWTd) invented by Carroll and Wong [177].

Consider a waveguide with light travelling in the fundamental mode, with forward and backward amplitudes $a(z), b(z)$. In the CW condition they have a time and spatial dependence $\exp(j\omega t \pm j\beta z)$, where β is the propagation constant of the mode. If the envelopes $A(z), B(z)$ of $a(z), b(z)$ vary slowly in time and space, we can apply the slowly varying envelope approximation:

$$A(z) = \frac{a(z)}{\exp(j\omega t + j\beta z)}, B(z) = \frac{b(z)}{\exp(j\omega t - j\beta z)} \quad (5.7)$$

i.e. $A(z), B(z)$ vary slowly with z and t . The TWTd method then uses the advection equations for $A(z), B(z)$. Adding terms for distributed feedback due to a DFB grating with coupling coefficient k ; and optical gain g , we have:

$$\begin{aligned} \frac{1}{v_g} \frac{\partial A}{\partial t} + \frac{\partial A}{\partial z} &= j\kappa B + (g - j\delta)A + F_A(N_e) \\ \frac{1}{v_g} \frac{\partial B}{\partial t} - \frac{\partial B}{\partial z} &= j\kappa A + (g - j\delta)B + F_B(N_e) \end{aligned} \quad (5.8)$$

grating feedback gain detuning

These equations can be solved in the time domain, rather like FDTD except now we only have a one-dimensional system and we can take large steps in both time and space since $A(z, t)$ is varying much more slowly than $a(z, t)$. Each waveguide of the PIC

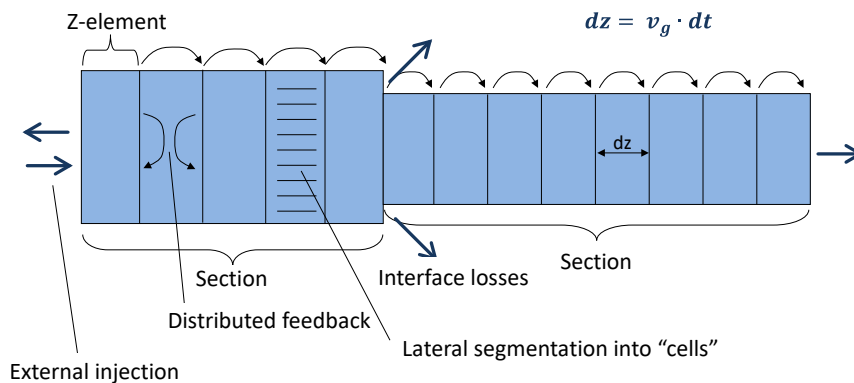


Figure 5.13: Propagation of light along a PIC using the TWTD method. Photons propagate one step forward or backwards each time step.

is divided into short z -elements and at each time step, photons travel one z -element up or down the waveguide.

5.1.3.1 Disadvantages of the TWTD method

Spectral artefacts. The TWTD method has a time step δt which equates to a free spectral range $\Delta f = 1/\delta t$ – any optical frequencies outside the range $f_0 \pm \Delta f/2$ are aliased back into the range. Also the amplitude and gradients of all optical signals must be continuous around the boundaries $\pm(f/2)$ of the free spectral range f , giving amplitudes $a(-\Delta f/2) = a(+\Delta f/2)$ etc. This means that artefacts will be introduced into the spectral response of any surrogate model imported into the circuit simulator. See Fig. 5.14.

Figure 5.15 shows an example of a DQPSK receiver. Dual polarisation Quadrature Phase Shift Keying is a technology to increase the data capacity of an optical link by a) switching the phase of the signal rather than the amplitude, b) transmitting light in two polarisations. The receiver consists of Mach Zehnder interferometers and balanced photodiodes. Figure 5.16 shows the constellation diagram computed for the link shown in Fig. 5.15.

Group velocity. The advection equations (Eq. 5.8) imply the relationship between the time-step t and the z -step z as $z/t = v_g$. So z is fixed once t is chosen. If you have a circuit element of short length say $10 \mu\text{m}$, and z is $4 \mu\text{m}$ then you must divide the element into 2 or 3 z -elements, giving rise to an error in the group velocity. To reduce the error one must choose a shorter t which has a speed penalty. The simulation time also increases with $(1/\delta t)^2$ implying that a single short segment in the circuit will dramatically increase the simulation time. There are tricks that the best simulators play to partially solve this problem but frequency domain circuit simulators do not have these problems in the first place.

5.1.3.2 Advantages of TWTD method

Despite the disadvantages listed above, the TWTD method is nevertheless very beneficial for three main reasons:

1. The time-domain approach provides a response over a spectrum of wavelengths in one run.

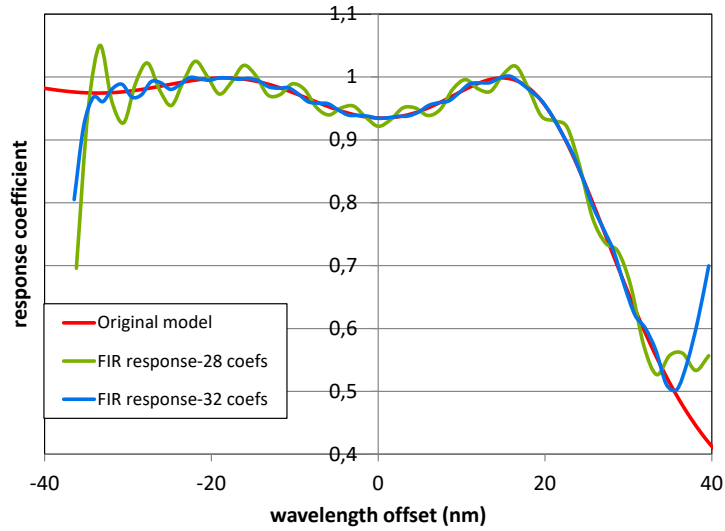


Figure 5.14: Generating a digital filter response (green, blue) to a reference response spectrum of a surrogate model (red). This example uses an FIR filter (finite impulse response), with 28 (green) and 32 (blue) coefficients. The response filter must be periodic hence the artefacts at the sides of the free spectral range -40 to +40 nm. With longer filters, these artefacts become confined to a relatively narrow region close to the edges of the FSR. Frequency domain based circuit simulators do not have this problem.

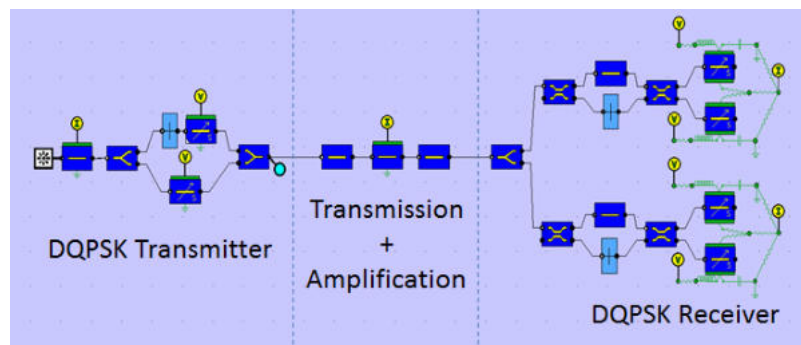


Figure 5.15: DQPSK digital link simulated in the Photon Design's TDTW simulator PICWave. (symbolic circuit design).

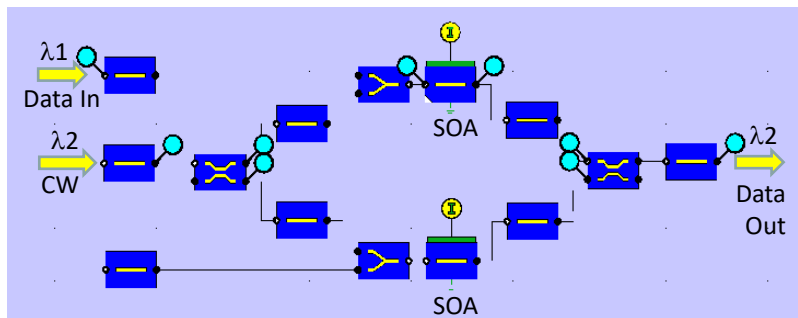


Figure 5.16: An optical regenerator using SOAs to both amplify the signal data and change its wavelength.

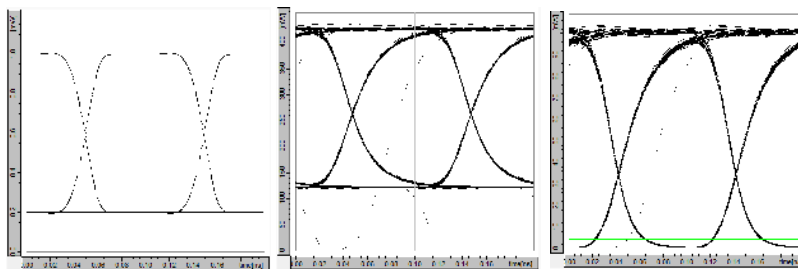


Figure 5.17: Eye-diagrams of (left) the input data at λ_1 , (middle) the total output power, (right) the output signal at λ_2 .

2. The time-domain approach lends itself readily to combination with rate equations describing electron-hole populations in a semiconductor such as a diode laser, allowing it to model the dynamics of an active PIC.
3. The time-domain approach lends itself readily to the modelling of nonlinearities.

5.1.3.3 Example – an optical regenerator

Figure 5.16 shows a schematic of an optical regenerator. The regenerator takes a data signal at wavelength λ_1 and a) amplifies it, b) improves the on/off ratio of the signal, and c) changes the data wavelength to λ_2 . When there is no data signal the λ_2 input is split equally between the two SOAs and when it recombines in the final directional coupler the two SOA outputs cancel each other out, causing no output at the output port. When power is injected into the Data port, the top SOA sees more optical power and so the carrier density decreases, giving rise to a change in the refractive index of the waveguide core. This in turn means that we no longer have phase cancellation in the output and the λ_2 CW signal can now pass through. The operation is illustrated in Figure 5.17.

5.1.4 New Component/Circuit Approach

A disadvantage of the circuit simulator approach described above is that it requires a 3-stage design flow:

1. Compute surrogate models for each building block

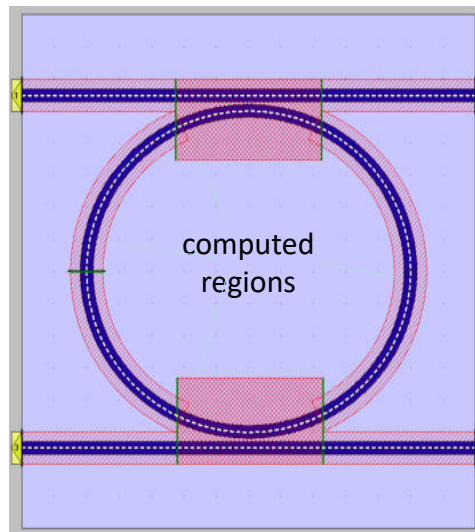


Figure 5.18: Fused component/circuit simulator. Parts of the chip where light may be present are marked as shown in red. Only the red regions are fed into the EM solvers, allowing the fused simulator to model even large circuits. Combining this with an EME algorithm which will compute S-matrices allows repeated parts of the circuit to be simulated just once.

2. Combine the building blocks into a circuit and run a circuit-simulator
3. Export the circuit to a layout tool to generate fab masks

In fact there may be several iterations between these stages – for example the layout tool may indicate that two building blocks are too close together and the circuit simulator re-run.

The author's company introduced a new approach which allows all 3 stages to be combined into one. In order to keep the simulation time manageable, the circuit is annotated indicating which parts need to be computed by the electromagnetic solver and which parts may be ignored. This is illustrated in Figure 5.18 for a ring resonator.

Figure 5.19 below shows a PIC of an 8:1 WDM component displayed in a more traditional circuit simulator. Figure 5.21 shows the same circuit again in the unified environment. This one environment can simulate the individual components and the whole circuit and generate the full layout files (GDS-II) all from this same environment. The unified environment also allows multiple other effects to be modelled including

- Optical crosstalk – light leaking out of one waveguide bend and coupling back into a remote bend (see Figure 5.20 - inset).
- Thermal cross-talk – heat generated by e.g. one laser diode from one channel may perturb the waveguides of a neighbouring channel by thermal diffusion.
- Chip stress fields.

The Unified Environment, being based on S-matrices, allows the simulator to take advantage of any repetition in the circuit. In Figure 5.20 there is just one unique straight waveguide, one unique bend, one unique 1×2 MMI and one unique 2×2 MMI, permitting this circuit to be simulated in just a few seconds in 2D and a couple of minutes in 3D – even as a fully vectorial simulation and taking any reflections into account.

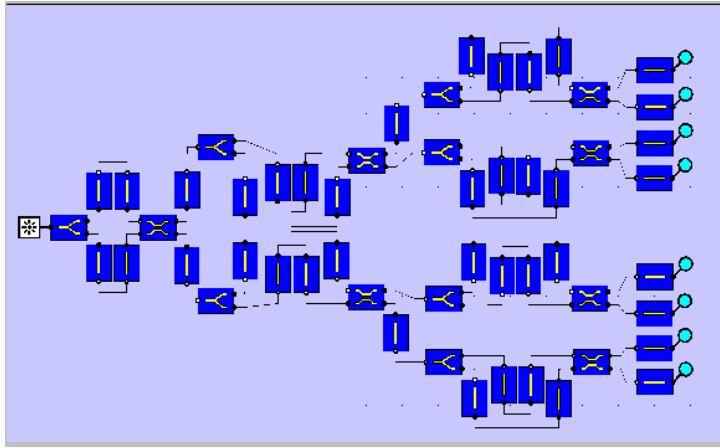


Figure 5.19: A 4:1 wavelength division multiplexer constructed using cascaded MZIs. The first one splits the 8 channels into even and odd channels, the second level of two MZIs then split the 8 channels into 4 pairs (1,5)-(2,6)-(3,7)-(4,8). A final level of 4 MZIs then separates the 4 pairs.

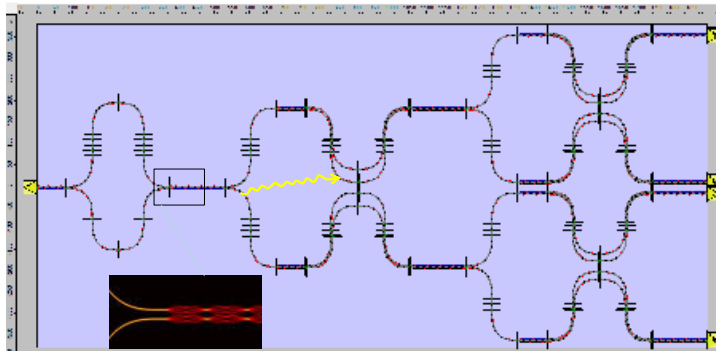


Figure 5.20: Unified Component/Circuit Editor (MT-FIMMPROP) showing the same circuit presented in Figure 5.19. The geometric presentation more readily permits the modelling of stray light as depicted by the yellow arrow – in this case radiation from one bend coupling back into the circuit at another bend

5.2 Simulation Tools

5.2.1 VPIphotonics Design Automation

Photonic Design Automation PDA **VPIphotonics** first coined the term Photonic Design Automation (PDA) in 1998 to describe the design methodologies, software tools, and services used to engineer complex photonic networks and products. PDA helps optical component and system vendors manage intellectual property and reduce operating expenses by streamlining processes, from R&D to technical sales and marketing.

Today, VPIphotonics offers a flexible software ecosystem to support requirements in the design of optical components, systems, and networks. VPIphotonics' simulation solutions span from photonic integrated circuits and devices to optical transmission systems and optical network link engineering, enabling advancements in optical communications, medical sensing, and various defense applications.

VPIphotonics users are leveraging these solutions across a broad range of innovative application areas for cutting-edge technologies, including short- to long-haul transmission systems, high-capacity terrestrial and satellite communications, 5G & 6G networks, data center interconnects, sensing and analytics, optical coherence tomography, quantum communication, autonomous driving, high-power lasers and amplifiers, highly integrated programmable photonic circuits.

VPIphotonics ensures interoperability of its solutions with users' in-house developments and the tools of their suppliers and customers, third-party software (for example, for electronic design automation and circuit layout design), laboratory equipment, and existing databases.

5.2.1.1 Photonic Integrated Circuits Design

As part of the ecosystem, VPIphotonics offers a convenient, accurate, and fast design flow for developing modern and next-generation photonic integrated circuits (PICs). Starting from the design of the individual photonic devices with **VPIdeviceDesigner**, one can extend the scope to simulate large-scale PICs in **VPIcomponentMaker Photonic Circuits** and ultimately test a designed circuit or component within the entire application scope using **VPItransmissionMaker Optical Systems**.

finite-difference mode solver beam propagation method BPM eigenmode expansion method EME process design kit PDK **VPIdeviceDesigner** addresses the problem of optimizing the properties of integrated waveguides and waveguide-based devices (splitters, couplers, photonics lanterns, and others). It offers a set of full-vectorial finite-difference mode solvers for waveguide analysis, as well as a beam propagation method (BPM) and an eigenmode expansion method (EME) for simulating 2D and 3D photonic devices. The design tool supports the flexible definition of 2D waveguide cross-sections and 3D device layouts with realistic optical materials (dispersive, temperature-dependent, and others), widely customizable nonuniform meshing, and perfectly matched layers. Additionally, **VPIdeviceDesigner** facilitates the easy creation of compact simulation models for the designed waveguides and devices, enabling seamless integration with **VPIcomponentMaker Photonic Circuits** and foundry-specific photonics process design kits (PDK).

VPIcomponentMaker Photonic Circuits provides a focused modeling and simulation environment for integrated photonic and optoelectronic circuit design experts. It includes advanced libraries of circuit-level abstracted models that facilitate rapid prototyping of complex, large-scale structures independent of the dispersed technology process (Indium Phosphide, Silicon, Silicon Nitride, Polymer, or other materials).

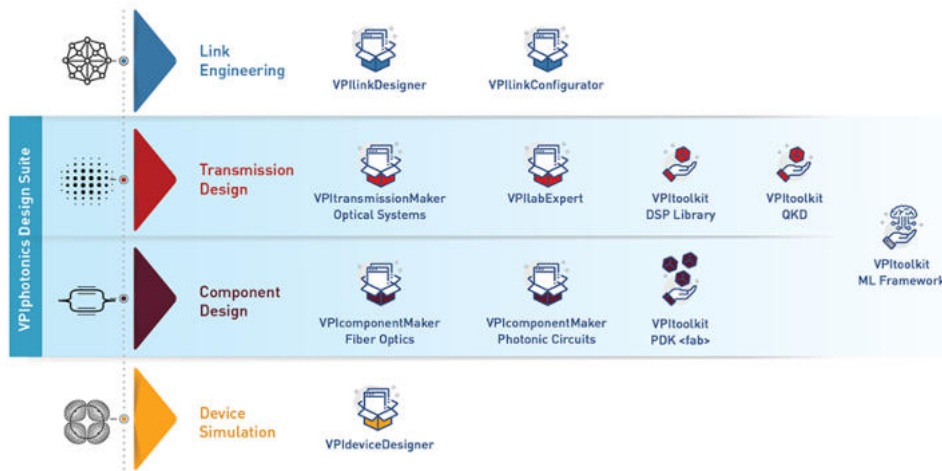


Figure 5.21: VPIphotonics software

The **VPI Transmission-Line Laser Model (TLLM)** handles the modeling of multisection semiconductor devices with bulk or MQW active mediums [1]. Addressed applications include buried-heterostructure lasers, amplifiers, electrooptic modulators, distributed Bragg reflectors, and others. The TLLM accounts for Kerr and two-photon absorption effects, spatial hole burning, measured gain and absorption spectra, and carrier dynamics.

Transmission-Line Laser Model (TLLM)

The S-matrix approach sustains the modeling of passive photonic and linear electrical elements such as passive waveguides, directional couplers, MMIs, star couplers, microring resonators, Bragg reflectors, grating couplers, AWGs, resistors, capacitors, inductors, and voltage and current sources. This method provides support for frequency-dependent effective mode indices and attenuation.

S-matrix approach

VPIcomponentMaker Photonic Circuits introduces several novel approaches for PIC simulations. One is the so-called hybrid time-and-frequency simulation domain [2]. The intelligent combination of time- and frequency-domain modeling techniques enables fast and accurate simulations of heterogeneous PICs, consisting of active and passive building blocks (BBs) frequently fabricated using different technologies. This approach greatly reduces the overall simulation inaccuracies inherent to time-domain simulations. It thus enables fast and accurate simulations of large-scale PICs with thousands of dispersive components of different length scales (from a few to thousands of microns on the same circuit).

hybrid time-and-frequency simulation

5.2.1.2 Foundry-specific and custom PDK libraries

To prototype application-specific PICs with prerequisite functionality using foundry-specific PDKs, VPIcomponentMaker Photonic Circuits can be extended with pluggable toolkits VPItoolkit PDK <fab>. These PDK toolkits include libraries of BBs for specific foundries and technologies (Indium Phosphide, Silicon, Silicon Nitride, Lithium Niobate, or Polymer), represented with adequate simulation models. The models are usually characterized with just a few user-controllable parameters so that the designer can complete the PIC design without knowing the details of the device layout and fabrication process.

VPItoolkit PDK

VPIcomponentMaker Photonic Circuits powered by VPItoolkit PDK <fab> add-ons en-

layout design rules
packaging
specifications
schematic capture

able another novel simulation technique called layout-aware schematic-driven PIC design methodology [3]. In this methodology, the circuit-level simulator supports the capability to specify exact physical locations and orientations of PDK BBs on the final layout (required, for instance, by layout design rules and packaging specifications) and to connect sub-circuits with fixed locations by special connector BBs, called smart elastic optical connectors. This method combines graphical schematic capture with automated waveguide routing by seamlessly integrating circuit and layout tools. Within this approach, VPIcomponentMaker Photonic Circuits automatically and invisibly for users invokes a layout design for elastic connectors to determine their actual physical lengths and shapes, constructs compact simulation models for them, and then initiates the circuit simulations.

At any design stage, PICs simulated with VPItoolkit PDK <fab> BBs can be automatically exported to various layout tools [3] to fit the layout to the die package, add proper electrical wire routing, perform design-rule-check (DRC) verification, and generate a GDS mask for circuit fabrication. Moreover, the foundry-specific PDK libraries with tolerance-aware compact simulation models allow for estimating fabrication variations and accurately predicting PIC performance. Finally, a custom PDK framework [4] enables the customization of the PIC designs using PDKs while ensuring intellectual property security.

References:

- [1] C. Arellano, S. Mingaleev, A. Novitsky, I. Koltchanov, and A. Richter, "Design of complex semiconductor integrated structures," Proc. SPIE 7631, Optoelectronic Materials and Devices IV, 76310K - 8 pages (2009).
- [2] S. Mingaleev, A. Richter, E. Sokolov, C. Arellano, and I. Koltchanov, "Towards an automated design framework for large-scale photonic integrated circuits," Integrated Optics: Physics and Simulations II (Vol. 9516, p. 951602), SPIE (2015).
- [3] S. Mingaleev, A. Richter, E. Sokolov, S. Savitzki, A. Polatynski, J. Farina, and I. Koltchanov, "Rapid virtual prototyping of complex photonic integrated circuits using layout-aware schematic-driven design methodology," In Smart Photonic and Optoelectronic Integrated Circuits XIX (Vol. 10107, pp. 23-36). SPIE (2017, February).
- [4] A. Polatynski, S. Savitski, C. Maloney, and A. Richter, "The best of both worlds: Design freedom and IP protection," PIC Magazine, Issue 1, 2024.

5.3 Computation Examples

The description in this section is not systematic, but built around a number of practical examples and organized per software provider. It assumes that the reader has some basic experience with how to run the programs. More detailed information for readers that are not yet familiar with the software can be found in the Appendices. There is an appendix with detailed explanations for each participating software vendor.

Table 5.4 shows the graphs from the Handbook for which computation examples are provided in the following sections.

Property	Graph	Photon Design	VPIPhotonics
Effective index of a slab guide	Figure 2.24		Page 5-36
Mode profile of a slab mode	Figure 2.27		Page 5-35
Effective index of a ridge waveguide	Figure 2.34		Page 5-39
Mode profile of metal-loaded waveguide	Figure 2.40		Page 5-38
Bending loss	Figure 2.44		Page 5-40
Taper loss	Figure 21.5	Page 5-26	
Y-junction loss	Figure 22.3	Page 5-27	
Star coupler loss	Figure 23.4	Page 5-29	
Star coupler reflection	Figure 23.5	Page 5.27	

Table 5.4: Overview of computation examples provided in this section.

5.3.1 Photon Design

Waveguide tapers. Figure 5.22 (i.e. figure 21.5 from Chapter 21) shows the insertion loss of a linear taper with a taper ratio 2 as a function of the taper angle and length, both for shallow and deep standard waveguides (with 2 and 1.5 μm waveguide width, respectively).

With the FIMMWAVE/ FIMMPROP tool the curves can be calculated as follows (see Figure 5.23). For a detailed description see Appendix D, Section D.1).

- Create a node for the InP **Waveguide Stack** with the following layers: top layer (air, $t=0$, $n=1$); top-cladding (InP, $t=0.5$, $n=3.17$); waveguide layer (Q1.25, $t=0.5$, $n=3.36$); substrate (InP, $t=1$, $n=3.17$). This is representative for the standard waveguide in the Smart Photonics platform, but slightly reduced to avoid unnecessary computations in the region without field.
- Create a node **Waveguide Taper** using **Add FIMMPROP Device**. Select the taper in the **Waveguide Taper** icon bar, and add access waveguides at both sides of the taper, and two joints. For the shallow waveguide taper set the taper input and output widths to 2 and 4 μm , respectively, and match the input and output waveguide widths. Use the **Device Options** to set for all sections the etch depth to 0.65 μm (150 nm into the waveguide) and the computation window width to 7 μm . Specify the Horizontal symmetry of the structure in the MOLAB options.
- Now we are going to sweep the taper length. Create a node **Taper Length** by clicking the **Add Variables** icon in the icon bar on top, define Taper_Length as scan parameter and enter the Variable name Taper_Length in the box **length** of the taper section.
- Finally create the node **Taper Length Scan** with the **FIMMPROP Scanner**. Set **paramName** to Taper_Length, **paramStart** and **paramEnd** to 0 and 30, **nstep** to 31 and start the computation. It will produce a graph of the transmitted power in the fundamental mode as a function of the taper length. Export the data to Excel. After converting the transmission coefficients to dBs, we get the curve **Shallow** in Fig. 5.22 (right).

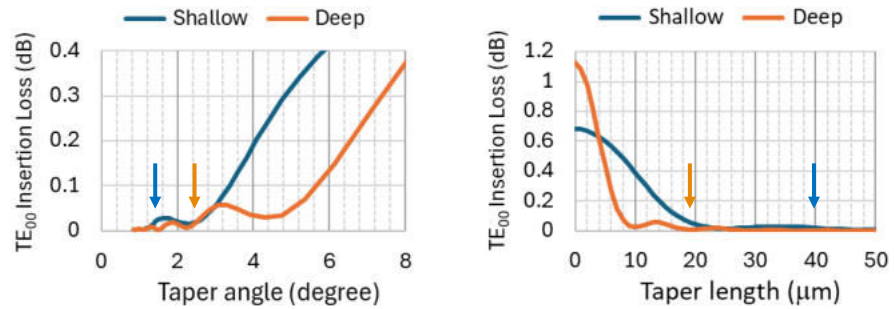


Figure 5.22: Insertion loss of linear tapers with varying length, for shallow and deep etched waveguides.

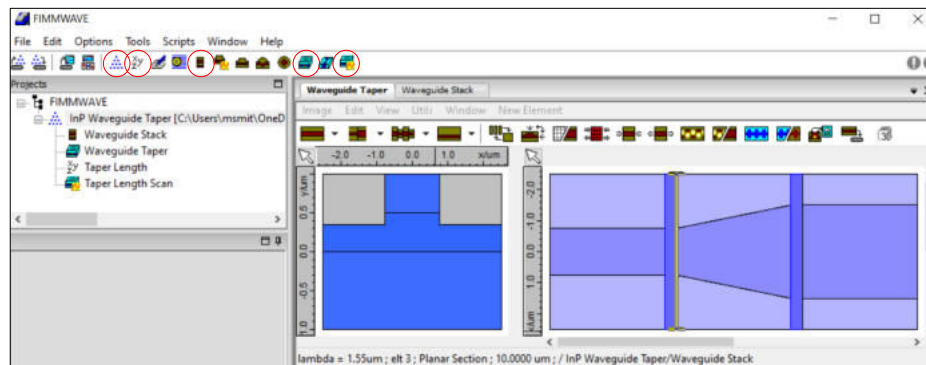


Figure 5.23: FIMMWAVE Projects panel.

- For analyzing a deep-etched taper we have to adapt the waveguide width in the left and right sections to 1.5 and 3 μm , respectively, and adapt the taper. In the **Device Options** menu we adapt the etch depth to 1.1 μm (0.1 μm into the substrate) and run the **Taper Length Scan** again. This brings us the curve labeled **Deep** in Fig. 5.22 (right).

For the deep etched taper we can improve the accuracy by increasing the number of modes to 20. Optimizing the computation mesh, the number of modes involved in the computations and some other parameters as described in Appendix D, Section D.1 and increasing the number of modes to 30 brings only a small additional improvement, so that the results shown can be considered as close to the final solution.

The left figure can be obtained by calculating the taper angles corresponding to the taper lengths and using the X,Y Scatter graph of Excel to plot both curves in one graph as a function of the the taper angle.

Y-junctions. Figure 5.24 (i.e. figure 22.3 from Chapter 22) shows the dependence of the excess loss and the reflection of a Y-junction on the gap closure and the junction angle.

With the FIMMWAVE/ FIMMPROP tool the curves can be calculated as follows.

- Create a node for the shallow InP **Waveguide Stack** as described in Section 5.3.1. In this example a top-cladding thickness of 1 μm is chosen.

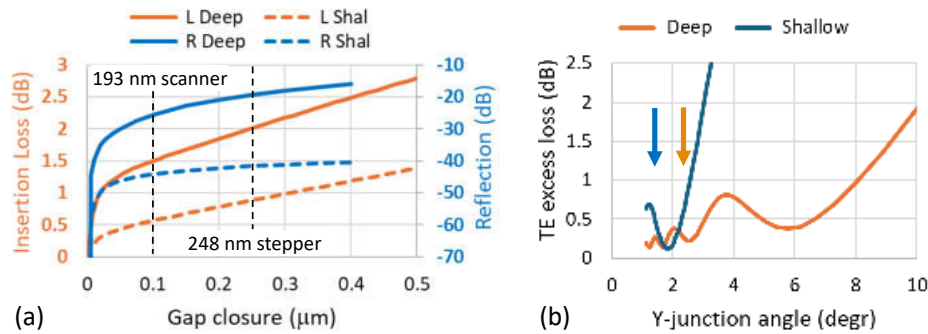


Figure 5.24: Excess Loss.



Figure 5.25: FIMMWAVE Projects panel

- Create a **FIMMPROP Device** as shown in Fig. 5.25, with two straight waveguide sections connected with a butt joint: a straight waveguide with a width of $4\ \mu\text{m}$ + the gap width and a twin waveguide with $2\ \mu\text{m}$ waveguide width and a gap in between ($4\ \mu\text{m}$ is the waveguide width at the end of a shallow 1:2 waveguide taper). In the **Device Options** set etch depth to $1.15\ \mu\text{m}$, the computation window width to $7\ \mu\text{m}$ and the length of both sections to $1\ \mu\text{m}$ (the length is not relevant for calculating the overlap). In the **MOLAB Options** click **Edit Solver parms** and set **Hsymmetry** to **ExSymm** and **meshType** to **Non-uniform** for X and Y.
- Define the gap width as variable and create a **FIMMPROP Scanner** as described in Section 5.3.1. With the scanner we can calculate the coupling efficiency (overlap) between the fundamental modes in the waveguides at both sides of the junction, as a function of the gap width, and the reflection at the junction. Scan the gap from 0 to $0.5\ \mu\text{m}$ in 50 steps and in the window **FP3D Mode Coefficients**, which appears when we click the scan button, we edit **inp DEV - out RHS norm(mode[1])** to overlap the mode of the Left Hand Section (default the fundamental mode) with the fundamental mode of the Right Hand Section. And we add **inp DEV - out LHS norm(mode[1])** to calculate the overlap of the reflected field with the fundamental mode of the Left Hand Section. For the coupling loss between the fundamental modes of both sections, we need only the fundamental modes in both sections (**maxNmodes=1** in the **MOLAB Options**). For the calculation of the reflection we need way more modes in the Right Hand Section. Convergence occurs with about 150 modes. With this number of modes the computation time is significant and the number of steps in the scan may have to be reduced.

- With 150 modes in the Left Hand Section, we find the curves in the left graph labeled **L Shal** and **R Shal**, after Exporting to Excel and converting to dB.
- For the deep etched waveguide reset the waveguide widths to 3 and 1.5 μm , and the computation Window to 6 μm , respectively, and repeat the procedure, this gives us the curves labeled **L Deep** and **R Deep**.
- For the calculation of the excess loss as a function of the Y-junction angle we use the the same **FIMMPROP device** as for analysing the effect of the gap closure. In the **Variables** node we set the Gap to zero and create a new variable **Length** for the twin guide section, which we enter in the box **length** using the **Edit properties ...** menu. In the twin guide section we set the **offsetType** to linear and vary it from **offsetLhs [um] = 1** to **offsetRhs [um] = 2**, which will vary the Y-junction gap from 0 to 1 μm along the section.
- Because we are now interested in the loss for the fundamental mode due to conversion to higher order modes in the twin guide section, we set the number of modes involved in the calculation to 10 for the twin guide section (using the **MO-LAB Options**).
- Next we scan the length of the twin guide section from 5 to 50 μm , which corresponds roughly to a Y-junction angle variation between 1 and 10 degrees. Exporting the data to Excel and converting the length axis to an angular axis gives us the curve in the right Figure, labeled **Shallow**.
- For the deep etched waveguide reset the waveguide widths to 3 and 1.5 μm , and the computation Window to 6 μm , respectively, and repeat the procedure. This gives us the curve in the right Figure, labeled **Shallow**.

In Figure (b) we see a fringe pattern at small angles, which suggests that there is beating between different modes. The design angles for deep and shallow waveguides according to Chapter 21 are indicated with brown and blue arrows. We do not expect such beating at these angles where the Y-junction should behave adiabatic, i.e., it should not show mode conversion. In Appendix D Sec. D.2, the origin of this unexpected behavior is discussed.

Transmission and reflection of a star coupler. Figure 5.26 (i.e. Fig. 23.4) shows the coupling loss and the reflection at the junction between the Free Propagation Region and the waveguide array of a star coupler, as a function of the width of the gap between the array waveguides at the junction (the gap closure). The curves are calculated for both a shallow and deep etched waveguide array.

With the FIMMWAVE/ FIMMPROP tool the curves can be calculated as follows. For a more detailed explanation see Appendix D, Sec. D.1.

- Create the node for the standard waveguide stack.
- Create the node for the variable Gap_Width.
- Create the node for the waveguide junction, using **Add FIMMPROP Device**. Create two straight waveguide sections with a junction in between. In the **Planar Section Editor** (select **Device Options**) define for the shallow etched waveguide $\text{psEtchDepth}=1.15 \mu\text{m}$ and for the width of the computation window $\text{psWidth}=2+\text{Gap_Width}$. For the boundary conditions choose Periodic Wall for

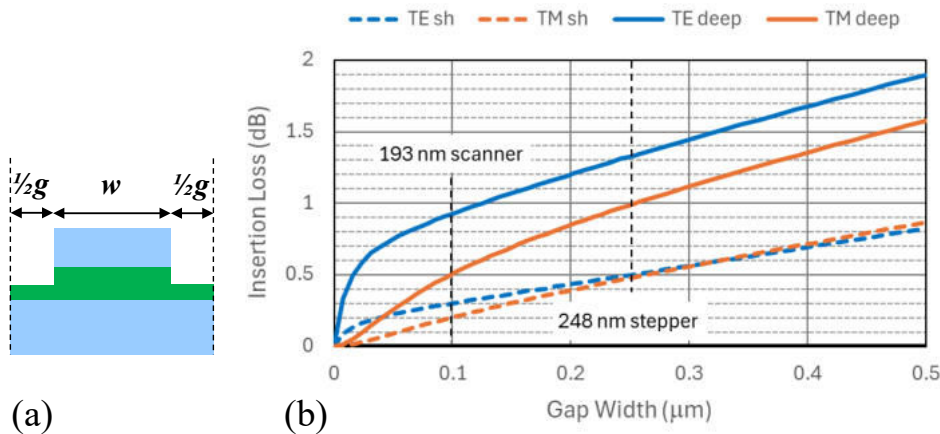


Figure 5.26: The insertion loss of TE- and TM-polarized modes at the junction between the Free Propagation Region and a deep and shallow etched waveguide array.

the $-x$ and $+x$ boundary. The y -boundaries are not relevant because there is no field. For the Device MOLAB options chose $\text{maxNmode}=2$ in order to do the calculations for both the TE_0 -mode ($N = 1$) and the TM_0 -mode ($N = 2$)

- For the Right Hand section set the waveguide width to $2\ \mu\text{m}$ in the Path Parameters window. Make sure that the offset is zero. For the Left Hand section set the waveguide width to $2\ \mu\text{m} + \text{Gap-Width}$, so that the waveguide fills the whole computation window. Together with the periodic boundary conditions this will cause the fundamental TE and TM modes in the Left Hand Section to be uniform in the computation window, corresponding to a plane wave incident on the array.
- Create a node for scanning the Gap_Width using the **FIMMPROP Scanner**. In the **Scanner Options** window select *Store entire S-matrix*. Select the scan parameters, e.g. 50 steps from 0 to $0.5\ \mu\text{m}$ and run the scanner. In the FP3D Mode coefficients window remove the default plot list, select for Device input LHS mode 1 and for output RHS mode 1 and add it to the plot list. This will yield the coupling efficiency from the plane TE wave to the TE_0 -mode in the (coupled) array waveguide. Next select LHS input mode 2 and RHS output mode 2 and add it to the plot list. This will yield the coupling efficiency for the TM-polarized wave. By clicking OK the scan will start and we can export the data to Excel to get the dashed curves in the Figure.

Coupling loss to the deep-etched array. To get the (solid) curves for the deep etched waveguide array we have to change the etch depth to $1.6\ \mu\text{m}$ in the **Device Options** sub-window of the **Junction** window and change the computation window width (psWidth) to $1.5 + \text{Gap_Width}$. Further we have to adapt the waveguide width of the LHS and the RHS sections of the junction to $1.5 + \text{Gap_Width}$ and $1.5\ \mu\text{m}$, respectively. For the calculation of the curves we need to account for the fact that for the $1.5\ \mu\text{m}$ wide deep etched waveguide the effective index of the TE_0 -mode is smaller than that of the TM_0 -mode, so that the TM_0 mode will be labeled $n = 1$ and the TE_0 mode $n = 2$. So if we calculate the coupling efficiency from LHS-mode 1 to RHS mode 1, we will find zero coupling. And the same holds for the coupling from LHS-mode 2 to RHS mode 2. To

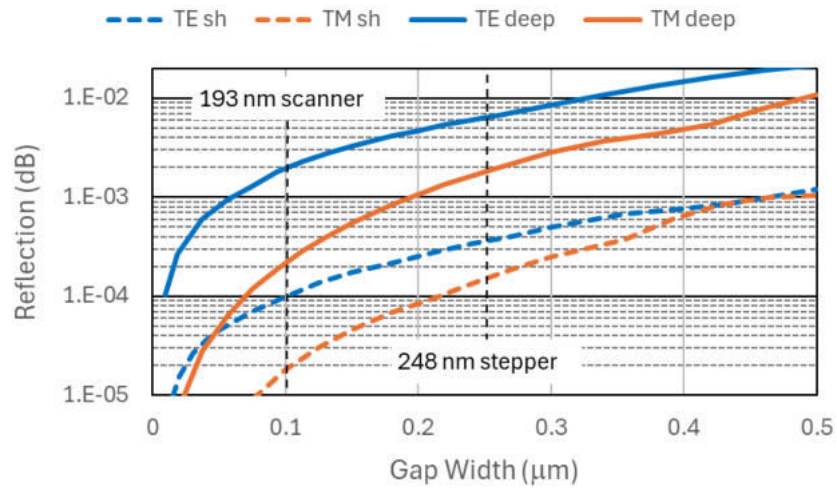


Figure 5.27: The reflection of TE- and TM-polarized modes at the junction between the Free Propagation Region and a deep and shallow etched waveguide array.

find the proper coupling efficiencies we have to calculate the coupling from LHS-mode 1 to RHS-mode 2 for TE-polarization, and from LHS-mode 2 to RHS-mode 1 from TM-polarization. However, for a gap width beyond 220 nm the LHS TE-mode gets the highest index and, consequently, the mode number 1, so that we find zero coupling from LHS-mode 1 to RHS-mode 2. To find the full curve for a gap width from 0 to 500 nm we have to calculate 4 curves: LHS[1]→RHS[1], LHS[1]→RHS[2], LHS[2]→RHS[1] and LHS[2]→RHS[2], and combine the non-zero parts of the 4 curves to the 2 curves shown in Fig. 5.26.

Reflection of the waveguide array. The calculation of the reflection at the junction between the Free Propagation Region and the waveguide array is analogous to the calculation of the coupling efficiency, but with the following modifications. Instead of calculating the coupling from LHS-mode 1 to RHS-mode 1 we now have to calculate the coupling from LHS-mode 1 to itself (but in the opposite direction). And similarly from LHS-mode 2 to itself for the reflection of the TM-polarized field. An important difference is the number of modes MaxNmode required at both sides of the junction. Convergence analysis (by stepwise increasing the number of modes) yields that at least 250 modes are required to get a stable output result. This will lead to a huge increase of the computation time, and it will be useful to reduce the number of points for which the curves are calculated, especially in the falt regions. We can reduce the computation time a bit further by enabling the *isStraight* flag in each of the section properties, which tells the solver that both sections are straight. This option is found in the bottom left corner of the **Planar Section Editor** window.

The results are shown in Fig. 5.27. From the fact that for gap widths beyond 300 nm the curves are not completely smooth we can see that the solver may still have some convergence problems.

5.3.2 VPIdeviceDesigner

Introduction

VPIdeviceDesigner is a commercial Python-based photonic device simulator. Simulations are usually implemented in Jupyter Notebooks, but can also be realized as plain Python scripts. The heart of VPIdeviceDesigner is the `vpipda` Python package, which contains classes, methods, and functions to define optical materials and layouts, run simulations, as well as post-process and visualize the results.

For the waveguide cross-section analysis, the usual simulation workflow is as follows:

- import all necessary packages
- define the materials
- define the waveguide cross-section layout
- solve for the waveguide modes
- analyze, post-process, and visualize the results

Here, we will illustrate the workflow with a very simple generic rib waveguide to give a general feeling of the usage of VPIdeviceDesigner. In the following subsections, which discusses the simulation of selected examples from this handbook, we will only focus on specific details of the respective simulation, but not repeat all steps. The complete simulation setups are described in detail in the appendix.

Usually, all `import` statements should be collected at the beginning of a Python file. Here, however, it will be easier to follow the code snippets when we import modules and classes just where they are needed.

The first step is usually to define the materials. Besides using a library of predefined materials, we can define our own material models. Here, we use constant refractive indices for all materials:

```
from vipda.material import Dielectric

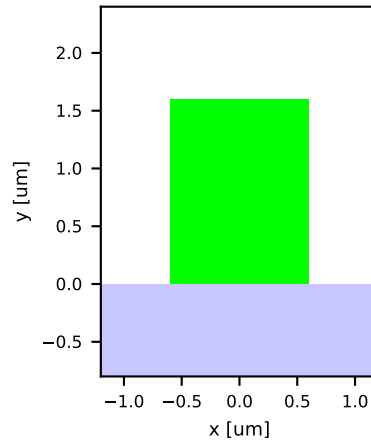
Air = Dielectric(n=1.0, name='Air', color='white')
InP = Dielectric(n=3.17, name='InP', color='#C8C8FF')
InGaAsP_Q125 = Dielectric(n=3.36, name='InGaAsP_Q125', color='#00FF00')
```

Next, we use these materials to define geometrical shapes and combine them into a cross-section. For example, a simple rib waveguide could be defined as follows:

```
from vipda.layout import Rectangle, HalfPlane, Layout2D

sub = HalfPlane(InP)
core = Rectangle(InGaAsP_Q125, width='1200 nm', height='1600 nm',
                 bottom=sub.center)
xs = Layout2D(sub, core)

xs.plot()
```



The last step before the mode calculation can be started is the definition of the mesh. VPIDeviceDesigner uses finite-difference-based methods. Hence, the mesh will be a rectangular grid.

```
from vpipda.mesh import Mesh2D

mesh = Mesh2D(xs)
mesh.box = xs.box.expand('1.5 um')
mesh.update(dx='50 nm', dy='50 nm')
```

Now we can initialize the mode solver and calculate, for example, the first two eigenmodes at $\lambda = 1550\text{nm}$:

```
from vpipda.mode import SolverModeFDM # FDM = Finite Difference Method

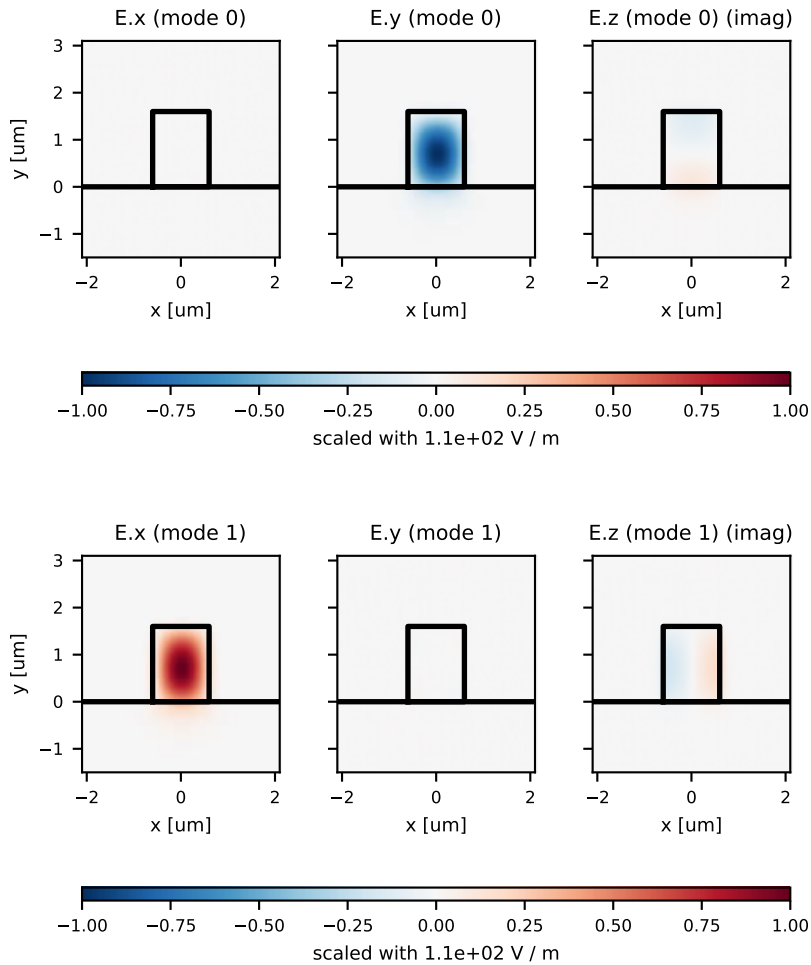
solver = SolverModeFDM(xs, mesh)
modes = solver.calc(freq='1550 nm', nmodes=2)
```

After the modes are calculated, they are returned as feature-rich Python objects. For example, we can evaluate their effective indices and plot their field profiles:

```
print(f'neff(0) = {modes[0].neff()}')
print(f'neff(1) = {modes[1].neff()}')
```

```
neff(0) = 3.282931022386663
neff(1) = 3.2739634864628937
```

```
modes[0].E.plot(contour=False, logscale=False)
modes[1].E.plot(contour=False, logscale=False)
```



It is important to note that this book and VPIdeviceDesigner use different coordinate system conventions. In this book, the coordinate system for waveguides is defined as follows:

- z is the propagation direction.
- x is the vertical axis, perpendicular to the layer interfaces.
- y is the lateral axis, parallel to the layer interfaces and orthogonal to z .

In VPIdeviceDesigner, however, a different convention is used:

- z is also the propagation direction.
- x is the lateral axis, parallel to the layer interfaces and orthogonal to z .
- y is the vertical axis, perpendicular to the layer interfaces.

The simulations shown in this section will be done in the VPIdeviceDesigner coordinate system. For the resulting plots, however, we will relabel the axes to match the convention used in this book.

Calculation of slab waveguide modes

One of the most canonical problems in optical waveguide theory is the mode analysis of slab waveguides. Here we show how to calculate the modes of the standard waveguide structures described in this handbook, as shown in Fig. 5.28 from Chapter 2.

The task is to calculate the guided modes of a *slab waveguide*. We can follow the generic simulation workflow described in the introduction of E, with the following modification:

1. Slab waveguides are essentially one-dimensional because their layer-based geometry is invariant in the lateral direction. In VPIdeviceDesigner, we realize that by using only `HalfPlane` and `Layer` objects to define the geometry. The mode solver will automatically detect that the cross-section is one-dimensional and select a suitable solver.
2. It will be convenient to encapsulate the waveguide and mesh creation into a function that returns a waveguide cross-section and the associated mesh for any desired layer thickness and cover material:

```
from vpipda.layout import Layer

def create_slab(thickness='500 nm', cover=Air):
    substrate = HalfPlane(InP)
    core = Layer(InGaAsP_Q125, height=thickness, bottom=substrate.center)
    wg = Layout2D(cover, substrate, core)

    mesh = Mesh2D(wg)
    mesh.box = wg.box.expand('10 um')
    mesh.update(dy='10 nm')

    return wg, mesh
```

We can pass this waveguide-creating function directly to `SolverModeFDM`. It recognizes the available parameters so that we can modify them just before the mode calculation:

```
solver = SolverModeFDM(create_slab)

modes_sym = solver.calc(freq='1550 nm', nmodes=2, cover=InP)
modes_asy = solver.calc(freq='1550 nm', nmodes=2, cover=Air)
```

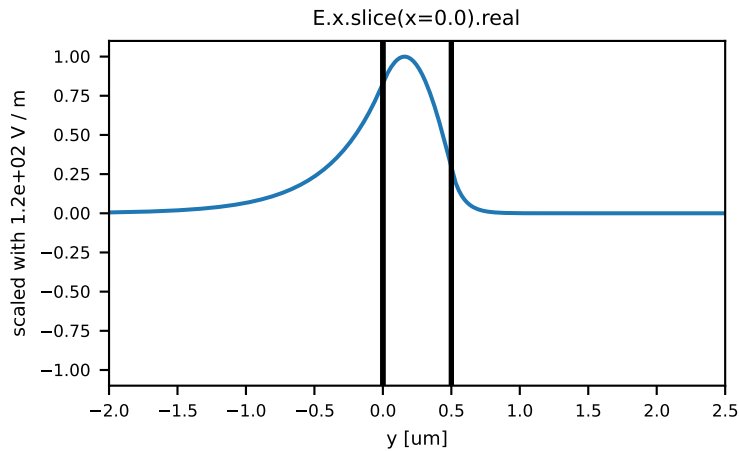
As an example, we analyze the asymmetric waveguide's TE mode:

```
print(f'neff(asy, TE) = {modes_asy[0].neff()}')
```

```
neff(asy, TE) = 3.2305315372468204
```

The modes' fields are feature-rich Python objects and are equipped with their own plot method. Therefore, the field visualization is straight forward. For example:

```
from vpipda.layout import Box1D
modes_asy[0].E.x.slice(x=0).real.plot(edges=True, box=Box1D(-2, 2.5))
```



As the plot functionality is based on `matplotlib`, we can use additional `matplotlib` features to create a publication-ready figure (as shown in figure 5.28) from the results in any desired layout (see Appendix E for the complete code).

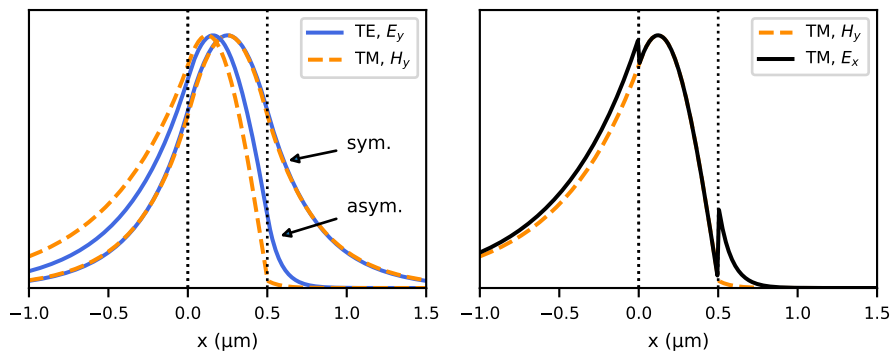


Figure 5.28: Mode profiles in semiconductor slab waveguides, calculated with `VPIDeviceDesigner`.

Propagation constant and effective index in a slab guide. The task is to conduct a *layer thickness sweep* for the slab waveguides defined above. The `create_slab` function is parameterized by thickness. Hence, we can just pass an array of desired thicknesses to `solver.calc(...)` and it will perform the parameter sweep automatically:

```
import numpy as np
thicknesses = np.linspace(1.11, 0.01, 51)

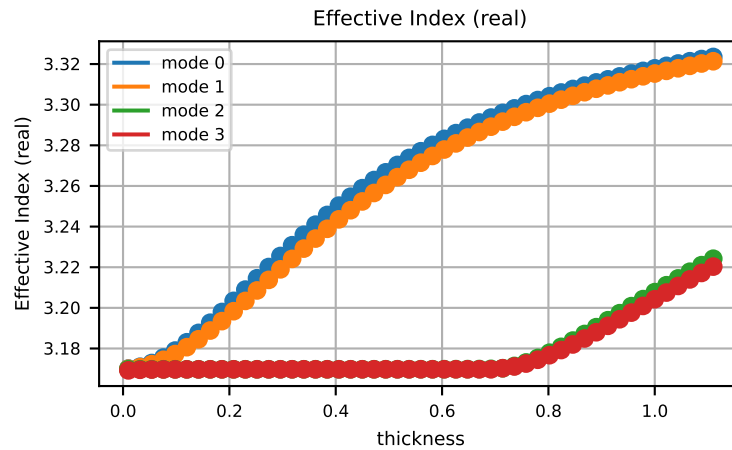
modes_InP = solver.calc('1550 nm', nmodes=4,
                        thickness=thicknesses, cover=InP)
```

Modes calculation: 100%|=====| 51/51 [00:03<00:00, 15.44point/s]

The modes' effective index property comes with its own plot method, which is compatible with the standard `matplotlib` plots:

```
import matplotlib.pyplot as plt

fig, ax = plt.subplots()
for m in modes_InP:
    m.neff.plot(fig=ax, label='mode ' + str(m.name))
```



The modes 0 and 1 are the fundamental TE and TM modes. The modes 2 and 3 are higher-order modes. Below 700 nm thickness, they are cut off. Due to the finite simulation domain, they turn into discrete cladding modes.

With some additional matplotlib code we can create a publication-ready figure (as shown in figure 5.29) from that result in any desired layout.

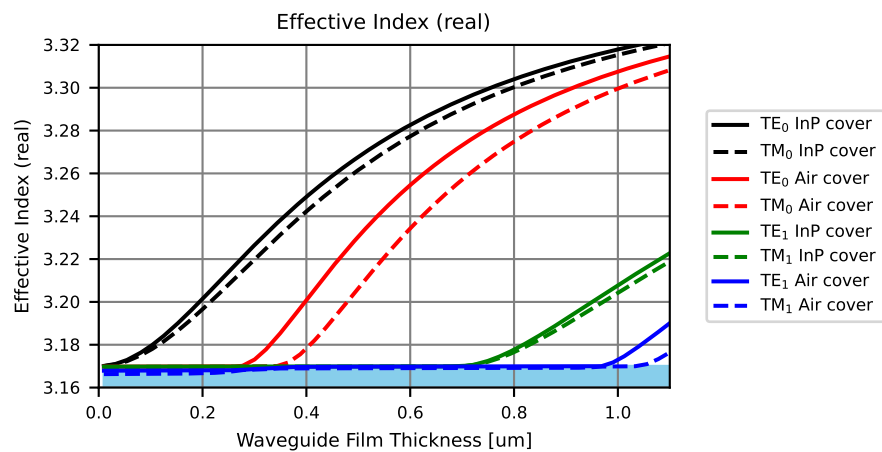


Figure 5.29: Effective indices of the first four modes in a standard slab waveguide with or without InP cover layer, as a function of the Q1.25 layer thickness, calculated with VPIdeviceDesigner.

Mode profiles and propagation loss in lossy waveguides. The investigated structure is again a slab waveguide. To model the lossy materials, we simply define materials with complex refractive indices and use them in the layer stack.

```
InGaAs = Dielectric(n=3.7434-0.287j, name='InGaAs', color='#FF0000')
Ti = Dielectric(n=3.7-4.5j, name='Ti', color='#FFFF00')
```

In order to investigate the modes' absorption in dependence on the distance to the metal layer, we again define a waveguide-generating function, parameterized by the buffer layer thickness:

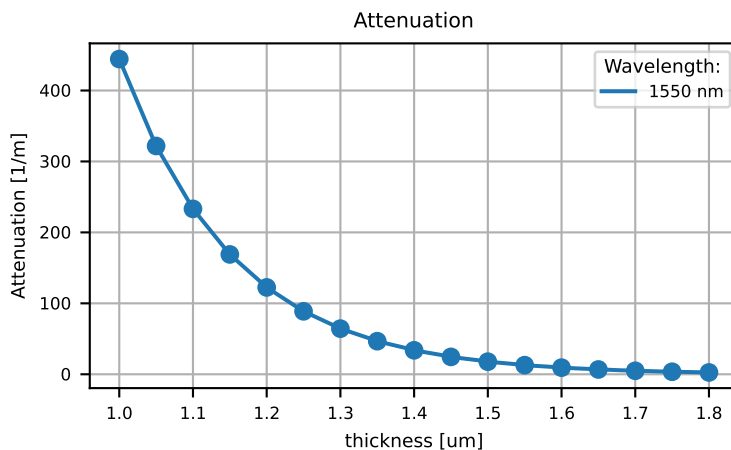
```
def create_slab_with_metal(thickness='1.5 um'):
    ...
    buffer = Layer(InP, height=length2default(thickness))
    ...
    wg = Layout2D(..., buffer, ...)
    mesh = Mesh2D(wg)
    ...
    return wg, mesh

solver = SolverModeFDM(create_slab_with_metal)

buffer_thicknesses = np.linspace(1.0, 1.8, 17)*u.um
modes_vs_thickness = solver.calc(wl, nmodes=2,
                                thickness=buffer_thicknesses, ...)
```

The further simulation workflow is identical to the above illustrated simulations, which allows us to focus here on the results. The propagation loss can be conveniently accessed as a property of the respective mode object:

```
modes_vs_thickness[0].attenuation.plot()
```



Please see the appendix for the complete Python code. The desired figure design (see figure 5.30) is more elaborate, but `matplotlib` (with the supporting code shown in the appendix) readily supports its creation, and conceptually it does not differ from the previous examples.

Effective indices in a 2D waveguide

The task is to conduct a *waveguide width sweep* for a *channel waveguide*. VPI-deviceDesigner uses full-vectorial finite-difference 2D mode solvers to calculate the

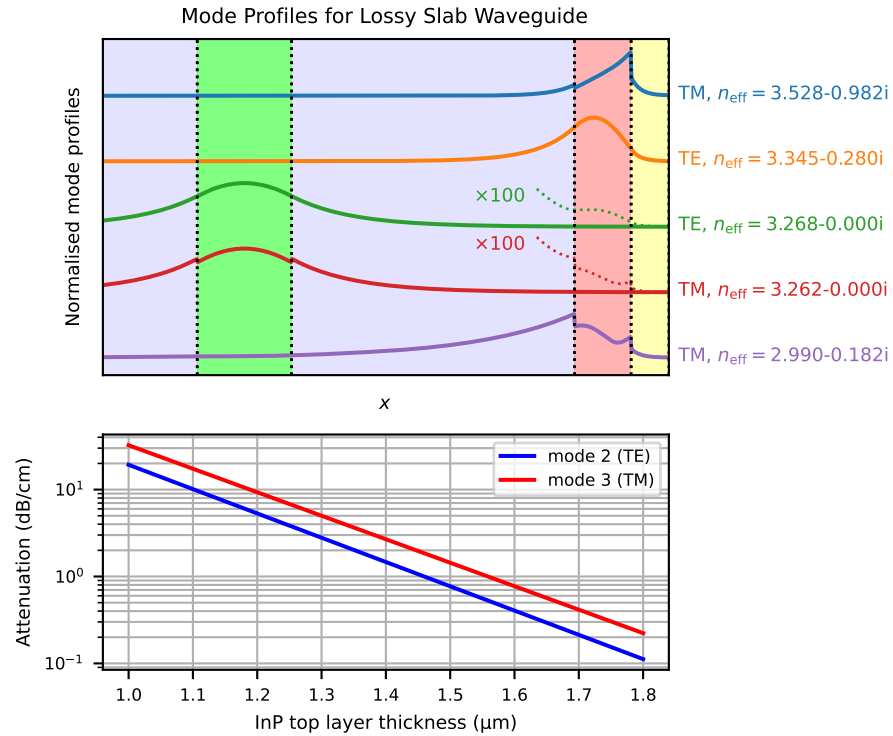


Figure 5.30: Mode profiles in a waveguide with metal electrodes, calculated with VPI-deviceDesigner.

waveguide modes without approximations (except for the discretization and the finite simulation domain). In contrast to slab waveguide examples above, we use additionally Rectangle shapes to create the ridge waveguide cross-section. The mode solver will automatically detect that the cross-section is two-dimensional and select a suitable solver. Please see the appendix for the complete definition of the parameterized waveguide-generating function that can create deep- and shallow-etched waveguides with variable width. The further workflow is exactly as described above.

Figure 5.31 shows how the modes' effective indices depend on the respective waveguide widths for the deep- and shallow-etched channel waveguides.

Propagation loss in curved waveguides. The mode solvers in VPIdeviceDesigner natively support the calculation of modes in bent waveguides by transforming the waveguide cross-section to a cylindrical coordinate system. This is achieved by simply adding the special argument `bend_radius=...` to the `solver.calc(...)` method.

Modes in bent waveguides will be intrinsically leaky and radiate a small fraction of the propagating power into the substrate or cladding. This makes it necessary to have a closer look at the boundary conditions of the simulation domain. So far, we have used the default boundary conditions (which are perfectly conducting boundaries, i.e. they enforce $\vec{E}_{\text{tangential}} = 0$), which perfectly reflect all the incident waves. To avoid undesired back-reflections of the radiated waves, we need to add the so-called perfectly matched layers, which are reflection-less, strongly absorbing artificial layers (PMLs) in front of the simulation domain boundaries. They attenuate all incident waves to a neg-

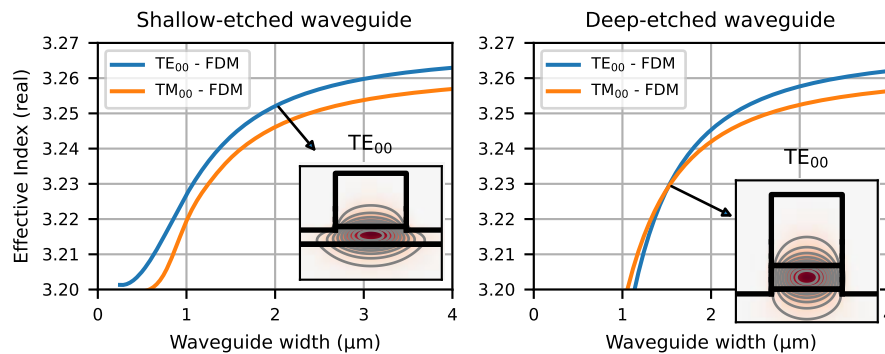


Figure 5.31: Effective indices of the fundamental modes in the shallow- and deep-etched waveguides, calculated with VPIDeviceDesigner.

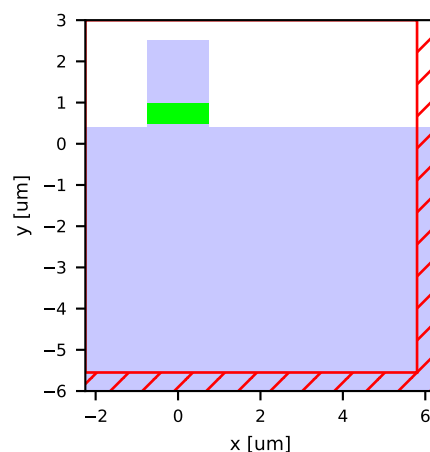
ligible amplitude before they reach the simulation domain boundaries. Assuming that we already defined a `create_channel(...)` function, we can add PMLs as follows:

```
def create_channel_pml(width, deep=True):
    wg, mesh = create_channel(width=width, deep=deep)

    mesh.box = mesh.box.expand((0, 4), (4, 0))
    mesh.set_boundary('bottom', 'PML', depth=10, damping=1e-8)
    mesh.set_boundary('right', 'PML', depth=10, damping=1e-8)
    mesh.update(dx=0.05, dy=0.05)

    return wg, mesh
```

```
wg, mesh = create_channel_pml(width=1.5)
mesh.plot(grid=False)
```



The default bending direction in VPIDeviceDesigner is to the left and we can expect the bent mode to radiate into the substrate due to the higher refractive index. Hence, it is sufficient to add PMLs to the bottom and right boundaries. The bending losses will increase with smaller bend radii (i.e. larger curvature). At the same time, the mode

field center will shift to the right and the effective mode index increases. It will help with tracking the fundamental modes if we sweep the bend radius from large to small values:

```
bend_radii_deep = np.linspace(80, 10, 15)
```

The remaining simulation workflow is standard. Once the modes are calculated and stored, for instance in the variable `modes_bent_deep`, the attenuation in dB over a 90° bend can be calculated according to

$$a = -10 \log_{10}(e) \frac{4\pi}{\lambda_0} \operatorname{Im}(n_{\text{eff}}) \frac{\pi}{2} R.$$

For example:

```
a_TE_deep = -10 / np.log(10) * (4 * np.pi / wl) * \
    modes_bent_deep[1].neff.imag() * (np.pi / 2) * bend_radii_deep
```

The results for the TM mode of the deep-etched and the TE and TM modes of the shallow-etched waveguide are calculated in the same way. Finally, the figure 5.32 can be composed with standard `matplotlib` features.

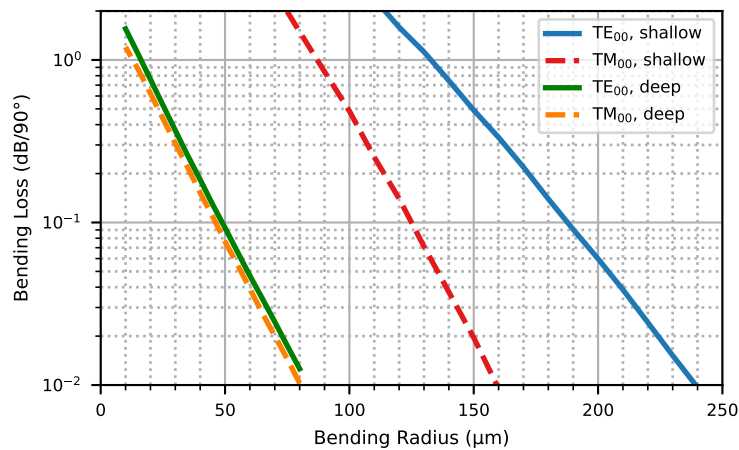


Figure 5.32: Bending loss for the fundamental modes in standard shallow and deep etched waveguides, calculated with `VPIdeviceDesigner`.

Mode profiles in asynchronous coupled waveguides. The illustrated asynchronous coupled waveguides can be viewed as a 1D approximation of two coupled channel waveguides. One important difference to the previous 1D simulation is that the invariance is now in the vertical direction. This translates to `VPIdeviceDesigner` as a 90° rotation of the core layers, e.g.:

```
core = Layer(InGaAsP_Q125, height=..., rotation='90 deg')
```

Apart from that, the cross-section can be assembled as usual. This time, it is y invariant. Again, a suitable mode solver will be chosen automatically. The mode solver will return the system modes (or super modes) of the coupled waveguide. They can be analyzed further (e.g. calculate $\Delta\beta$) and visualized as usual (see figure 5.33).

It would also be possible to calculate the individual core's modes and use them for a coupled-mode analysis.

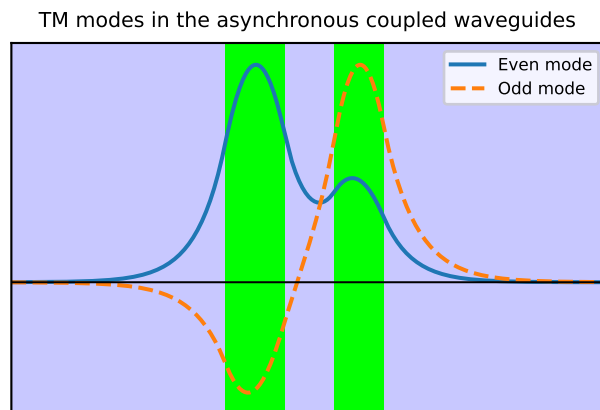


Figure 5.33: System modes in asynchronous coupled waveguides, calculated with VPIdeviceDesigner.